MediaHub:
Bayesian Decision-making
in an Intelligent Multimodal Distributed Platform Hub

Glenn G. Campbell, B.Eng. (Hons.) (University of Ulster)

School of Computing & Intelligent Systems
Faculty of Computing & Engineering
University of Ulster

A thesis submitted in partial fulfilment of the requirements for
the degree of Doctor of Philosophy

April, 2009

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my appreciation to Dr. Tom Lunney and Prof. Paul Mc Kevitt for their continuous advice, support and guidance throughout my Ph.D. work. Their knowledge and expertise in distributed processing, intelligent multimedia and multimodal systems has made an immense contribution to my research. Thanks to Aiden Mc Caughey for all his advice and assistance with Java programming. I also want to express gratitude to other Ph.D. student members of the Intelligent Systems Research Centre (ISRC) at Magee, Jonathan Doherty, Rosaleen Hegarty and Sheila McCarthy, who were always willing and able to share their knowledge of IntelliMedia. Additionally, I would like to thank Dr. Minhua (Eunice) Ma for her advice on semantic representation and Dr. Tony Solon for engaging in numerous meetings and discussions on multimodal decision-making. I wish to express sincere appreciation to various members of the ISRC who frequently offered valuable feedback, criticisms and advice on my research, in particular Dr. Caitríona Carr, Bryan Gardiner, Dr. Pawel Herman, Simon Johnston, Dermot Kerr, Fiachra MacGiolla-Bhride and Dr. Tina O'Donnell. Their friendships and solidarity have been invaluable to me throughout the duration of my Ph.D. work. Thanks to Frank Jensen for his assistance with the Hugin software tools, Dr. Thor List and Dr. Kristinn Thórisson for their valuable help with Psyclone and the OpenAIR specification. Many thanks to Kristiina Jokinen and Jim Larson for their advice at the Elsenet Summer School 2007. Thanks to Prof. Pádraig Cunningham for his comments at AICS 2006. Thanks to Dr. Kevin Curran, Prof. Sally McClean, Prof. Bryan Scotney and Prof. Mike McTear who provided valuable feedback on my Ph.D. work and to Dr. Philip Morrow for his assistance. Thanks also to Pat Kinsella, Ted Leath, Paddy McDonough and Bernard McGarry for their technical support. I want to express my appreciation to Margaret Cooke and Heather Law (now at the University of Ulster Research Office) at the Faculty of Computing and Engineering Graduate School and also to Eileen Shannon at the University of Ulster Research Office for all their assistance. Thanks to Annemarie Doohan, Barry Harper, Lee Tedstone and Dr. Ramesh Kanagapathy of Nvolve Limited for their practical support that enabled the submission of this thesis.

I want to offer sincere thanks to my parents, Imelda and Joe, to my brothers, Joe and Seamus, and my sister, Katrina for their unending encouragement and support. Thanks to Jim McGrath for all his advice and to many other friends and relations who took an interest in my research and encouraged me along the way. Last, but not least, I want to express my appreciation to my wife, Leona for her continuous patience, encouragement and support throughout my research.

Abstract

Intelligent multimodal systems facilitate natural human-computer interaction through a wide range of input/output modalities including speech, deictic gesture, eye-gaze, facial expression, posture and touch. Recent research has identified new ways of processing and representing modalities that enhance the ability of multimodal systems to engage in intelligent human-like communication with real users. As the capabilities of multimodal systems have been extended, the complexity of the decision-making required within these systems has increased. The often complex and distributed nature of multimodal systems has meant that the ability to perform distributed processing is a fundamental requirement in such systems. The hub of a multimodal distributed platform must interpret and represent multimodal semantics, facilitate communication between different modules of a multimodal system and perform decision-making over input/output data.

This research has investigated distributed processing and intelligent decision-making within multimodal systems and proposes a new approach to decision-making based on Bayesian networks within the hub of a multimodal platform. The thesis is demonstrated in a test-bed multimodal distributed platform hub called *MediaHub*. MediaHub performs Bayesian decision-making for semantic fusion and addresses three key problems in multimodal systems: (1) semantic representation, (2) communication and (3) decision-making. MediaHub has been tested on a number of problems such as anaphora resolution, domain knowledge awareness, multimodal presentation, turn-taking, dialogue act recognition and parametric learning across a series of application domains such as building data, cinema ticket reservation, in-car information and safety, intelligent agents and emotional state recognition. Evaluation of MediaHub gives positive results which highlight its capabilities for decision-making and it is shown to compare favourably with existing approaches. Future work includes the integration of MediaHub with existing multimodal systems that require complex decision-making and distributed communication, and the automatic population of Bayesian networks.

Notes on access to contents

Chapter 1 Introduction

Multimodal systems provide the potential to transform the way in which humans communicate with machines. Already there have been significant advances towards the goal of achieving natural human-like interaction with computers (Bunt et al. 2005; López-Cózar Delgado & Araki 2005; Maybury 1993; Mc Kevitt 1995/96; Stock & Zancanaro 2005; Thórisson 2007; Wahlster 2006). Speech is a common form of communication between humans and computational devices (Mc Tear 2004). Of course, speech is just one modality that humans use to communicate. We use a vast array of modalities to interact with each other, including gestures, facial expressions, gaze and touch. In order to realise truly natural human-computer interaction, there is a need to design multimodal systems that can process these modalities in intelligent and complementary ways. Such systems must be flexible, enabling the user to choose the interaction modalities. They must adapt to the changing needs of a dialogue with the user, accessing various modalities as required. Communication must not be restricted to a particular modality, but use a wide range of potential modalities. Multimodal systems must also facilitate communication through a combination of modalities in parallel (e.g. speech and gesture, speech and gaze) and be able to adapt their output to suit both the current context and the needs and preferences of the user. A more natural form of human-machine interaction has resulted from the development of systems that facilitate multimodal input such as natural language, eye and head tracking and 3D gestures.

1.1. *Overview of multimodal systems*

With respect to multimodal systems, of particular importance are their methods of semantic representation (Mc Kevitt 2005, Ma & Mc Kevitt 2003), semantic storage (Thórisson et al. 2005), and decision-making, i.e., semantic fusion and synchronisation (Wahlster 2006). Ymir (Thórisson 1996, 1999) is a multimodal architecture for creating autonomous creatures capable of human-like communication. A prototype interactive agent called Gandalf has been created with the Ymir architecture (Thórisson 1996, 1997). Gandalf is capable of fluid turn-taking and dynamic sequencing. Chameleon (Brøndsted et al. 2001) is a platform for the development of intelligent multimedia applications. In Chameleon, communication between modules is achieved by exchanging semantic representations via a blackboard. SmartKom (Wahlster 2006) is a multimodal dialogue system that deploys rule-based pre-processing together with probabilistic based decision-making in the form of a stochastic model. SmartKom primarily focuses on three application domains: home, e.g., interfacing with home entertainment; public, e.g., tourist information, hotel reservations, banking; and mobile, e.g., driver interaction with mobile services in the car. SmartKom deploys a combination of speech, gestures and facial expressions to facilitate a more natural form of human-computer interaction, allowing face-to-face interaction with its conversational agent, Smartakus. Interact (Jokinen et al. 2002) aids the creation of agent-based distributed systems. Agents are scored with regard to their suitability in performing particular tasks and an Interaction Manager deals with interactions between Interact modules. An early application of Interact was an intelligent bus-stop that enables multimodal access to city transport information. MIAMM (Reithinger et al. 2002) is an abbreviation for Multidimensional Information Access using Multiple Modalities. The aim of MIAMM is to develop new concepts and techniques that will facilitate fast and natural access to multimedia databases through multimodal dialogues. Considerable work has also been conducted on semantic representation within multimodal systems. Approaches to representing semantics include frames, typed feature structures, melting pots and XML (Mc Kevitt, 2005).

1.1.1. Distributed processing

Advances in the field of distributed processing have seen the emergence of various software tools that aid the design of distributed systems. Psyclone (Thórisson et al. 2005) is a powerful and robust message-based middleware that enables the development of large distributed systems. Psyclone facilitates a publish-subscribe mechanism of communication, where a message is routed through one or more central whiteboards to modules that have subscribed to that message type. Psyclone implements the OpenAIR (Mindmakers 2009; Thórisson et al. 2005) routing and communication protocol and enables the creation of single- and multi-blackboard based Artificial Intelligence (AI) systems. The Open Agent Architecture (OAA) (Cheyer et al. 1998; OAA 2009) is a framework for developing distributed agent-based applications. .NET (MS .NET 2009) is the Microsoft Web services strategy that enables applications to share data across different operating systems and hardware platforms. The Web services provide a universal data format that enables applications and computers to communicate with each other. Based on XML, the Web services facilitate communication across platforms and operating systems, irrespective of what programming language is used to write the applications. Other tools for distributed processing include CORBA (CORBA 2009; Vinoski 1993), an architecture for developing distributed object-based systems, and DACS (Distributed Applications Communication System) (Fink et al. 1996), a software tool for system integration that provides useful features for the development and maintenance of distributed systems.

1.1.2. Bayesian networks

Bayesian networks (Bayes nets, Belief networks, Causal Probabilistic Networks (CPNs)) (Pearl 1988; Charniak, 1991; Jensen 1996, 2000; Jensen & Nielsen 2007; Pourret et al. 2008) are an AI technique for reasoning about uncertainty using probabilities. There are a number of properties of Bayesian networks that make them suited to modelling decision-making within multimodal systems. Bayesian networks are appropriate where there exist causal relationships between variables of a problem domain, but where uncertainty forces the decision-maker to describe things probabilistically, e.g., where a multimodal system is 75% sure that the user is happy because the person is believed to be smiling. Becoming more prevalent over the last two decades, Bayesian networks have been applied in a number of application scenarios including medical diagnosis (Peng & Reggia, 1990; Milho & Fred, 2000), story understanding (Charniak & Goldman, 1989) and risk analysis (Agena 2009). A key advantage of Bayesian networks is their ability to perform intercausal reasoning, i.e., evidence supporting one hypothesis explains away competing hypotheses. Several software tools facilitate development of Bayesian networks, with many offering both a Graphical User Interface (GUI) and a set of Application Programming Interfaces (APIs). Popular Bayesian software includes Hugin (2009), MSBNx (Kadie et al. 2001, MSBNx 2009), Elvira (2009) and the Bayes Net Toolbox (Murphy 2009).

1.2. *Objectives of this research*

The key objectives of the research are summarised as follows:

- Develop a Bayesian approach to decision-making, i.e., fusion and synchronisation of multimodal input and output data.
- Implement and test this Bayesian approach within MediaHub, a multimodal distributed platform hub, with a generic approach to decision-making over multimodal data.
- Generate/interpret semantic representations of multimodal input/output within MediaHub.
- Coordinate communication between the modules of MediaHub and between MediaHub and external modules.
- Design, implement and evaluate MediaHub.

In pursuing these objectives several key problems in multimodal systems are addressed including anaphora resolution, domain knowledge awareness, multimodal presentation, turn-taking, dialogue act recognition and parametric learning. MediaHub is tested in a number of different application domains including building data, cinema ticket reservation, in-car information and safety, intelligent agents and emotional state recognition. The testing of MediaHub across a range of application domains demonstrates the breadth of its applicability in decision-making over multimodal data. The focus here is to demonstrate the breadth of MediaHub's decision-making, as opposed to its depth in any specific application domain.

1.3. *Outline of this thesis*

This thesis consists of seven chapters. Chapter 2 reviews related research and a number of concepts that are fundamental to multimodal systems, including semantic representation and storage, communication and the fusion and synchronisation of multimodal input/output data, i.e., decision-making. The chapter includes a discussion on tools for distributed processing and a review of existing multimodal platforms and systems. Intelligent multimedia agents are discussed and available multimodal corpora and annotation tools are reviewed. Also considered are the problems of dialogue act recognition and anaphora resolution.

Chapter 3 includes a detailed discussion of Bayesian networks and their application to decision-making. A definition and discussion of the history of Bayesian networks is provided initially before the structure of Bayesian networks is discussed. The ability of Bayesian networks to perform intercausal reasoning is given particular attention. Chapter 3 also considers the key problems that are encountered when constructing Bayesian networks and highlights their advantages over other approaches to decision-making. Applications of Bayesian networks are then discussed and a review of their current usage in multimodal systems is given. Chapter 3 concludes by reviewing existing software for implementation of Bayesian networks.

Chapter 4 presents a Bayesian approach to decision-making in a multimodal distributed platform hub. First, a generic architecture for a multimodal platform hub is given. This is followed by a discussion on the nature of decision-making and the key problems that arise within multimodal decision-making. The decisions are grouped into two areas: (1) synchronisation of multimodal data and (2) multimodal data fusion. Also considered are semantic representation, multimodal fusion and ambiguity resolution. Next, the features of a multimodal system that support decision-making including distributed processing, dialogue history, domain-specific information and learning are discussed. Following this, a list of necessary and sufficient requirements criteria for an intelligent multimodal distributed platform hub is compiled. Finally, the chapter ends with a discussion on the rationale for a Bayesian approach to decision-making within a multimodal distributed platform hub.

Chapter 5 details MediaHub, a multimodal distributed platform hub for Bayesian decision-making over multimodal input/output data. First, MediaHub's architecture is presented and its modules described in detail. Semantic representation and storage within MediaHub is addressed, before the role of Psyclone (Thórisson et al. 2005), which facilitates distributed processing in MediaHub, is described. Next, MediaHub's five decision-making layers are outlined: (1) *psySpec* and Contexts, (2) Message Types, (3) Document Type Definitions (DTDs), (4) Bayesian networks and (5) Rule-based. Following this, the functionality of Hugin (Jensen 1996) in implementing Bayesian networks for decision-making in MediaHub is discussed. Multimodal decision-making in MediaHub is then demonstrated through worked examples that investigate key problems in various application domains.

Chapter 6 discusses evaluation of MediaHub. First, hardware and software specifications of test environment systems are discussed. Next, initial testing of MediaHub is outlined, including NetBeans IDE, Hugin GUI and Psyclone's *psyProbe*. Then, the results of testing MediaHub on six key problems in multimodal decision-making are presented. The six problem areas considered are: (1) anaphora resolution, (2) domain knowledge awareness, (3) multimodal presentation, (4) turn-taking, (5) dialogue act recognition and (6) parametric learning across five application domains: (1) building data, (2) cinema ticket reservation, (3) in-car information and safety, (4) intelligent agents and (5) emotional state recognition. Next, the performance and potential scalability of MediaHub is considered. Chapter 6 then provides a discussion on how MediaHub meets the essential and desirable requirements criteria for a multimodal distributed platform hub. Finally, Chapter 7 concludes the thesis with a comparison to other work and a discussion on potential future work.

Chapter 2 Approaches to Multimodal Systems

This chapter reviews multimodal systems and the concepts and technology related to their development. First, the fusion and synchronisation of multimodal input and output data is considered. Then, three problems fundamental to the design of intelligent multimodal systems are discussed; semantic representation and storage, communication and decision-making. Tools for distributed processing are addressed and a review of existing multimodal platforms and systems is given. Intelligent multimedia agents are discussed and the pertinent problem of turn-taking in such agents considered. Multimodal corpora and annotation tools are reviewed, before a discussion on two key problems in the design of intelligent multimodal systems: dialogue act recognition and anaphora resolution. The chapter concludes with a discussion on the limitations of current AI research.

2.1. *Multimodal data fusion and synchronisation*

Multimodal data fusion refers to the process of combining information from different modalities (information chunks), "so that the dialogue system can create a comprehensive representation of the communicated goals and actions of the user" (López-Cózar Delgado & Araki 2005, p. 34). More specifically, it refers to fusion of semantics relating to various modalities. Fusion of semantics is a critical task at both the input and the output of a multimodal system. An example of semantic fusion can be found in Brøndsted et al. (2001), where semantics of the utterance, "Whose office is this?" needs to be fused with semantics of the corresponding gesture input, i.e., pointing to the intended office.

Fusion, as discussed in López-Cózar Delgado & Araki (2005, p. 34), can be performed at a number of levels including signal, contextual, micro-temporal, macro-temporal and semantic. Signal (lexical) fusion involves attaching hardware primitives to software events. Only temporal concerns, e.g., synchronisation, are considered without any regard to interpretation at a higher level. Signal fusion frequently occurs in audio-visual speech recognition, i.e., fusion of speech with lip movement signals. Semantic fusion interprets the multimodal information chunks separately and considers their meaning before combining them. Fusion at the semantic level is normally conducted in two stages: (1) the multimodal events are combined at a low level and (2) the high level meaning of the combination is extracted. As discussed in López-Cózar Delgado & Araki (2005, p. 36), semantic fusion, "is mostly applied to modalities that differ in the time scale characteristics of their features". For example, semantic fusion often occurs when combining speech and pointing, as in the IntelliMedia WorkBench application of Chameleon (Brøndsted et al. 1998, 2001). Nigay and Coutaz (1995) consider three kinds of fusion: microtemporal, macrotemporal and contextual. Microtemporal fusion combines the information chunks as they are provided in parallel. Macrotemporal fusion combines information chunks that are provided in sequence, provided in parallel but processed sequentially, or delayed due to processing limitations of the system. Information chunks may be delayed due the processor-hungry nature of the data being processed, e.g., speech or visual input will take longer to process than mouse input. Contextual fusion, according to Nigay and Coutaz (1995), combines the information chunks without giving any consideration to temporal constraints.

Decisions on the output of a multimodal system can relate to both the choice of output modality, e.g., speech output only when a car is moving, as discussed in Berton et al. (2006) and to the synchronisation of multimodal output, e.g., synchronising the movement of a pointing laser with corresponding speech output, as in the IntelliMedia WorkBench application of Chameleon (Brøndsted et al. 2001) where a statement of the form, "This is the route from Paul's office to Tom's office", needs to be synchronised with the laser output tracing the route between the two offices.

Systems capable of integrating speech and gestures have been proposed since the early eighties. Bolt (1980, 1987) introduced the 'Put-That-There' system, which enables the user to move shapes about a graphical display with speech commands and pointing gestures. Mc Kevitt (1995/96) includes a collection of computational models and systems that have been designed to address the challenge of multimodal integration. Ó Nualláin & Smith (1994) investigate the relationship between the semantics of language and vision, which led to the development of Spoken Image (Ó Nualláin et al. 1994). Spoken Image enables a user to quickly build an envisaged house or town scene, within a 3D environment, by describing the scene with natural language. Aesopworld (Okada 1996; Okada et al. 1999) aims to create an architectural foundation for intelligent agents. Aesopworld involves the creation of a computational agent that simulates various kinds of mental activities. QuickSet (Johnston et al. 1997, Johnston 1998) focuses on the integration of speech with pen-based input. Hall and Mc Kevitt (1995) observe that language and vision are combined by humans in a non-linear fashion which makes analysis of the phenomenon a very difficult task. They argue that their use of a knowledge representation that is independent of both the vision processing and natural language processing modules makes vision-language integration feasible. Pastra & Wilks (2004) highlight the need for a comprehensive language and vision integration standard and present the Vision-Language intEgration MechAnism (VLEMA) as a possible solution. Martin et al. (2001) achieve the linking of a speech/gesture segment with its visual reference object by nesting the reference object in an XML segment annotation. This work was then continued in Martin & Kipp (2002), where it was put to practice in the ANVIL annotation tool (Kipp 2001, Martin & Kipp 2002). Several mechanisms have been employed for the semantic fusion of multiple modalities, including frames, melting pots, neural networks and agents. An important prerequisite of modality integration is the means of representing the individual modalities. This is referred to as semantic representation and is discussed in the next section.

2.2. *Multimodal semantic representation*

One of the central questions in the development of intelligent multimedia or multimodal systems is what form of semantic representation should be used (Ma & Mc Kevitt 2003, Mc Kevitt 2005). This semantic representation needs to support the interpretation/generation of multimodal input/output and semantic theory. The representation can contain architectural, environmental and interactional information. Architectural information comprises the producer/consumer of the information, associated confidence and input/output devices. Environmental information contains timestamps and spatial information, whilst interactional information includes the speaker/user's state. Here, four approaches to semantic representation are considered: frames, typed feature structures, melting pots and XML.

2.2.1. Frames

A frame is a set of attributes together with associated values that represent some real world entity. Minsky (1975) first introduced frames as a method of semantically representing situations in order to facilitate decision-making and reasoning. The idea of frames is based on human memory and the psychological view that, when humans are faced with a new problem, they select an existing frame, or remembered framework, and adapt it to fit the new situation by changing appropriate details. Although frames have limited capabilities on their own, a frame-based system provides a powerful mechanism for encoding information to support reasoning and decision-making. Frames can be used to represent concepts, including real world objects, for example, "the village of Dromore". The frames for representing each concept have slots which represent the attributes of the concept. Frame-based methods of semantic representation are implemented in several multimodal platforms, including Ymir (Thórisson 1996, 1999), Chameleon (Brøndsted et al. 1998, 2001), Waxholm (Carlson & Granström 1996; Carlson 1996) and DARBS (Choy et al. 2004a, 2004b; Nolle et al. 2001). Figure 2.1 shows example frame-based semantic representations from Chameleon. The example frames in Figure 2.1 illustrate how speech and gesture input are represented. The *SPEECH-RECOGNISER* frame in Figure 2.1 has three slots: (1) *UTTERANCE:*, which contains the speech recognised by the system, (2) *INTENTION:*, which indicates the perceived intention of the user, i.e., the user is intending to give an instruction, and (3) *TIME:* which contains a timestamp for speech input. The *GESTURE* frame in Figure 2.1 also captures the intention and timestamp of the gesture input and contains the recognised coordinates of the pointing gesture in the *GESTURE:* slot. Note that, although the syntax and structure of frames will vary from system to system, the basic principles of knowledge representation will remain the same.

```
[SPEECH-RECOGNISER
UTTERANCE:(Point to Hanne's office)
INTENTION: instruction!
TIME: timestamp]

[GESTURE
GESTURE: coordinates (3, 2)
INTENTION: pointing
TIME: timestamp]
```

Figure 2.1: Example frames from Chameleon (Brøndsted et al. 1998, 2001)

2.2.2. Typed Feature Structures

Typed Feature Structures (TFSs) (Carpenter 1992) are another means of combining chunks of information across different modalities. TFSs allow partial information chunks to be specified. For example, a gesture that partially specifies the user's intention can be represented by a TFS containing some un-instantiated features. The dialogue manager can then ask the user to provide missing information to help clarify the intentions of the user. TFSs have been used in the implementation of a variety of dialogue systems, including QuickSet (Johnston et al. 1997) and in Holzapfel et al. (2002) which used multidimensional TFSs to annotate multimodal information chunks with, e.g., confidence scores and input channels.

2.2.3. Melting pots

Melting pots (Nigay & Coutaz 1995, López-Cózar Delgado & Araki 2005) are a semantic representation technique for encapsulating time-stamped multimodal information to support fusion based on the complementarity of the melting pots, time and the current context. Separate melting pots can combine information from different modalities. For example, when the user says, "What are the available flights from Chicago to this city?", and points to a map, one melting point can handle speech input stating the start location, and the other handle a pointing gesture to the destination. The two melting pots can then be integrated by a fusion module to fully represent the intention of the user. Melting pots are applied in MATIS (Multimodal Airline Travel Information System) (Nigay & Coutaz 1995), which gives information on flight schedules. The integration of two melting pots in the MATIS system is illustrated in Figure 2.2.

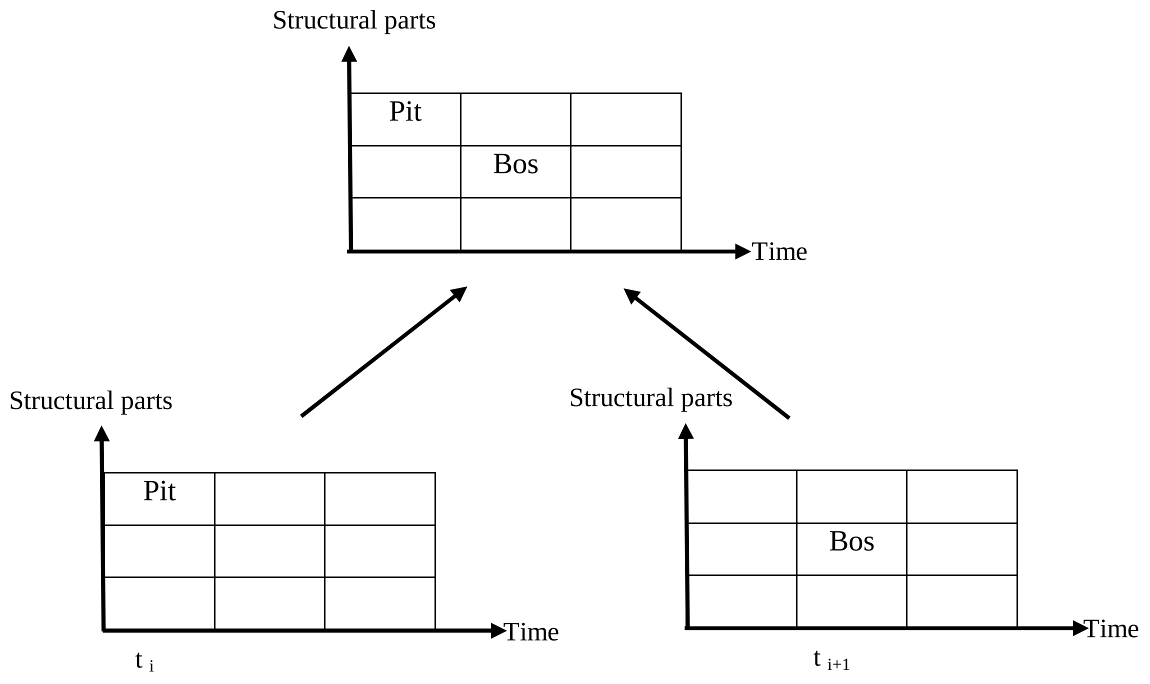


Figure 2.2: Melting Pots in MATIS (Nigay & Coutaz 1995)

The fusion depicted in Figure 2.2 occurs when the user issues a multimodal request for information on flights from Pittsburgh to Boston. The left melting pot in Figure 2.2 is generated by the speech act triggered by the utterance, "flights from Pittsburgh", whilst the right melting pot results from the user selecting Boston with the mouse.

2.2.4. XML and derivatives

The eXtensible Markup Language (XML), created by the W3C (World Wide Web Consortium) (W3C 2009), is a derivative of SGML (Standard Generalised Mark-up Language). XML was initially developed for use in electronic publishing, but is now used extensively in the exchange of data via the Web. The main purpose of XML is to provide a mechanism for the mark-up and structuring of documents. XML is different to HTML in that tags are only used within XML to delimit segments of data. Interpretation of the data is left to the application that reads it. Another advantage of XML is that it is possible to easily create new XML tags. SmartKom (Wahlster

2003, 2006; Wahlster et al. 2001; SmartKom 2009), Interact (Jokinen et al. 2002) and MIAMM (Reithinger et al. 2002) have an XML-based method of multimodal semantic representation.

Derivatives of XML also exist for multimodal semantic representation. SmartKom (Wahlster 2006) has an XML-based mark-up language, M3L (MultiModal Markup Language), for semantically representing information passed between its various components. The exchange of information within MIAMM is facilitated by a derivative of XML called MMIL (Multi-Modal Interface Language). The Synchronised Multimedia Integration Language (SMIL) (Rutledge 2001, Rutledge & Schmitz 2001, SMIL 2009a,b) is a representation language for encoding multimedia presentations for distribution over the Web. SMIL consists of XML elements and attributes describing the temporal and spatial coordination of media objects. The focus of SMIL is on the integration and synchronisation of independent media. Another XML-based mark-up language is EMMA (Extensible MultiModal Annotation markup language) (EMMA 2009). EMMA is a canonical structure for marking up a variety of modalities including speech, natural language text and gestures.

2.2.5. Other semantic representation languages

MPEG-7 (MPEG-7 2009) is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). The MPEG-7 standard provides tools for describing multimedia content. MPEG-7 aims to enable, “interoperable searching, indexing, filtering and access of audio-visual (AV) content by enabling interoperability among devices and applications that deal with AV content description” (MPEG-7 2009, p. 1). It is expected that MPEG-7 will make the Web more searchable with multimedia content as the search criteria, as opposed to the traditional method of searching for text.

The Semantic Web (Berners-Lee et al. 2001; SW 2009) aims to provide, “a common framework that enables data to be shared and reused across application, enterprise and community boundaries” (SW 2009, p. 1). Based on the Resource Description Framework (RDF) (Klein 2001, 2002, Nejdl et al. 2000, RDF Schema 2009), it is a joint effort led by the W3C (World Wide Web Consortium) with input from other research and industrial partners. According to Berners-Lee et al. (2001, p. 36), “the Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation”. Research in this area is motivated by the need to develop a knowledge representation framework that can manage the mass of unstructured data on the current Web. An important goal of the Semantic Web is to put information on the Web that can be processed by machines in addition to humans. Semantic Web technologies such as RDF (Resource Description Framework) (Klein

2001, 2002, Nejdl et al. 2000, RDF Schema 2009), OIL (Ontology Inference Layer) (Fensel et al. 2001) and OWL (Ontology Web Language) (OWL 2009) enable rich information networks to be rapidly created and can assist in the generation of hypotheses across large sets of data. RDF is a general-purpose language for the representation of information on the Web. RDF Schema, an extension of RDF, provides mechanisms for describing groups of resources and relationships between them. RDF's class and property system is comparable to that of object-oriented programming languages such as Java. However, RDF, rather than defining a class in terms of the properties its instances may have, describes properties in terms of the classes of resource to which they may be applied. OIL (Ontology Inference Layer) (Fensel et al. 2001) is an ontology architecture that is being developed for use with the Semantic Web. An ontology, as defined by Gruber (1993, p. 200), is, "an explicit specification of a conceptualization". OIL represents a standard for specifying and exchanging ontologies, and aims to provide a general-purpose semantic mark-up language for the Semantic Web. OWL enables applications to process the content of information rather than simply presenting it to humans. OWL facilitates greater machine interpretability of Web content than that provided by XML, RDF and RDF Schema. This is achieved by providing additional vocabulary together with a formal semantics. OWL, based on DAML+OIL (DAML 2009, DAML+OIL 2009, Mc Guinness et al. 2002), has three sublanguages: OWL Lite, OWL DL, and OWL Full. DAML + OIL evolved from the merging of DAML + ONT, an earlier ontology language, and OIL. The goal of DAML + OIL is to, "support the transformation of the Web from being a forum for information presentation to a resource for interoperability, understanding, and reasoning" (Mc Guinness et al. 2002, p.1). A further advancement in the DAML project led to the development of DAML-S (DAML Services) (DAML-S 2009), which provides a set of mark-up language constructs to describe Web Services in a format understandable to computers.

SOAP (Simple Object Access Protocol) (Chester 2001) enables software applications developed using different programming languages and running on different platforms to interoperate. SOAP uses a combination of HTML and XML to send and receive messages in a distributed environment. HTML facilitates communication between modules, while the actual network conversation is encoded in XML. SOAP is a network protocol for calling components, methods, objects and services on remote servers. This is enabled by representing parameters, return values and errors in an XML document known as a SOAP envelope. A standardised XML vocabulary called WSDL (Web Services Definition Language) facilitates description of the methods, parameters and return values of a SOAP Web Service. The SOAP model is language/platform independent, secure and scalable. It is a loosely coupled protocol that enables information exchange in a decentralised, distributed environment. The NKRL (Narrative Knowledge Representation Language) (Zarri 1997, 2002) language can represent the semantic content of complex multimedia information in a standardised way. NKRL is a language for representing the meaning of natural language narrative documents such as, for example, news stories and corporate documents.

2.3. *Multimodal semantic storage*

In addition to semantic representation, another important factor is how semantics is stored within a multimodal system. Semantic storage is important for maintaining dialogue history in a multimodal system, which becomes increasingly more complex as dialogue systems become more intelligent. The storage of semantics is important for deictic reference ambiguity resolution in a dialogue where a user may ask, "How can I get from his office to that office [→][1]?". It is also important for communication between modules of multimodal systems.

There are three key approaches to semantic storage: blackboard based, non-blackboard based and multi-blackboard based. Similar to the traditional classroom-based blackboard, a blackboard within a multimodal system is a repository of shared information containing semantic representations of the reasoning performed, other information important to the reasoning process and conclusions reached by the system during the resolution of a problem (e.g. a dialogue). This information is maintained over time and can be accessed later to support decision-making in a multimodal dialogue. Blackboards (Thórisson et al. 2005) help simplify the construction of AI systems with large numbers of interacting modules. A key advantage of a blackboard is modularity. That is, a problem space can be divided into a number of problem areas with each being addressed by an 'expert' which works on the problem and communicates with other experts via the blackboard. The principle of modularity makes blackboards popular in the design of distributed and multimodal systems. Multimodal distributed platforms that deploy a blackboard model of semantic storage include Chameleon (Brøndsted et al. 1998, 2001), Psyclone (Thórisson et al. 2005) and DARBS (Nolle et al. 2001).

Non-blackboard based systems typically store semantics in individual modules and communicate this semantics directly between modules through message-passing. Waxholm (Carlson & Granström 1996; Carlson 1996), Aesopworld (Okada 1996, Okada et al. 1999) and

[1] [→] indicates a deictic gesture.

InterACT (Waibel et al. 1996) all implement a non-blackboard based model of semantic storage. Some large, complex platforms and systems need to implement multiple blackboards. Multi-blackboard based approaches are deployed in Ymir (Thórisson 1996, 1999) and SmartKom (Wahlster 2006). Psyclone (Thórisson et al. 2005) enables the creation of single- and multi-blackboard based systems. Psyclone actually implements a special type of blackboard referred to as a *whiteboard*. A whiteboard can perform all the functions of a blackboard and has the same primary purpose, i.e., the storage of information relating to a problem in a shared space to support decision-making in the multimodal system. However, whiteboards in Psyclone significantly extend the blackboard model in a number of ways, including independence of programming language, both query and publish-subscribe mechanisms, and implementation of an explicit temporal model.

2.4. *Communication*

Communication is an important consideration in the development of any system. It is particularly important in the design of multimodal systems, where the system will process input and output from a variety of sources across multiple modalities. A common approach to communication within large multimodal systems is to implement a blackboard, or shared space, and have all modules in the system communicate by exchanging semantic representations via the blackboard (Brøndsted et al. 1998, 2001). Alternatively, communication may be achieved by exchanging semantic representations directly between the modules. It is also possible to both implement a blackboard and allow direct communication between the modules themselves, as is the case in Chameleon (Brøndsted et al. 1998, 2001).

A number of communication protocols exist, a frequently used protocol being TCP/IP. This is a suite of protocols used by Internet browsers and servers to connect to the Internet. TCP/IP takes its name from the two key protocols contained in the suite: Transmission Control Protocol (TCP) and Internet Protocol (IP). In addition to the Internet, TCP/IP also enables communication within multimodal and unimodal systems. MATCH (Multimodal Access To City Maps) (Johnston et al. 2002) uses TCP/IP for communication between the user interface and its Java-based facilitator MCUBE. TCP/IP is also used in JATLite (Kristensen 2001, Jeon et al. 2000), which encompasses a collection of Java packages for constructing multi-agent systems and DARBS (Distributed Algorithmic and Rule-Based System) (Choy et al. 2004a, 2004b, Nolle et al. 2001). The OpenAIR specification (Mindmakers 2009; Thórisson et al. 2005), implemented within the Psyclone platform (Thórisson et al. 2005), is a routing and communication protocol.

Based on the publish-subscribe architecture paradigm, and built upon standard TCP/IP and XML, OpenAIR aids the creation of large distributed systems.

2.5. *Multimodal decision-making*

In order to solve complex real-world problems, a system needs to combine knowledge and techniques from various sources. Such problem-solving has required systems to be developed that mimic the reasoning and decision-making capabilities of a human being. A system that is capable of tackling complex problems in a human-like manner is said to be an intelligent system and the intelligence that the system exhibits is termed Artificial Intelligence (AI) (Hopgood 2003). Many definitions of AI exist in standard text books, some more complex than others. A simple definition may be found in Hopgood (2003, p. 1), which states that AI, "is the science of mimicking human mental faculties in a computer". This definition, although perfectly correct, leaves the notion of intelligence somewhat vague. It is important to appreciate the incredible complexity of the human mind and the various levels of human intelligence that need to be replicated using intelligent systems. Figure 2.3 provides some clarification by illustrating the spectrum of intelligence based on the level of understanding involved.

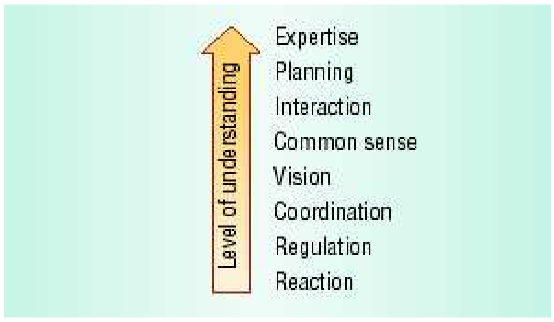


Figure 2.3: Spectrum of intelligent behaviour (Hopgood, 2003)

Although there have been considerable advancements made at the top and bottom end of the spectrum of intelligence, replicating human behaviour in the middle of the spectrum has proven to be the greatest challenge. For example, a significant gap still exists in the "common sense" region. It remains difficult to build systems that are capable of making sensible decisions under uncertainty.

According to Hopgood (2003), AI technology for modelling intelligent behaviour can be broadly categorised into the following two areas:

- Explicit modelling – with words and symbols
- Implicit modelling– with numerical techniques

Explicit modelling includes techniques such as rule-based and case-based reasoning. With explicit modelling, words and symbols create explicit rules to model the problem. An example is, 'if the temperature is hot, then turn the fan on'. This technique has the disadvantage that it can only deal with explicit situations and cannot deal with unfamiliar situations. Implicit modelling uses numerical techniques in an attempt to overcome this problem. Numerical techniques such as Neural Networks, Genetic Algorithms, Fuzzy Logic and Bayesian Networks enable the computer to create its own model based on observations and past experience. For example, Neural Networks can learn rules from a set of example data and apply these rules in previously un-encountered situations.

2.5.1. Uncertainty

Real world decisions are seldom taken with 100% certainty that they are correct. Humans deal with uncertainty all the time, often during the course of a dialogue. Background noise, for example, can increase the uncertainty involved in speech recognition. In a noisy environment there may be increased uncertainty relating to the recognition of speech. Here, people can use other non-verbal cues (e.g. lip movement, gestures) to compensate for uncertainty in recognising speech and increase their overall certainty about the message being communicated. Of course, even without noise, uncertainty frequently arises in everyday conversations. As an example consider the following dialogue segment, as discussed in Mc Tear (2004, p. 47):

1 A: John caught up with Bill outside the pub.
2 B: Did he give him the tickets?

Here there is uncertainty as to who 'he' and 'him' refer to. In this example, the uncertainty can be resolved if it is known who has the tickets. Whilst uncertainty is something that decision-making mechanisms try to resolve it is important, in the first instance, that uncertainty relating to certain inputs and beliefs are adequately recognised within the multimodal system.

It should also be noted that a dialogue which contains no uncertainty in the real world can have uncertainty introduced when interpreted by a multimodal system. Also, the choice of the wrong decision-making strategy for a particular situation can lead to the system being both 100% certain and 100% wrong. For example, consider the following conversation discussed in Mc Tear (2004, p. 48):

1 A: Click on the "install" icon to install the program.
2 B: OK.
3 B: By the way, did you hear about Bill?

4 A: No, what’s up?

5 B: He took his car to be fixed and they’ve found all sorts of problems.

6 A: Poor Bill. He’s always in trouble.

7. A: OK. Is it ready yet?

It will be obvious to the reader that the referent of “it” in turn 7 is not Bill’s car, but the program that *B* is trying to install. However, if the system were to use the last recently mentioned pronoun that matches syntactically it would believe *A* is referring to Bill’s car. This is just one unimodal example of how an ill thought-out decision strategy could lead to uncertainty or incorrectness. The presence of multiple modalities can further complicate the decision-making process unless care is taken to design a decision-making strategy that appropriately weights the various inputs and ensures that the multiple modalities are harnessed to reduce uncertainty and ambiguity.

Another potential source of uncertainty in a dialogue is sarcasm. Consider the dialogue below:

1 A: Any holidays planned this year?

2 B: Off for a week on Monday. Going to stay local, planning to tour Donegal.

3 B: Have you heard the weather forecast for the next few days?

4 A: I heard there will be a lot rain on Monday and Tuesday.

5B: Oh Great! (☺).

Note that the ☺ symbol indicates the speaker is smiling whilst uttering turn 5. Here, participant *B* is not happy that there is a lot of rain forecasted for Monday and Tuesday. The utterance, “Oh Great!’, and the accompanying smile of turn 5 are examples of sarcasm. Again, this will be obvious to the reader, but turn 5 could easily be misinterpreted by a computer. The effective use of dialogue history, labelling turn 4 as negative information for participant *B,* and an understanding that a positive comment and expression can sometimes constitute sarcasm could help avoid uncertainty in this dialogue.

2.5.2. Fuzzy Logic

Fuzzy Logic (Zadeh 1965; Passino & Yurkovich 1997) is a problem solving methodology that can be used to arrive at definite conclusions based on vague, imprecise or missing data. This enables Fuzzy Logic to mimic the approximate reasoning capability of human beings. As an example of such reasoning, consider what we do in the shower when the temperature is too hot: we can quickly fix the problem by adjusting the temperature control knob without knowing the exact temperature of the water. Note that the water will be perceived to be too hot over a wide range of temperature values and not only at a fixed temperature. Thus our action in adjusting the control knob is based on perception, not just on a measured value. Fuzzy Logic attempts to replicate this form of human reasoning.

Fuzzy Logic uses linguistic or fuzzy variables, typically nouns such as, “temperature”, “error” or “voltage”. All of these linguistic variables have linguistic values assigned to them. Examples of linguistic values are, “very hot”, “zero”, “positive large”, “negative small”, “ok”, “cold” and “very cold”. This will be further clarified by a simple example. Consider the problem of balancing an inverted pendulum as discussed in Passino & Yurkovich (1997). A model diagram of the problem is shown in Figure 2.4.

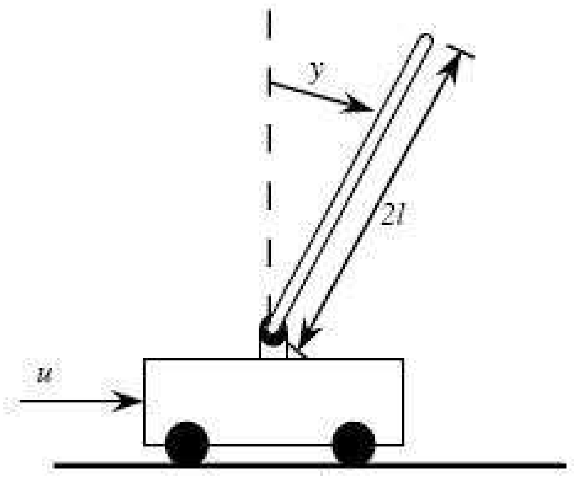


Figure 2.4: Inverted Pendulum (Passino & Yurkovich 1997)

Note that ‘u’ in Figure 2.4 represents the force that must be applied to the moving cart in order to balance the pendulum, and ‘y’ represents the displacement of the pendulum from the vertical (balanced) position. Suppose that an expert on the system says that he/she will use *e(t)*, i.e., error, and *d/dt e(t)*, i.e., change in error, as the variables on which to base decisions. The error is mathematically expressed as follows:

$$e(t) = r(t) - y(t) \qquad (2.1)$$

where *r(t)* is the reference point, indicated by the dashed line in Figure 2.4, and *y(t)* is the displacement from this position. The following linguistic variables will be used in the fuzzy system:

“error” describes *e(t)*

“change-in-error” describes *d/dt e(t)*

“*force*” describes *u(t)*

The linguistic variables will assume linguistic values over time. That is, *“error”*, *“change-in-error”* and *“force”* will take on the following linguistic values:

"neglarge"

"negsmall"

"zero"

"possmall"

"poslarge"

Note that, *"neglarge"*, is an abbreviation for, *"negative large"*, and, *"poslarge"*, is an abbreviation for *"positive large"*. We can now define linguistic rules that will capture the expert's knowledge about how to control the pendulum. A few of these rules are:

If error is negsmall and change-in-error is zero Then force is possmall

If error is zero and change-in-error is possmall Then force is negsmall

If error is poslarge and change-in-error is negsmall Then force is negsmall

We can now use membership functions to plot the meaning of the linguistic values. Consider Figure 2.5. This is a plot of a function μ versus an error $e(t)$. The function μ quantifies the certainty that $e(t)$ can be classified linguistically as *"possmall"*. For example, if $e(t) = \pi/4$ then $\mu(\pi/4) = 1.0$, indicating that we are absolutely certain that $e(t) = \pi/4$ is what we mean by *"possmall"*.

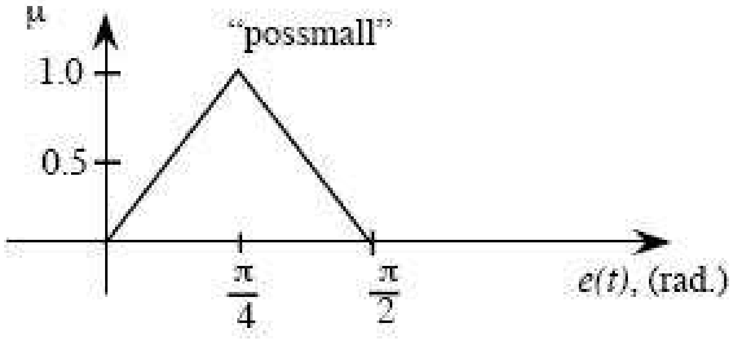


Figure 2.5: Membership function for *"possmall"* (Passino & Yurkovich 1997)

To better understand the concept of the membership function, we will draw the membership functions for the inverted pendulum using the following default initial conditions:

$y(0) = 0.1$ *radians*, $r(0) = 0$, *and* $u = 0$ (2.2)

Therefore, using (2.1):

$e(t) = r(0) - y(0) = 0 - 0.1 = -0.1$ (2.3)

$d/dt\ e(t) = dy/dt = 0$ (2.4)

The membership functions for these input conditions are shown in Figure 2.6. For $e(t) = -0.1$ and $d/dt\ e(t) = 0$, we can see from Figure 2.6 that there are two active rules:

If error is zero and change-in-error is zero then force is zero

If error is negsmall and change-in-error is zero then force is possmall

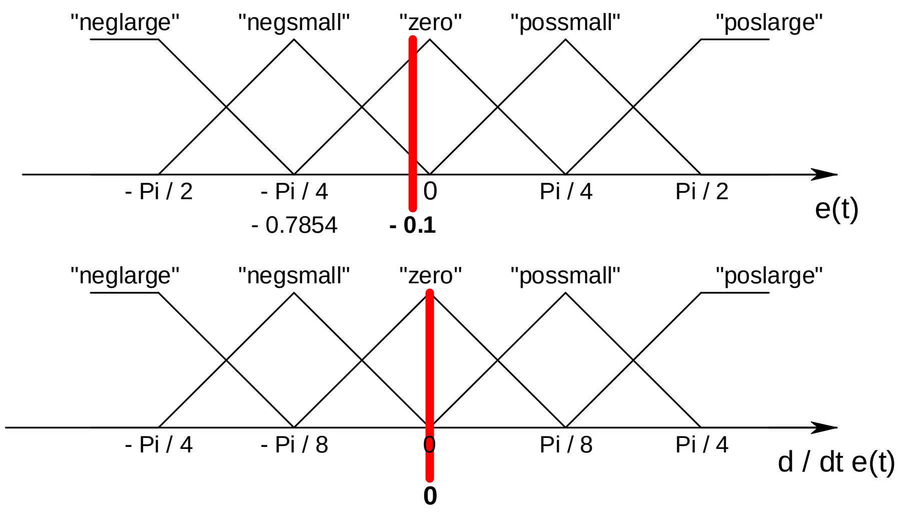


Figure 2.6: Membership functions for "error" and "change in error"

Having defined which rules are on, our next step (the inference step) is to determine which conclusions should be reached in deciding what force should be applied to the cart in order to keep the inverted pendulum balanced. The final step is to convert the decisions reached by each of the rules into actions. This step is known as defuzzification and the result is a crisp (or non-fuzzy) output force that should be applied to the cart in order to keep the inverted pendulum balanced. Fuzzy logic is one method of AI that could be applied in a multimodal system where decisions need to be made based on noisy or ambiguous multimodal data. The TeleMorph Fuzzy Inference System (Solon et al. 2007) implements fuzzy logic for decision-making in TeleTuras, a mobile intelligent multimedia tourist information system.

2.5.3. Genetic Algorithms

Genetic Algorithms (GAs) (Davis 1991; Goldberg 1989; Holland 1992) are another paradigm within Artificial Intelligence that can facilitate decision-making within a multimodal system. GAs were inspired by Darwin's Theory of Evolution which states how all living things evolve, adapting themselves to constantly changing environments in order to survive. Darwin observes that the weaker members of a species have a tendency to die away, leaving the stronger to mate and create offspring. This theory is illustrated in Figure 2.7.

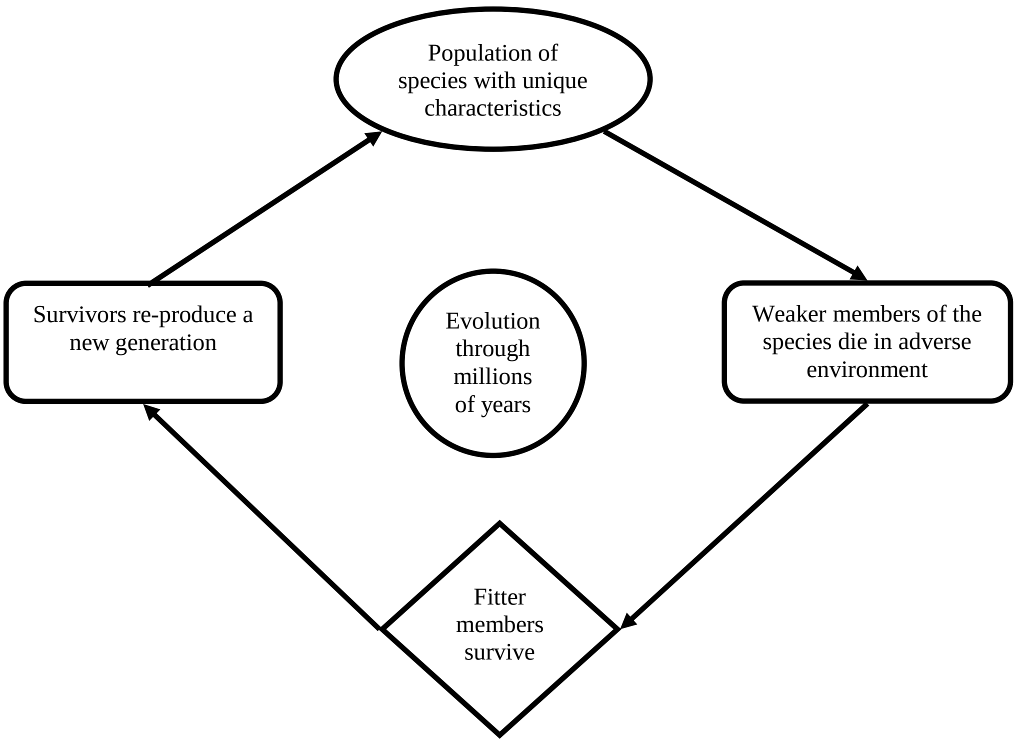

Population of species with unique characteristics
Survivors re-produce a new generation
Evolution through millions of years
Weaker members of the species die in adverse environment
Fitter members survive

Figure 2.7: Darwin’s Theory of Evolution

Genetic Algorithms are essentially exploratory search and optimisation methods, where each potential solution to the problem is represented by an individual in the population. Each individual within the GA is represented by a string. Every iteration of the GA creates a new population from the old by interbreeding the fittest strings to create new ones which may be closer to the optimal solution to the problem in question. So in each generation, new strings are created from the segments of previous strings. Additionally, new random data is occasionally added in order to keep the population from stagnating. GAs are parallel search procedures, i.e., they can be implemented on parallel processing machines which can greatly speed up their operation. The main elements of GAs are:

- Chromosomes - Each point in a solution space is encoded into a bit string called a chromosome.
- Encoding schemes – An encoding scheme transforms a point in a solution space into a bit string representation.

Although several encoding schemes exist, the most commonly used is the binary coding scheme. In the binary coding scheme, each decision variable in the solution set is encoded in a binary string and concatenated to form a chromosome. For example, a point (3, 12, 7) in a 3D parameter space could be represented by the following chromosome:

C = { 0011 1100 0111}

Three basic operators are found in every GA. These are selection (reproduction), crossover and mutation. The selection operator allows an individual string to be copied, based on its fitness value, and possibly included in the next generation. A fitness function is used to calculate the fitness value of a string. Crossover in GA is analogous to crossover in biological terms where chromosomes from the parents are blended to produce new chromosomes for the offspring. The mutation operator introduces new genetic material into an existing individual, thus increasing the genetic diversity of the population.

2.5.4. Neural Networks

Neural Networks (Haykin 1999) are information processing paradigms that have been inspired by the manner in which biological nervous systems, e.g., the human brain, process information. A neural network is made up of large numbers of interconnected processing elements, or neurons, working in parallel to solve a problem. Neural networks, "learn by example". That is, they are trained using input and output data sets to adjust the synaptic weight connections between the neurons. An example of a neural network is shown in Figure 2.8. Three important properties of neural networks are:

- Parallelism
- Learning
- Generalisation

Parallelism enables large volumes of data to be processed in parallel. Due to the distributed nature of the neurons within the layers of a neural network, the processing capabilities of the network are distributed across all of its neurons and synapses. This means that the corruption or damage of one layer may not significantly degrade the output. Learning can be applied to solving a problem when a model of the problem is not known. The network is trained using known input/output data sets of the problem. Generalisation enables a neural network to process data on which it has never been trained. The network learns rules from a set of examples, thus it is not necessary to explicitly know and program the necessary actions. Neural Networks are another technique within Artificial Intelligence for potential decision-making within multimodal systems.

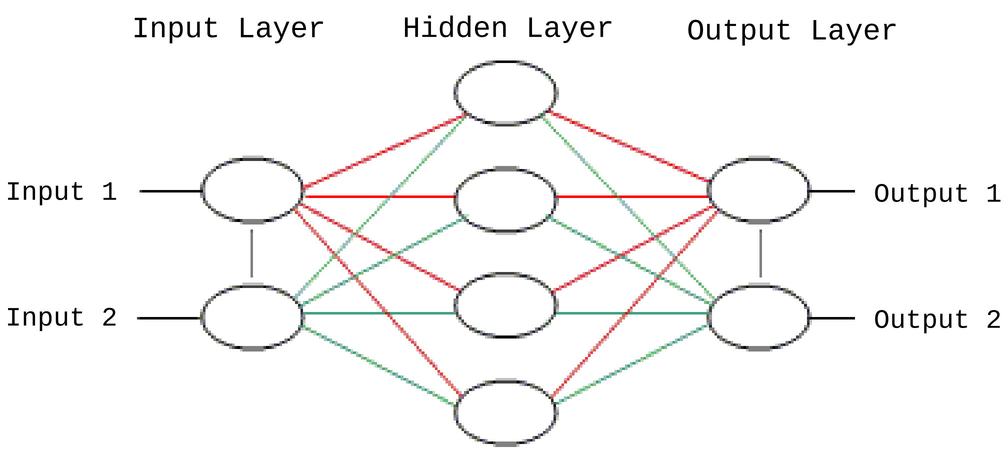


Figure 2.8: Example of a typical neural network structure

2.5.5. Bayesian networks

Bayesian networks (Pearl 1988; Jensen 2000; Kadie et al. 2001; Jensen & Nielsen 2007; Hugin 2009; Pourret et al. 2008) are also known as Bayes nets, Causal Probabilistic Networks (CPNs), Bayesian Belief Networks (BBNs) or belief networks. A Bayesian network consists of a collection of nodes with directed edges between the nodes. Each of the nodes represents a random variable, with each edge representing a cause-effect relationship within the domain, i.e., causal impact from one node to another. An edge connecting two nodes A and B indicates that a direct influence exists between the state of A and the state of B. All edges in the graph are directed and directed cycles are not permitted, i.e., it is a Directed Acyclic Graph. As an example, consider Figure 2.9, where the directed edges from 'Diet' and 'Exercise' have an impact on 'Weight Loss'. It can be seen in Figure 2.9 that the nodes are represented by ovals and the directed edges are illustrated by arrows. In Figure 2.9, there is a causal dependency from 'Diet' to 'Weight Loss' and from 'Exercise' to 'Weight Loss'.

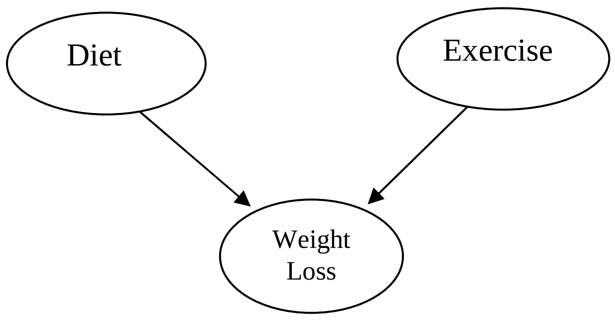


Figure 2.9: Example of a simple Bayesian network

Each node in a Bayesian network contains the states of the random variable it represents together with a Conditional Probability Table (CPT), or a Conditional Probability Function (CPF). The CPT contains the probabilities of the node being in a particular state given the states of its parents. The effects in a Bayesian network, represented by the directed edges, are not completely deterministic (e.g. disease -> symptom) and the strengths of these effects are modelled as probabilities. The following example gives the conditional probability of having a certain medical condition given the variable temp (where *P(…)* represents a probability function):

1) If tonsillitis then P(temp>37.9) = 0.75
2) If whooping cough then P(temp>37.9) = 0.65

Since 1) and 2) could be mistakenly read as rules, the following notation was developed, where '|' represents a directed edge in the Bayesian network from the latter node (whooping cough) to the first node (temp>37.9):

P(temp>37.9 | whooping cough) = 0.65

If 1) and 2) are read as 'If otherwise healthy and...then...', it is also necessary to specify how the two causes combine. That is, one needs to know the probability of having a fever if both, or none, of the symptoms are present. Hence, it is necessary to specify the conditional probabilities:

P(temp>37.9 | whooping cough, tonsillitis)

where 'whooping cough' and 'tonsillitis' each have the states 'yes' and 'no'. Thus it is necessary to define the strength of all combinations of states for all of the possible causes. Inference, or model evaluation, computes the conditional probability for some variables given information, or evidence, on other variables. This is easiest when all evidence relates to variables that are ancestors, or parent nodes, of the variable(s) of interest. However, when evidence relates to a descendant of the variable(s) of interest, one has to perform inference against the direction of the edges. In order to do this, Bayes' Theorem is employed:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \qquad (2.5)$$

In other words, the probability of some event A occurring, given that event B has occurred, is equal to the probability of event B occurring, given that event A has occurred, multiplied by the probability of event A occurring and divided by the probability of event B occurring.

2.6. *Distributed processing*

A distributed system allows the various components of the system to be distributed over multiple computers. The processing power of a distributed system may thus be spread over several computers and the processing speed of the system can be greatly increased. Each computer within a distributed system can be configured to operate on a specific task. The collection of computers that form a distributed system can appear to the user as a single system. In addition to the ability to run applications faster, distributed systems offer several other advantages. These include:

- Price – desktop computers provide cheap but powerful processing.
- Data sharing – data can be shared amongst all computers in a distributed system. For example, a central computer might keep the records of all the customers of a bank. Staff at all branches of the bank can access these records as if they were available locally on their own computers.
- Reliability – if one computer in a distributed system goes down, the others can still perform.
- Incremental growth – machines can be upgraded, replaced and added incrementally.
- Scalability – if more processing is needed, new computers can be added to the distributed system.

A disadvantage associated with the use of distributed systems is that they are often critically dependent on the network. If the network is down or unreliable, serious problems can arise. Another disadvantage concerns security, as data sharing increases the possibility of security violations.

2.7. *Tools for distributed processing*

Recent advances in the area of distributed systems have seen the development of several software tools for distributed processing. These tools are utilised in the creation of a range of distributed platforms. Numerous software tools for distributed computing are available and an overview of such tools will now be given.

2.7.1. PVM

PVM (Parallel Virtual Machine) (Sunderam 1990, Fink et al. 1995) is a programming environment that provides a unified framework where large parallel systems can be developed. It enables development of large concurrent or parallel applications consisting of relatively independent interacting components. An objective of PVM is to enable existing software to be incorporated into a larger system, with little or no modifications. A demon task is present on each machine of the distributed system. This allows communication to be monitored by a debugger, thus simplifying the analysis of heterogeneous interaction between the modules. A disadvantage with the PVM system is that it does not have a centralised instance of a service that can monitor the system configuration. This limitation causes additional overhead during reconfiguration.

2.7.2. ICE

ICE (INTARC Communication Environment) (Amtrup 1995) is a communication mechanism for AI projects developed at the University of Hamburg. ICE is based on the Parallel Virtual Machine (PVM), with an additional layer added to interface with several programming languages. Support for visualisation is provided by the use of the Tcl/Tk scripting language, which has graphical capabilities. ICE supports the following programming languages:

- C / C++
- Allegro Common Lisp
- C LISP
- LUCID Common Lisp
- Sicstus Prolog
- Quintus Prolog
- Tcl/Tk

The overall structure of a system constructed using ICE is shown in Figure 2.10. A system developed using ICE can be composed of several components written in different programming languages. Each component is given a unique name and can communicate with all other components within the system.

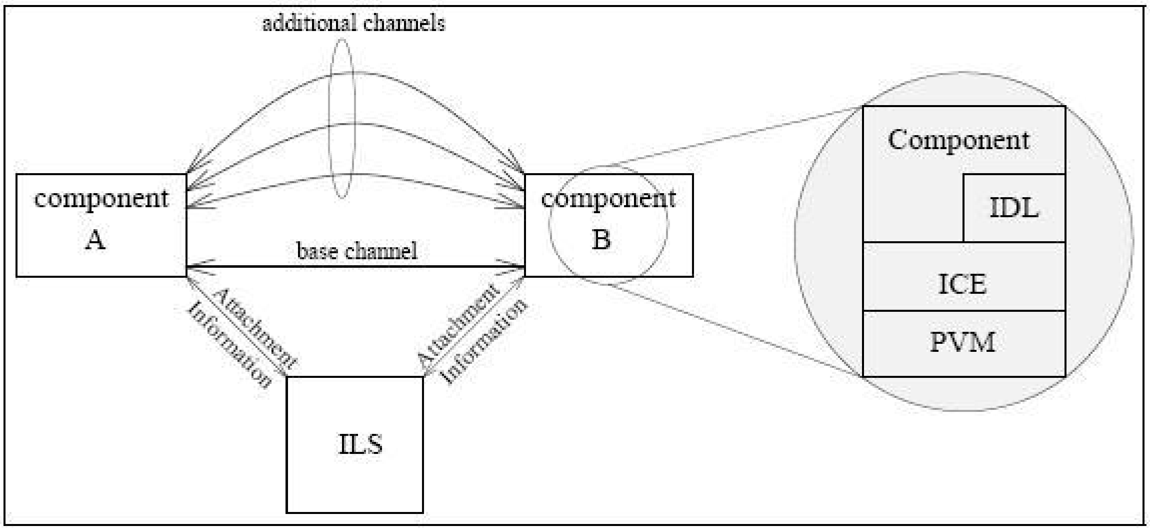


Figure 2.10: Typical structure of an ICE system (Amtrup 1995)

As shown in Figure 2.10, components communicate with each other via channels. ICE provides a base channel between components that uses eXternal Data Representation (XDR) to encode data as hardware-independent.

2.7.3. DACS

DACS (Distributed Applications Communication System) (Fink et al. 1995, 1996) is a powerful tool for system integration that provides a multitude of useful features for developing and maintaining distributed systems. Communication within DACS is based on asynchronous message passing, with additional extensions to deal with dynamic system reconfiguration at run-time. Other more advanced features include both synchronous and asynchronous remote procedure calls and demand streams that can handle data in continuous data streams.

All messages passed within DACS are encoded in a Network Data Representation, which enables the inspection of data at any stage and the development of generic tools capable of processing different kinds of information. Similar to PVM, DACS uses a communication demon that runs on each participating machine. DACS differs from PVM in that the communication demon enables multiple users to access the system simultaneously and virtual machines dedicated to individual users are not provided. The purpose of the DACS demon is to route all internal traffic and establish connections to other demons located on remote machines. A central name server keeps track of all registered demons and modules. This avoids the overhead that would result if changes in the system configuration were broadcasted. Each participating module must register with the system using a unique name, which is passed to the name server and enables other modules to address it. DACS constitutes a flexible communication tool that can be utilised in the creation of distributed systems. DACS could be used in different applications that necessitate the integration of existing heterogeneous software systems.

2.7.4. Open Agent Architecture (OAA)

The Open Agent Architecture (OAA) (Cheyer et al. 1998, OAA 2009) is a general-purpose infrastructure for creating systems that contain multiple software agents. OAA enables such agents to be written in different programming languages and running on different platforms. According to its developers, “OAA enables a truly cooperative computing style wherein members of an agent community work together to perform computation, retrieve information, and serve user interaction tasks” (OAA 2009, p.1). OAA distinguishes itself from other methods of distributed computing, such as the Blackboard approach, in that it enables both human users and software agents to express what they want done without specifying who should perform the task or how it should be done. For example, a user may issue the request "notify me immediately when a message for me arrives in relation to security" or "print this page on the nearest printer". Figure 2.11 shows the agent interaction within an OAA-based system.

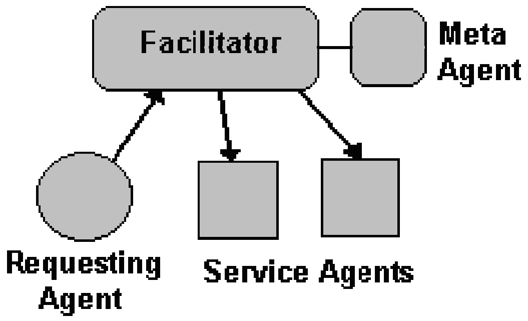


Figure 2.11: Agent interaction in OAA (OAA 2009)

As shown in Figure 2.11, the requesting agent issues a request to the Facilitator, which matches the request with an agent, or agents, which provides that service. All agents interact using the Interagent Communication Language (ICL), which can express high-level, complex tasks and natural language expressions. A major advantage of OAA is the ability to add new agents on the fly.

2.7.5. JavaSpaces

JavaSpaces (Freeman 2009), developed by Sun Microsystems, enable developers to quickly create collaborative and distributed applications. JavaSpaces represent a distributed computing model where, in contrast to conventional network tools, processes do not communicate directly. Instead, processes exchange objects through a space or shared memory. A process can write an object to a space, take an object from a space, or read an object in a space, i.e., make a copy of an object. These operations are shown in Figure 2.12.

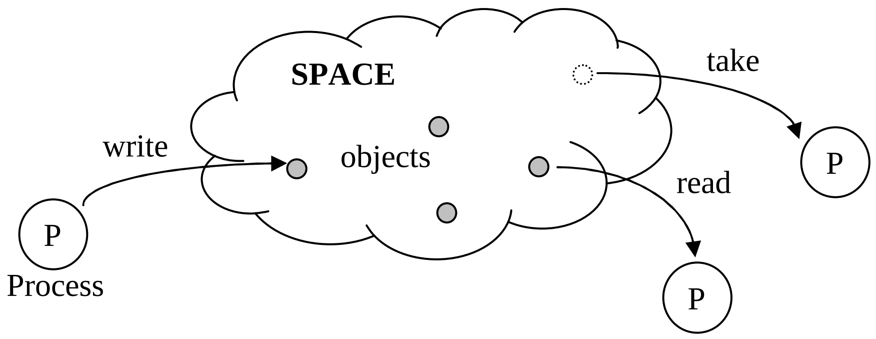


Figure 2.12: Read, write and take operations within JavaSpaces

To read or take an object from the space, processes use simple matching, based on the value of fields, to find the object that they require. Because processes can communicate through spaces, rather than communicating directly, more flexible and scalable distributed applications can be developed.

2.7.6. CORBA

The CORBA (Common Object Request Broker Architecture) (CORBA 2009, Vinoski 1993) specification was released by the Object Management Group (OMG) in 1991. CORBA is a standard architecture for the creation of distributed object systems, enabling distributed heterogeneous objects to work together. CORBA facilitates the requesting of the services of a distributed object. The services of an object can be accessed through the object's interface, defined using the Interface Definition Language (IDL) and which has a syntax similar to C++. A major component of CORBA is the Object Request Broker (ORB), which delivers requests to objects and returns results back to the client. The role of the ORB is shown in Figure 2.13.

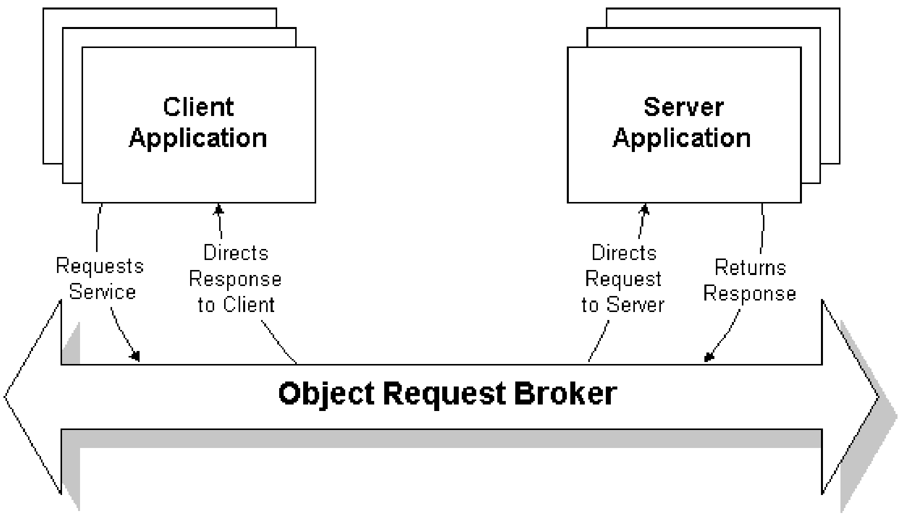


Figure 2.13: A typical object request (CORBA 2009)

Note that the client object holds a reference to the distributed object. The operation of the ORB is completely transparent to the client. That is, the client doesn't need to know where the objects are, how they communicate, how they are implemented, stored or executed. The client and the CORBA object uses exactly the same request mechanism irrespective of where the object is located. In situations where the requesting client is written in a different programming language from that of the CORBA object, the ORB will translate between the two programming languages. One of the key objectives of CORBA is to enable client and object implementations to be portable. To meet this requirement two application programmer's interfaces (APIs) are defined, one for implementing the CORBA object and another for the clients of a distributed object.

2.7.7. JATLite

JATLite (Kristensen 2001, Jeon et al. 2000) provides a set of Java packages that enable multi-agent systems to be developed using Java. JATLite provides a Java agent platform that uses the KQML Agent Communication Language (ACL) for inter-agent communication. An important component within the JATLite platform is the Agent Message Router (AMR). The AMR is responsible for registering agents with a name and password. JATLite facilitates modular construction of systems and consists of the following layers:

- Abstract layer, which provides a collection of abstract classes necessary for JATLite implementation.
- Base layer, which provides communication based on TCP/IP and the abstract layer.
- KQML (Knowledge Query and Manipulation Language) layer, which provides for storage and parsing of KQML messages.
- Router layer, which provides name registration and routing and queuing of messages via the AMR.

These four layers enable flexibility in the infrastructure of systems developed using JATLite, allowing developers to select the most appropriate layer for their system.

2.7.8. .NET

.NET (MS .NET 2009) is the Microsoft Web services strategy that enables applications to share data across different operating systems and hardware platforms. The Web services provide a universal data format that enables applications and computers to communicate with each another. Based on XML, the Web services enable communication across platforms and operating systems, irrespective of what programming language is used to write the applications. The .NET framework can either be installed as a client or a server. Both the client and the server configurations of the .NET framework offer their own advantages and the developer must consider these carefully before making a decision on which configuration to use. The .NET framework consists primarily of the following two components:

- The Common Language Runtime (CLR)
- The .NET Framework Class Library

The CLR is a system agent that runs and manages .NET code at run-time, managing basic services such as memory management and error control. The .NET Framework Class Library is a collection of object-oriented types for developing applications, services and components. .NET enables the components of a system to be shared over the Internet. The framework makes use of Web Services that can be utilised by both Windows-based applications and those running on other platforms – provided these applications use Internet standards such as TCP/IP, HTTP, XML and SOAP. The architecture of the .NET framework is shown in Figure 2.14. As shown in Figure 2.14, .NET enables three different kinds of applications to be developed: applications running managed code under the CLR, applications running unmanaged machine code and Web applications, and services running unmanaged code under ASP.NET. Using .NET, both managed and unmanaged applications can be written to co-exist on the same computer.

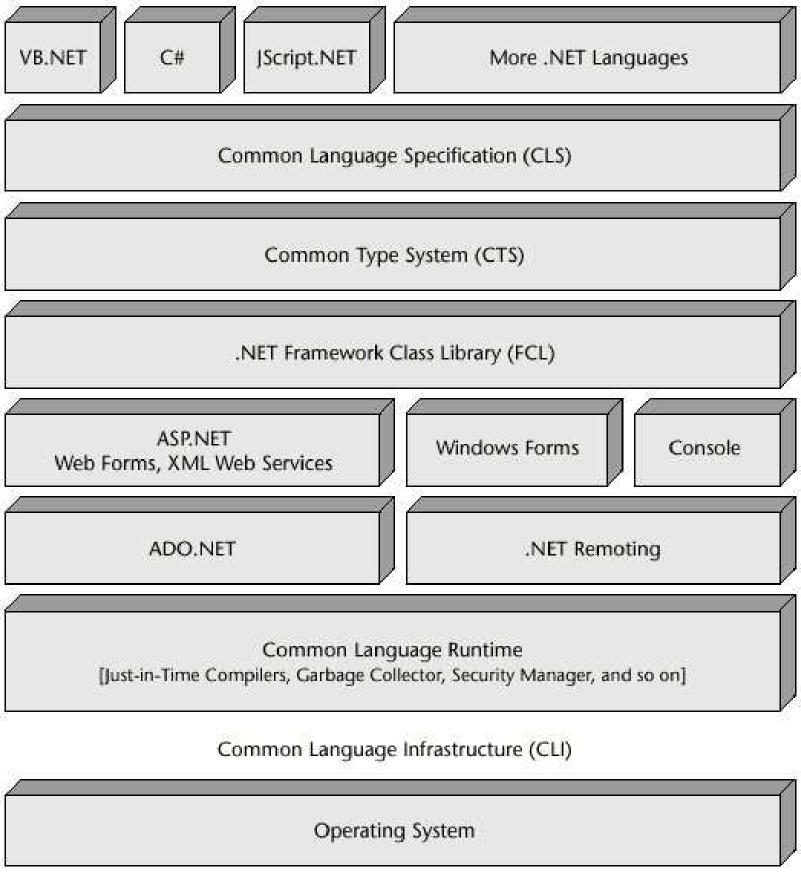

VB.NET
C#
JScript.NET
More .NET Languages
Common Language Specification (CLS)
Common Type System (CTS)
.NET Framework Class Library (FCL)
ASP.NET Web Forms, XML Web Services
Windows Forms
Console
ADO.NET
.NET Remoting
Common Language Runtime [Just-in-Time Compilers, Garbage Collector, Security Manager, and so on]
Common Language Infrastructure (CLI)
Operating System

Figure 2.14: Architecture of .NET framework (MS.NET 2009)

2.7.9. OpenAIR

The OpenAIR specification (Mindmakers 2009; Thórisson et al. 2005) can be applied to routing and communication within large distributed systems based on the publish-subscribe architecture. OpenAIR is an open-source specification of the AIR protocol implemented in C# and Java. Implementations are available for the Windows, Mac, and Linux operating systems. OpenAIR is used to implement an AIR server in Psyclone (Thórisson et al. 2005), a platform for creating large distributed AI projects. OpenAIR messages are in XML format and contain a ‘Type’ slot which is a dot-delimited string. The following is an example of a message type in OpenAir, delimited by XML tags:

<type>Internal.Perception.Hearing.Voice</type>

The dot-delimited string is used by the dispatcher, i.e., a blackboard or whiteboard, to decide which modules are subscribed to which message types, and to route messages accordingly. The asterisk can be used as a wild card when subscribing to messages. For example, a module subscribed to messages of type **Perception** or **Voice** would receive all messages headed by the previous message type. The OpenAIR specification defines a number of other XML tags and attributes that enables the design of publish-subscribe blackboard based architectures.

2.7.10. Psyclone

Psyclone (Thórisson et al. 2005) is a powerful message-based middleware for simplifying the creation of modular, distributed systems. Psyclone is applied in applications where complexity management or interactivity is of importance. Psyclone enables software to be easily distributed across multiple machines and enables communication to be managed through rich messages formatted in XML. A design methodology called, ‘divisible modularity’, enables the incremental construction of multi-granular, heterogeneous systems. Psyclone, written in C++, uses XML configuration files and bulletin boards to enable the easy set-up and testing of system architectures. Psyclone runs on the following platforms:

- Linux
- Windows
- Mac OS X
- PocketPC

Psyclone introduces the concept of a, ‘whiteboard’, which is essentially a blackboard that is capable of handling media streams. A *psySpec* in Psyclone initialises modules and whiteboards at start-up. Figure 2.15 shows an example *psySpec*. The code in Figure 2.15 creates a whiteboard (WB1) and an internal module called ‘Startup’. It is possible to create multiple whiteboards and multiple modules for a system. In order to enable multiple programs to communicate with each other via the whiteboards, programs can use plugs written in Java, C++, Python and Lisp.

2.7.11. Constructionist Design Methodology

Constructionist Design Methodology (CDM) (Thórisson et al. 2004) aids the construction of large AI systems. CDM is based on the principle of multiple interacting modules communicating via message-passing. At the heart of CDM is the notion of modularity, i.e., breaking down the functionalities of the system into software modules whose roles are defined in terms of associated message types and information content.

```
<psySpec name="Project X" version="1.2">

<whiteboard name="WB1">
   <description>This is a basic Whiteboard</description>
</whiteboard>

<module name="Startup">
  <description>Bootstrap by posting a root context upon system
  ready</description>
     <context name="System">
            <phase name="Look for System Created">
                   <triggers from="any">
                          <trigger type="System.Ready" after="100"/>
                   </triggers>
              <posts>
                  <post to="WB1" type="Psyclone.Context:SoB.Alive"/>
              </posts>
            </phase>
     </context>
 </module>

 </psySpec>
```

Figure 2.15: XML Code to initialise Psyclone at start-up (Psyclone 2009)

The methodology follows the idea that the human mind can be effectively modelled through a combination of interacting modules (Mc Kevitt et al., 2002). CDM is particularly suited to facilitating large teams of researchers collaborating on large and complex systems. Previous research has shown that a modular approach can hasten the development of such systems (Fink et al. 1996; Thórisson 1996, 1997; Martinho et al. 2000; Thórisson et al. 2004). To test CDM, Thórisson et al. (2004) chose to develop a system encompassing an embodied virtual character, called Mirage, living in an augmented reality setting. Mirage is shown in Figure 2.16. At the highest level, CDM follows the following fundamental design principles (Thórisson et al. 2004, p. 6):

- “To mediate communication between modules, use one or more blackboards with publish-subscribe functionality.
- Only build functionality from scratch that is critical to the raison d’entre of the system – use available software modules wherever possible”.

Figure 2.16: Embodied agent Mirage (Thórisson et al. 2004)

The implementation of one or more blackboards makes modularity explicit within a system. The fact that modules communicate through a blackboard, as opposed to communicating directly, enables parallel implementation of modules. The second design principle puts emphasis on defining the goal(s) of the project, e.g., practically motivated or research oriented. The following outlines the specific steps of CDM, as discussed in Thórisson et al. (2004):

1. Define the project's goal(s).
2. Define the project's scope.
3. Modularisation – build the system using modules communicating through a blackboard and/or publish-subscribe mechanism.
4. Test the system against scenarios.
5. Iterate – repeat steps 2 to 4 until satisfied that desired functionality is achieved.
6. Assign modules to suitable team members (based on their strengths and areas of interest).
7. Test all modules at run-time (at an early stage in their implementation).
8. Build modules to full specification.
9. Tune the system will all the modules running.

In the implementation of Mirage, Psyclone (Thórisson et al. 2005) was used to implement the blackboard, more precisely, a *whiteboard*, as discussed in Section 2.3, and the publish-subscribe mechanism. Mirage consisted of 8 modules and a total of 5 team members implemented the system. With CDM the design and implementation took only 9 weeks, thus demonstrating the merits of adopting the methodology in the construction of such a large complex system. The distributed processing capabilities of Psyclone (Thórisson et al. 2005) proved very useful in the implementation of Mirage and adherence to the principles of CDM. CDM primarily aims to support the development of large AI systems by a large team of researchers. However, many of its recommendations could be adopted by a single researcher (or small team), provided the system under construction is of sufficient complexity to justify the use of such a methodology.

2.8. *Multimodal platforms and systems*

Several multimodal platforms and systems have been developed that assist the creation of intelligent multimodal systems. These multimodal systems are capable of processing input/output in a wide range of modalities including speech, vision, eye-gaze, facial expression, gesture and tactile. These platforms enable systems to be developed that can communicate with users through the entire range of human communication modalities. The result of enabling such rich multimodal input and output is the development of systems that are more flexible and can adapt to the varying needs of each individual user. In this section several existing multimodal platforms will be discussed, with particular attention given to their methods of semantic representation, semantic storage and decision-making.

2.8.1. Chameleon

Chameleon (Brøndsted et al. 1998, 2001) is a distributed architecture capable of processing multimodal input and output. The system consists of ten modules, mostly programmed in C and C++. The ten modules are listed below:

- Blackboard
- Dialogue Manager
- Domain model
- Gesture recogniser
- Laser system
- Microphone array
- Speech recogniser
- Speech synthesiser
- Natural language processor
- Topsy

The ten modules of Chameleon are glued together by the DACS communication system (Fink et al. 1995, Fink et al. 1996). Chameleon implements a blackboard for semantic storage. The blackboard keeps track of interactions over time, using frames for semantic representation. The architecture of Chameleon is shown in Figure 2.17.

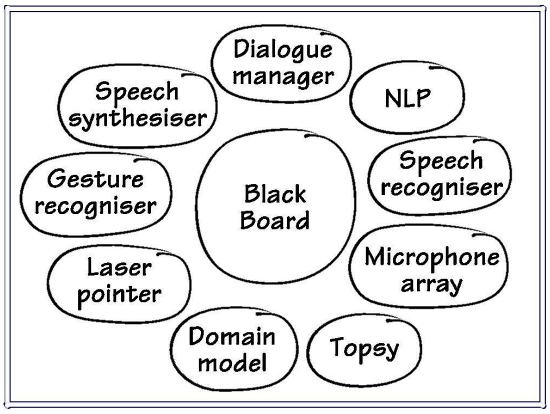


Figure 2.17: Architecture of Chameleon (Brøndsted et al. 1998, 2001)

The blackboard and dialogue manager are collectively the kernel of Chameleon. The blackboard stores the semantic representations produced by other modules, keeping a history of all interactions. Communication between modules is achieved by exchanging semantic representations between themselves or the blackboard. Figure 2.18 shows how the blackboard acts as a mediator for information exchange between the modules of Chameleon.

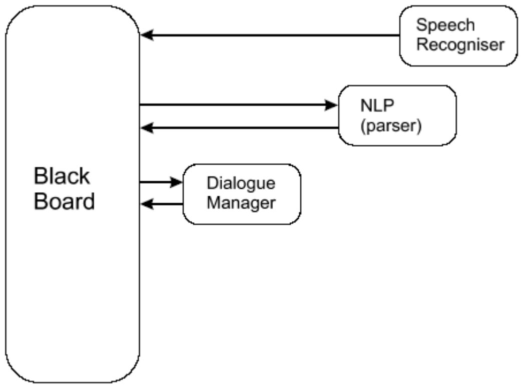


Figure 2.18: Information exchange using the blackboard (Brøndsted et al. 1998, 2001)

The blackboard consists of the following components:

- SQL database
- Communication interface
- Table of implicit requests

The internal architecture of the blackboard is shown in Figure 2.19. The semantic representation is in the form of input, output and integration frames which represent the meaning of user input and system output. Frames are coded as messages and are passed between modules using the DACS communication system. Figure 2.20 defines the predicate-argument syntax of frames, as they appear on Chameleon's blackboard. The code for the frame messages is compiled separately and included in other modules which operate on the representation using a rule-based method of decision-making.

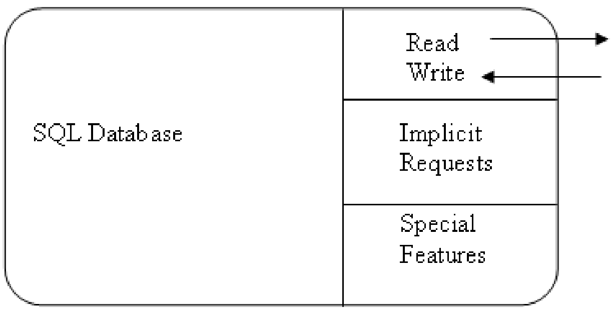


Figure 2.19: Internal blackboard architecture (Brøndsted et al. 1998, 2001)

```
FRAME       ::= PREDICATE
PREDICATE   ::= identifier(ARGUMENTS)
ARGUMENTS   ::= ARGUMENT
            |   ARGUMENTS, ARGUMENT
 ARGUMENT   ::= CONSTANT
            |   VARIABLE
            |   PREDICATE
 CONSTANT   ::= identifier
            |   integer
            |   string
 VARIABLE   ::= $identifier
```

Figure 2.20: Syntax of messages (frames) within Chameleon (Brøndsted et al. 1998, 2001)

The IntelliMedia WorkBench (Brøndsted et al. 2001) is a prototype application for Chameleon. An example domain of application is a Campus Information System for 2D building plans, where the user can ask the system for directions, using speech and pointing gestures, to various offices within a building. The Domain Model, implemented in C, contains all the relevant data for the Campus Information System, including information on rooms, their function and their occupants. Figure 2.21 shows an extract from a file containing the description of the physical environment.

```
area FRB7 1190 0 920
  building environment 0 1190 0 920

  building A1-1 871 1111 497 739
      tp    A1-1s1  725    898      2 A1-2s1 A2-1c1-1
  building A1-2 317 557 497 739
      tp    A1-2s1  725    341      2 A1-1s1 A2-2c1-1
  building A2-1 631 871 497 739
      room   00     624   663   746   860   meeting_room
      tp    A2-100  653    797      1 A2-1c2
      room   01     683   739   787   860   laboratory
      tp    A2-101  693    797      1 A2-1c2
      room   02     663   739   693   787   laboratory
      tp    A2-102  706    746      1 A2-102-2
      tp    A2-102-2  673    746    3 A2-1c2 A2-102 A2-102-4
      tp    A2-102-3  693    704    2 A2-103 A2-102-4
      tp    A2-102-4  673    704    3 A2-102-2 A2-102-3 A2-102-5
      tp    A2-102-5  640    704    3 A2-105 A2-102-4 A2-1c3-1
```

Figure 2.21: Physical environment data (Brøndsted et al. 1998)

Hierarchical levels are illustrated with indentation. As shown, the top level is identified by the keyword, ‘area’, which is followed by the area’s name and minimal/maximal *x* and *y* coordinates. The next level is identified by the keyword, ‘building’, which is again followed by the name and coordinates. Similar files store information about the occupants of the building. The functionality provided by the domain model is to answer questions relating to the environment through search functions. Examples of such functions are:

dm_get_person_coordinates(person_id)

dm_get_place_coordinates(place_id)

dm_get_person_name(person_id)

dm_get_place_name(place_id)

The first two functions return a C struct holding *x* and *y* coordinates, while the last two functions return a string containing the *name* of the person or *place*. Inputs to Chameleon include synchronised spoken dialogue and pointing gestures, and outputs include synchronised spoken dialogue and laser pointing. Topsy, implemented within Chameleon, is a general purpose distributed system for representing real world knowledge as patterns of synchronisation. Topsy consists of: *Sensors*, which represent words and intentions; *Effectors*, which interact with the outside world, through laser and speech synthesis; *Event Windows*, the learning mechanism in Topsy; and *Actions*.

2.8.2. TeleMorph

TeleMorph (Solon et al. 2007) is an inference system that uses fuzzy logic for decision-making on the selection of output for a mobile intelligent multimedia presentation system. TeleMorph dynamically adapts output based on available bandwidth and addresses the problems that typically plague mobile presentation systems, including bandwidth fluctuations and packet loss. A particular focus of the research is transmoding, the adaptation of multimedia content across different modalities. For example, if bandwidth is limited, TeleMorph can replace certain media combinations, e.g., audio and video, with an alternative combination using different modalities, e.g., text and images. TeleMorph defines fuzzy logic rules for deciding when to perform adaptations on output, e.g., from images-text to audio-images, from video to images-text. Matlab's Fuzzy Logic Toolbox (Babuska 1993) was used to implement fuzzy logic within TeleMorph. The architecture of TeleMorph's Fuzzy Inference System is shown in Figure 2.22.

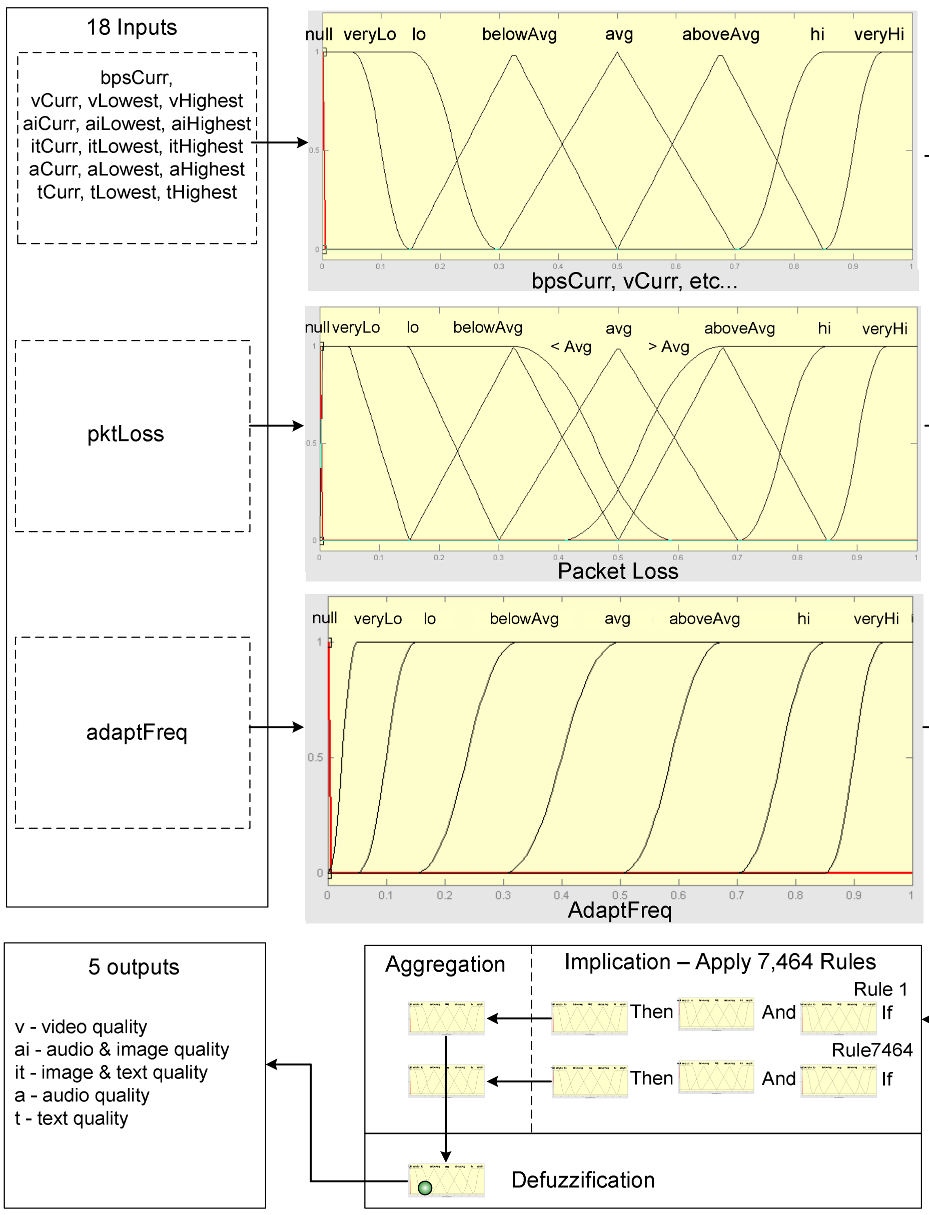


Figure 2.22: Architecture of TeleMorph's Fuzzy Inference System (Solon et al. 2007)

As shown in Figure 2.22, TeleMorph's Fuzzy Inference System defines 7 membership functions which provide a sufficient level of granularity for decision-making within the system. TeleTuras, a testbed for TeleMorph, is a mobile multimedia presentation system, which provides tourist information about the city of Derry. TeleTuras enables users to ask questions such as, "Where is the University of Ulster?", and responds by presenting multimedia content that automatically adapts in response to fluctuations in the bandwidth of the mobile network. A number of bandwidth specific test scenarios, created to test the transmoding capabilities of TeleMorph, have given positive results.

2.8.3. CONFUCIUS

CONFUCIUS (Ma 2006) is an intelligent storytelling system that performs language visualisation of English sentences. The architecture of CONFUCIUS is shown in Figure 2.23. A key problem addressed by CONFUCIUS is the mapping between language knowledge and visual/audio knowledge. CONFUCIUS can generate output animation based on speech input. For example, the utterance, "John put a cup on the table", causes CONFUCIUS to produce the output animation presented in Figure 2.24.

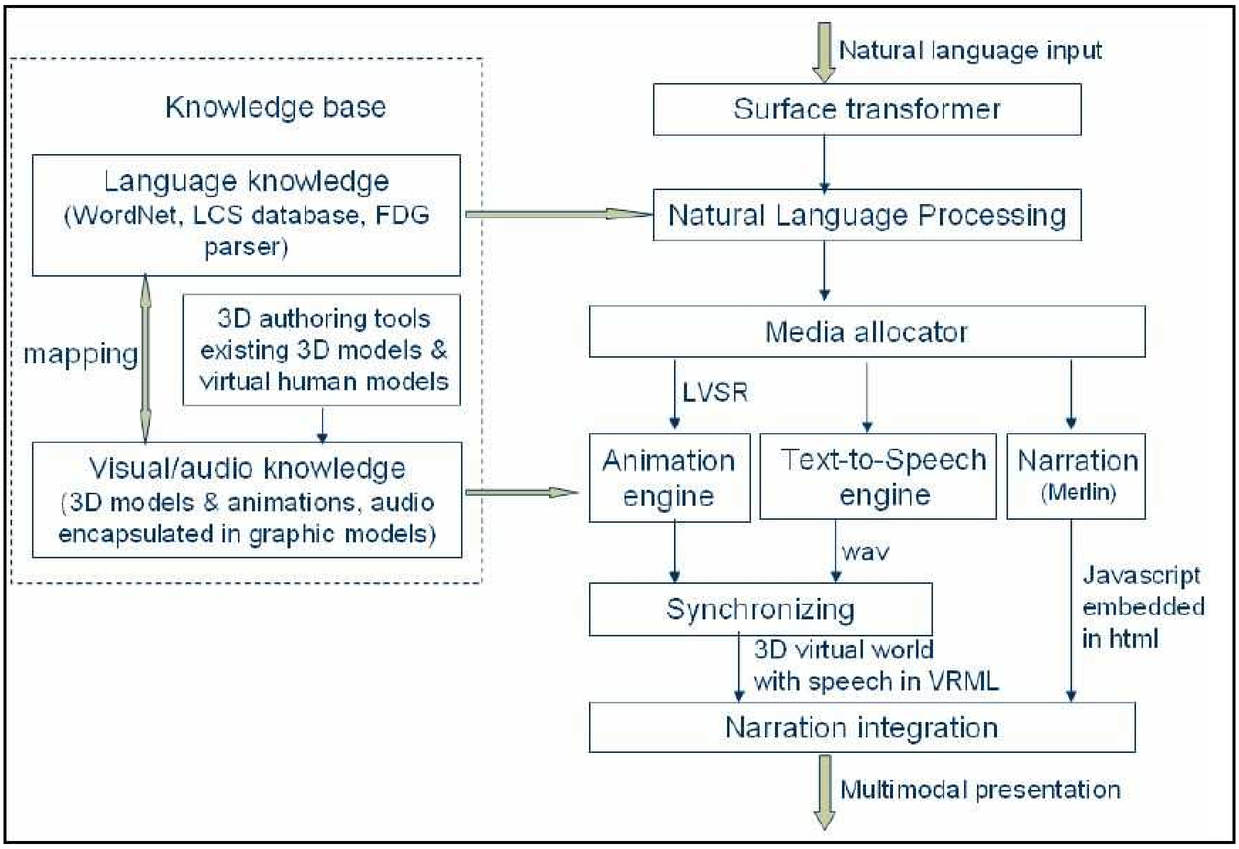

Natural language input
Knowledge base
Surface transformer
Language knowledge (WordNet, LCS database, FDG parser)
Natural Language Processing
mapping
3D authoring tools existing 3D models & virtual human models
Media allocator
LVSR
Visual/audio knowledge (3D models & animations, audio encapsulated in graphic models)
Animation engine
Text-to-Speech engine
Narration (Merlin)
wav
Javascript embedded in html
Synchronizing
3D virtual world with speech in VRML
Narration integration
Multimodal presentation

Figure 2.23: Architecture of CONFUCIUS (Ma 2006)

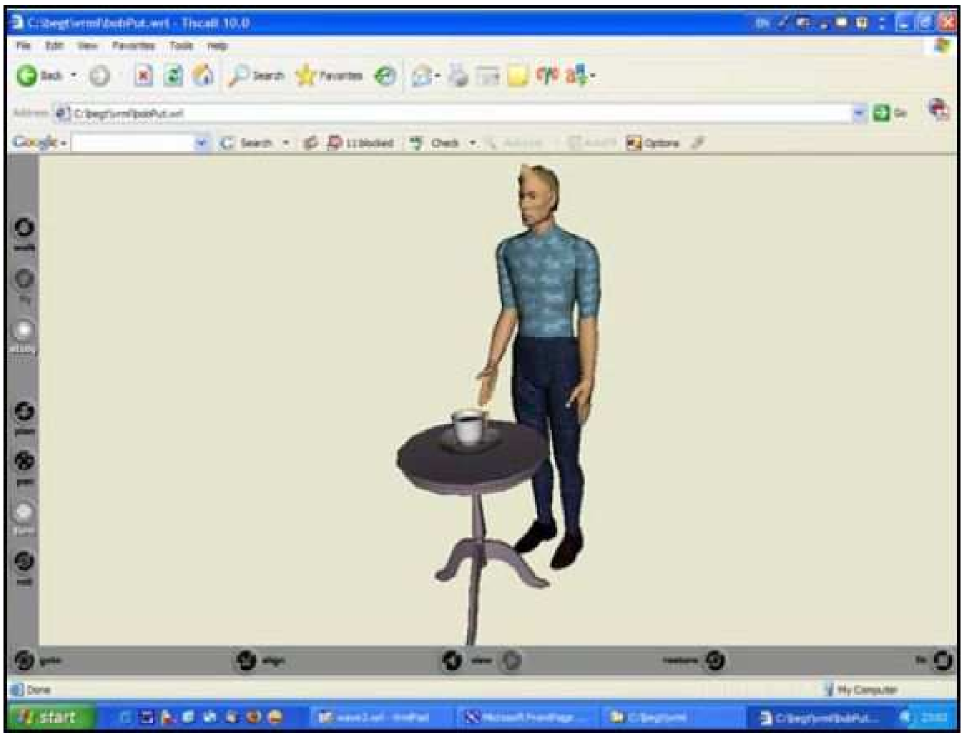


Figure 2.24: Output animation in CONFUCIUS (Ma 2006)

CONFUCIUS also includes a presentation agent called Merlin, shown in Figure 2.25, who acts as a narrator.

Figure 2.25: Narrator Merlin in CONFUCIUS (Ma 2006)

2.8.4. Ymir

Ymir (Thórisson 1996, 1999) is a multimodal platform for the creation of autonomous agents capable of human-like communication. Ymir represents a distributed, modular approach that implements a coherent framework to bridge between multimodal perception, decision-making and action. Within the Ymir architecture, a prototype agent called Gandalf (Thórisson 1996) has been created. The interactive agent, Gandalf, is capable of fluid turn-taking and dynamic sequencing. The modules within Ymir are divided into the following four process collections:

- The Reactive Layer (RL), which operates on relatively simple data.
- The Process Control Layer (PCL), which controls the global aspects of the dialogue and manages the communicative behaviour of the agent.
- The Content Layer (CL), which hosts the processes that interpret the content of the multimodal input and generates suitable responses.
- The Action Scheduler (AS), which coordinates appropriate actions.

Three main blackboards are implemented in Ymir. The first blackboard, called the Functional Sketchboard, is primarily used to exchange information between the Reactive Layer and the Process Control Layer. The Content Blackboard deals with communication between the Process Control Layer and the Content Layer. The messages that are posted on the Content Blackboard are less time-critical than those posted on the Functional Sketchboard. The third blackboard is called the Motor Feedback Blackboard and keeps track of which part of an action stream is presently being planned or carried out by the Action Scheduler (AS). Ymir uses a frame-based method of semantic representation. Within Ymir, the majority of messages take the form *[<Message>,<State>,<Timestamp]*, with state being either true or false. Figure 2.26 shows an example of a message posted on the Functional Sketchboard. The message can either be the form of a decision or a perception. For example, *I-GIVE-TURN* is a decision, whilst *HAND-IN-GEST-SPACE* indicates a belief about the dialogue. It is expected that Ymir's modular structure will allow systems to be easily extended, without affecting the simplicity or performance of the system. Ymir has been applied in the development of the Honda ASIMO humanoid robot (Ng-Thow-Hing et al. 2008) and, according to Thórisson (1999), looks promising in providing a general framework for communicative, task-knowledgeable agents.

(RHAND-IN-GEST-SPACE T 5140)
(HAND-IN-GEST-SPACE T 5150)
(R-DEICTIC-MORPH T 5180)
(SPEAKING T 5180)
(USER-TAKING-TURN T 5190)
(I-TAKE-TURN NIL 5330)
(I-GIVE-TURN T 5340)

Figure 2.26: Frame posted on Ymir's Functional Sketchboard (Thórisson 1999)

2.8.5. InterACT

InterACT (Waibel et al. 1996) aims to remove the limitations associated with traditional computer interfaces by enabling input using the entire range of human communication modalities, including speech, gesture, eye-gaze, face recognition, facial expression, lip motion, handwriting and sound localisation. InterACT uses frames for semantic representation and implements a non-blackboard model of semantic storage. An initial application of InterACT is an Audio/Visual Automated Speech Recognition (ASR) System. This involves the recognition of letters in the German alphabet using automated lip reading and speech recognition. A Multi-State Time-Delay Neural Network (MS-TDNN) performs recognition of visual data. The network architecture of InterACT is shown in Figure 2.27, where the acoustic and visual inputs are processed in isolation. The audio and visual inputs are then combined for further processing in the combined layer.

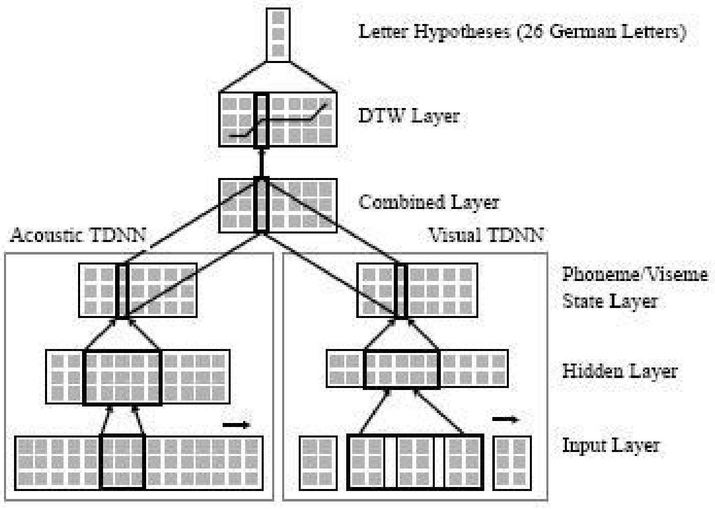


Figure 2.27: Network architecture of InterACT (Waibel et al. 1996)

Table 2.1 shows recognition performances for InterACT. The results show that when visual information is added to speech the overall recognition rate can be significantly improved. It can also be seen that, as expected, the improvement is greatest when the speech input is noisy.

Speaker	Acoustic	Visual	Combined
msm/clean	88.8	31.6	93.2
msm/noisy	47.2	31.6	75.6
mcb/clean	97.0	46.9	97.2
mcb/noisy	59.0	46.9	69.6

Table 2.1: Word accuracy of Speech/Lip system (Waibel et al. 1996)

2.8.6. JASPIS

JASPIS (Turunen & Hakulinen 2000; Jokinen et al. 2002) focuses on the exploration of natural human-computer interaction and the development of natural adaptive dialogue models. Another key objective of JASPIS is adaptivity - the ability of the system to adapt to the changing needs and activities of a user, e.g., in a mobile environment. An agent-based architecture has been developed in pursuit of these objectives. The architecture of JASPIS is depicted in Figure 2.28 where different agents focus on specific tasks within the system. Using these specialised agents modular, reusable interaction components are implemented. An information storage knowledge base acts similar to a blackboard, with other system components accessing the knowledge base via the Information Manager. An Interaction Manager facilitates interactions between the modules of JASPIS. An unlimited set of modules can be connected to JASPIS and the Interaction Manager caters for all connections between modules.

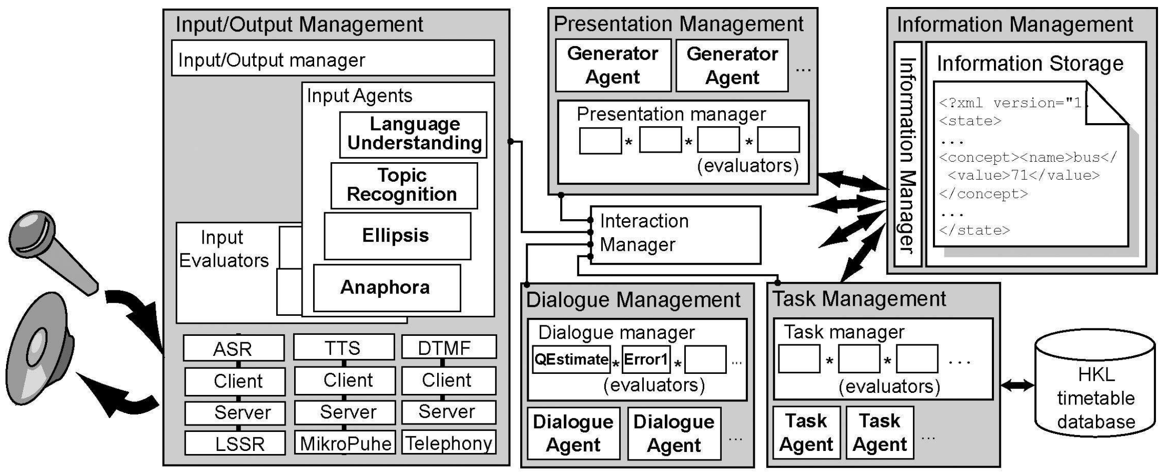


Figure 2.28: Architecture of JASPIS (Jokinen et al. 2002)

The architecture also enables JASPIS to be distributed over multiple computers. Agents within JASPIS have different capabilities and the most suitable agent for performing a given task can be selected dynamically at run-time based on its capabilities and the current context. Evaluators determine which agent is best suited to deal with a particular situation. The decision-making process involves the evaluator giving a score between zero and one to each of the agents. A score of zero indicates the agent is not suited to the situation, a score of one means the agent is deemed perfectly suited, whilst values between zero and one indicate the degree of suitability. Several evaluators give scores to an agent, indicating its suitability for the current situation. Scaling functions can give greater importance to certain evaluators, before the scores are multiplied to give final scores, or suitability factors, for each of the agents. Multiple evaluations are performed before an agent is chosen for a particular task. JASPIS uses an XML-based method of semantic representation and, through the use of a shared knowledge base, implements a blackboard-style method of semantic storage. An early application of JASPIS is an intelligent bus-stop that allows multimodal access to city transport information.

2.8.7. SmartKom

SmartKom (Wahlster 2003, 2006; Wahlster et al. 2001; SmartKom 2009) is a multimodal dialogue system that helps overcome the problems of interaction between people and machines. SmartKom focuses on developing multimodal interfaces for applications in the home, public and mobile domains. SmartKom uses a combination of speech, gestures and facial expressions to facilitate more natural human-computer interaction, enabling face-to-face interaction with its conversational agent Smartakus. For example, in the public domain, the user can allocate to Smartakus the task of finding a library. Together with its language capabilities, Smartakus uses facial expressions and body language to improve the naturalness of the interaction. It is intended that SmartKom will enable complex dialogic interactions, where both the user and the system will be capable of initiating interactions, asking questions, requesting clarification, signalling problems of understanding and interrupting the dialogue partner. SmartKom enables the following two modes of interaction:

- ‘Lean-forward’ mode, which supports touch and visual input.
- ‘Lean-back’ mode, where input and output is only achieved via the speech channel.

An XML-based mark-up language, M3L (MultiModal Mark-up Language), provides semantic representation of information passed between components of SmartKom. An example of the M3L code within SmartKom is shown in Figure 2.29. SmartKom also makes use of the OIL ontology language (Fensel et al. 2001) to represent domain and application knowledge. SmartKom constitutes a distributed multi-blackboard system, including more than 40 asynchronous modules coded in C, C++, Java and Prolog. The integration platform for SmartKom is called MULTIPLATFORM (MUltiple Language Target Integration PLATform FOR Modules) (Herzog et al. 2003), which enables the creation of open, flexible and scalable software architectures. MULTIPLATFORM uses a message-based middleware, based on the Parallel Virtual Machine (PVM), to provide a powerful framework for creating integrated multimodal dialogue systems. The ultimate aim of SmartKom is to provide a kernel system that can be utilised within different application scenarios.

2.8.8. DARPA Galaxy Communicator

The DARPA Galaxy Communicator project (Bayer et al. 2001) investigates ways to engage humans in robust, mixed-initiative spoken interactions, which would surpass the capabilities of current dialogue systems. This project has involved the development of a distributed message-passing infrastructure for dialogue systems, namely the Galaxy Communicator Software Infrastructure (GCSI).

```
<presentationTask>
   <presentationGoal>
      <inform> <informFocus> <RealizationType>list
</RealizationType> </informFocus> </inform>
<abstractPresentationContent>
<discourseTopic> <goal>epg_browse</goal> </discourseTopic>
<informationSearch id="dim24"><tvProgram id="dim23">
<broadcast><timeDeictic id="dim16">now</timeDeictic>
<between>2003-03-20T19:42:32 2003-03-20T22:00:00</between>
<channel><channel id="dim13"/> </channel>
</broadcast></tvProgram>
</informationSearch>
<result> <event>
<pieceOfInformation>
<tvProgram id="ap_3">
<broadcast> <beginTime>2003-03-20T19:50:00</beginTime>
<endTime>2003-03-20T19:55:00</endTime>
<avMedium> <title>Today’s Stock News</title></avMedium>
<channel>ARD</channel>
</broadcast>……..</event>
</result>
</presentationGoal>
</presentationTask>
```

Figure 2.29: Example of M3L code (Wahlster 2003)

An extension of the Galaxy-II distributed infrastructure for dialogue interaction, the GCSI is a distributed hub-and-spoke architecture. Semantic representation is managed by frames and communication is facilitated via message-passing. The architecture of the GCSI is illustrated in Figure 2.30.

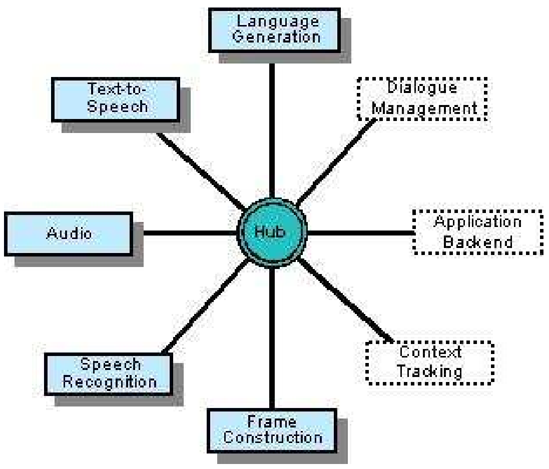


Figure 2.30: Hub-and-spoke architecture of GCSI (Bayer et al. 2001)

The GCSI's hub enables programmers to create programs using a simple scripting language that can control message traffic. The message-passing nature of the GCSI infrastructure means that the hub doesn't need to have any compile-time knowledge of the functional properties of the server it is communicating with. A scripting language enables the programmer to alter the flow of messages, allowing the integration of servers with a variety of interaction paradigms without making modifications to the servers themselves. This property also enables tools and filters to be inserted that can convert data between different formats. Included in the GCSI are libraries and templates, allowing Communicator-compliant servers to be created in C, Java, Python and Allegro Common Lisp.

2.8.9. Waxholm

Waxholm (Carlson & Granström 1996; Carlson 1996) combines speech synthesis and recognition in a human-computer dialogue framework. Waxholm gives information on boat traffic. Besides the spoken language capabilities, Waxholm has modules to deal with graphical information such as pictures, maps, charts and timetables. Waxholm uses SQL to access information, as requested by the user. During a dialogue the decision on which topic path to follow is based on dialogue history and the content of the utterance. Using a rule-based system, the utterance is encoded in a semantic frame with slots relating to both the grammatical analysis of the utterance and the current application domain. In order to decide on the topic, the semantic features found in the semantic and syntactic analysis are considered in the form of conditional probabilities. Probabilities are expressed in the form $p(topic \mid F)$, where F is a feature vector containing all the semantic features found in the utterance. The topic prediction is trained with utterances from the Waxholm database. Waxholm implements a non-blackboard model of semantic storage.

2.8.10. Spoken Image (SI)/SONAS

Spoken Image (SI) and its successor, SONAS (Ó Nualláin et al. 1994, Ó Nualláin & Smith 1994, Kelleher et al. 2000), are systems for interacting with a 3D environment through natural language, gestures and other modes of communication. SI and SONAS are systems that enable users to communicate and interact with them in a multimodal manner, inspired by the way humans communicate with ease through multiple modalities. The original project, Spoken Image, enabled a user to quickly build a house or town scene by describing the scene with natural language. The user could then refine the details until the presented scene matches how the user has envisioned it. Each element of a scene is an instance of a class implemented in C++. SONAS is an intelligent multimedia system that enables input comprising a combination of several modalities. SONAS enables objects in a 3D environment to be manipulated with natural language. Consider the following example, as discussed in Kelleher et al. (2000). With the input, “Put the book on the table”, the user sees the book moving onto the table. In order to achieve this, the following steps are necessary:

- The input phrase is parsed and broken down into the figure, “the book”, the reference object, “the table”, the action, “put”, and the spatial relation, “on”.
- Next the visual model is searched for the figure and reference objects.
- Then, at the conceptual level, the objects are considered with respect to their position in the physical ontology of objects.
- At the semantic level the objects are reduced to, e.g., geometric points, lines and planes.

An important theme in developing SI/SONAS has been an effort to find a common semantics between language and vision, i.e., to develop a meaning representation scheme that is common to both the language and vision data. This presents many challenges as the same word can mean different things in different situations. For example, the word, “park”, can have different meanings, e.g., play park, park the car. SI/SONAS uses frames for semantic representation and uses a blackboard for semantic storage.

2.8.11. Aesopworld

Aesopworld (Okada 1996, Okada et al. 1999) aims to create an architectural foundation of intelligent agents. Aesopworld involves the creation of a computational agent that simulates various kinds of mental activities. A key objective of Aesopworld is the development of human-friendly interfaces that can make decisions on a dialogue based on the user’s facial expressions, gestures and the tone of their voice. Aesopworld employs a frame-based method of semantic representation and a non-blackboard method of semantic storage. An example Aesopworld frame is shown in Figure 2.31. In its efforts to develop a truly intelligent agent, Aesopworld attempts to integrate seven intelligent activities: recognition, planning, action, desire, emotion, memory and language.

2.8.12. Collagen

Collagen (Rich & Sidner 1997) introduces the concept of a SharedPlan to represent the common goal of a user and a collaborative agent. Grosz and Sidner’s (1990) theory maintains that, in order to achieve successful collaboration, it is necessary that participants have mutual beliefs about the goals/actions that must be performed and the capabilities/intentions of the participants. In addition to the concept of a SharedPlan, a recipe within Collagen is defined as an agreed sequence of actions necessary to accomplish a common goal.

```
μ-agent(
     name(evt_get_out_of),
     domain(dom_evt_recognition),
     description(...),
     input(
      msg(subs,evt_get_out_of,C_agent,(natural,movable)),
      msg(subs,evt_get_out_of,C_origin,(artificial,has_inside))),
     execution(
      event_extraction(
           concept_feature_1([lapse(Before,After)]),
           concept_feature_2([existence([C_agent,(Before,After)]),
           concept_feature_3([existence([C_origin,(Before,After)]),
           concept_feature_4([inside(C_agent,C_origin,Before)]),
           concept_feature_5([movement([C_agent,(Before,After)]),
           concept_feature_6([outside([C_agent,C_origin,After)]))),
     output()).
```

Figure 2.31: An example Aesopworld frame (Okada et al. 1999)

Data structures and algorithms are provided within Collagen to represent and manipulate goals, actions, recipes and SharedPlans. Figure 2.32 illustrates how a user can collaborate with an agent using Collagen, whilst Figure 2.33 shows the internal architecture of Collagen.

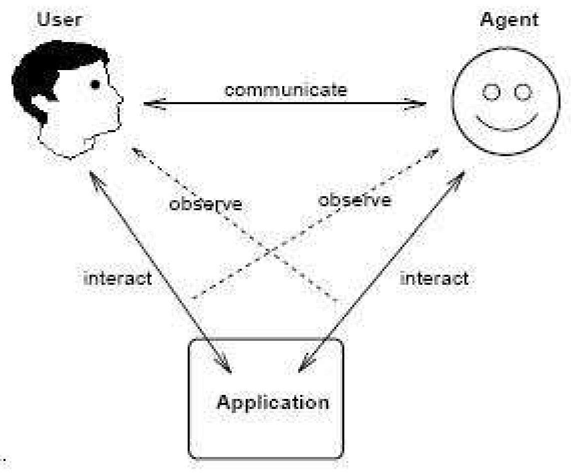


Figure 2.32: User-Agent Collaboration within Collagen (Rich & Sidner 1997)

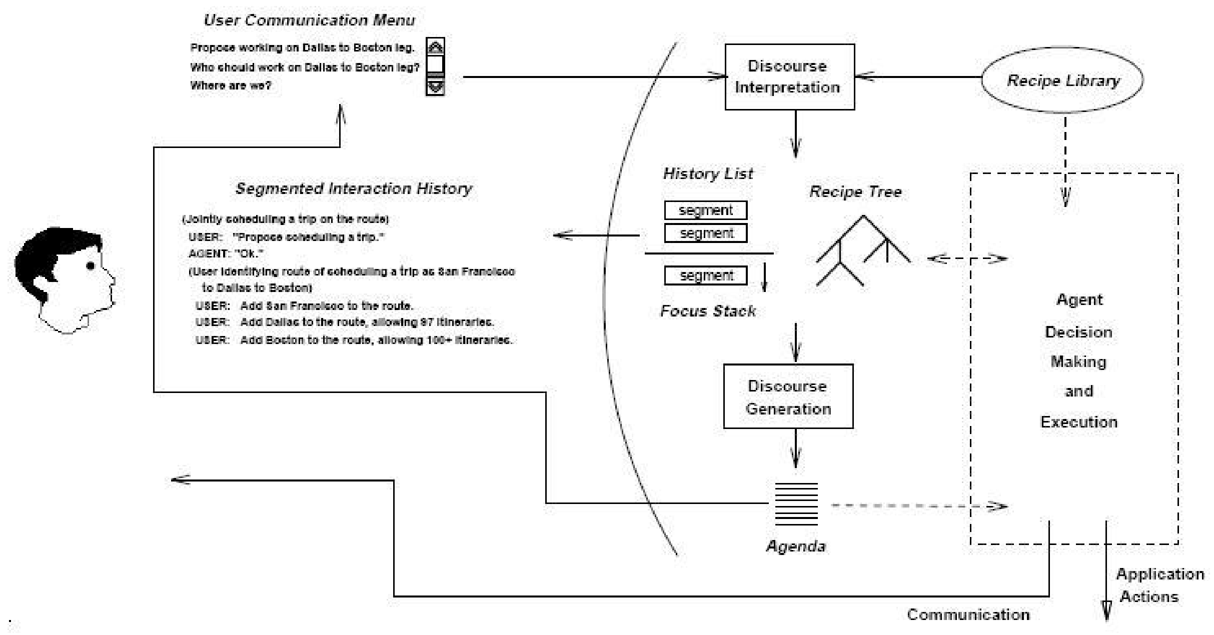

User Communication Menu
Propose working on Dallas to Boston leg.
Who should work on Dallas to Boston leg?
Where are we?
Segmented Interaction History
(Jointly scheduling a trip on the route)
USER: "Propose scheduling a trip."
AGENT: "Ok."
(User identifying route of scheduling a trip as San Francisco to Dallas to Boston)
USER: Add San Francisco to the route.
USER: Add Dallas to the route, allowing 97 itineraries.
USER: Add Boston to the route, allowing 100+ itineraries.
Discourse Interpretation
Recipe Library
History List
segment
segment
segment
Focus Stack
Recipe Tree
Agent Decision Making and Execution
Discourse Generation
Agenda
Communication
Application Actions

Figure 2.33: Collagen architecture (Rich & Sidner 1997)

Note from Figure 2.33, that the agent's decision-making and execution is a, 'black box'. That is, although Collagen provides a framework for communicating and recording decisions between the user and an agent, it does not offer a means of decision-making – this is left to the discretion of the developer. Collagen uses Sidner's (1994) artificial discourse language to represent agent communication acts. Within the artificial discourse language there is a set of constructors for basic act types, e.g., proposing, accepting and rejecting proposals. Examples of such act types are *PFA* (*Propose For Accept*) and *AP* (*Accept Proposal*). The syntax of a PFA is as follows:

PFA (t, participant$_1$*, belief, participant*$_2$*)*

The above states that at time *t*, *participant*$_1$ has a *belief*, communicates it to *participant*$_2$ with the intention that *participant*$_2$ will believe it also. If *participant*$_2$ now responds with an AP act, i.e., accepts the proposal, then the *belief* is considered to be mutually believed. There are two additional application-independent operators to model a belief about an action, *SHOULD (act)* and *RECIPE (act, recipe)*. The remainder of the belief sublanguage is application-specific. Collagen implements a frame-based method of semantic representation and a non-blackboard model for semantic storage.

2.8.13. Oxygen

Oxygen (Oxygen 2009) is motivated towards making computing available to everyone, everywhere in the world – just as accessible as the oxygen we breathe. Some of the aims of Oxygen are the development of a system that is:

- Human-centred and directly addresses human needs.
- Pervasive, i.e., all around us.
- Embedded in the world around us, sensing and affecting it.
- Nomadic, i.e., allowing users and computations to move around freely as necessary.
- Adaptable to changes in user requirements.
- Intentional, i.e., enabling people to name a service or software object by intent, e.g., "the closest printer", as opposed to by address.

The meeting of these objectives creates a system that adapts to the needs of the user, as opposed to traditional computer systems that force the user to learn how to interact with the machine using the keyboard and mouse. Oxygen aims to enable pervasive, human-centred computing by integrating various technologies that address human needs. Within Oxygen, spoken language and visual cues form the main modes of user-machine interaction. Speech and vision technologies are used to enable the user to interact with the system as if communicating with another person. Knowledge access technology allows information to be found quickly by remembering what the user looked at previously. Semantic representation is in the form of frames, whilst semantic storage is implemented with a non-blackboard model.

2.8.14. DARBS

DARBS (Distributed Algorithmic and Rule-Based System) (Choy et al. 2004a,b; Nolle et al. 2001) is a distributed system that enables several knowledge sources to operate in parallel to solve a problem. DARBS is an extension of ARBS, which was first developed in 1990. The original ARBS system only enabled one knowledge source to operate at any one time. A distributed version of the system was designed to deal with more complicated engineering problems. DARBS, programmed in standard C++, consists of a central blackboard with several knowledge source clients. A client is a separate process that may reside on a separate networked computer and can contribute to solving a problem when it has a contribution to make. Figure 2.34 shows the architecture of DARBS. As shown, DARBS comprises rule-based, procedural, neural network and genetic algorithm knowledge sources operating in parallel. DARBS uses frames for semantic representation. The major advantage that DARBS offers over its predecessor is parallelism. Knowledge about a problem is distributed across the client knowledge sources, with each of the clients seen as an expert in a specific area. DARBS implements client/server technology, with standard TCP/IP used for communication. The independent clients can only communicate via the central blackboard. This is illustrated in Figure 2.35.

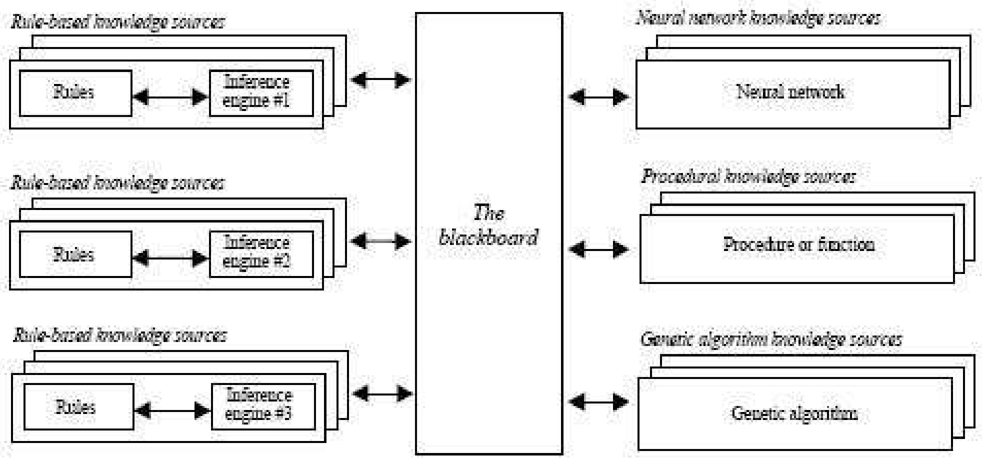

Rule-based knowledge sources
Rules
Inference engine #1
Rule-based knowledge sources
Rules
Inference engine #2
Rule-based knowledge sources
Rules
Inference engine #3
The blackboard
Neural network knowledge sources
Neural network
Procedural knowledge sources
Procedure or function
Genetic algorithm knowledge sources
Genetic algorithm

Figure 2.34: Architecture of DARBS (Nolle et al. 2001)

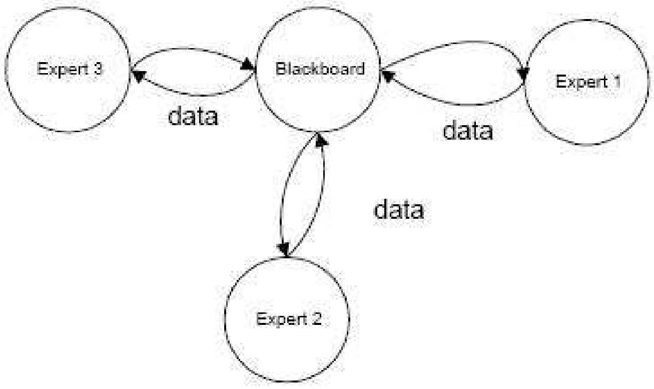


Figure 2.35: Communication within DARBS (Nolle et al. 2001)

The DARBS knowledge sources constantly examine the blackboard and only activate themselves when the information is of interest to them. Thus the knowledge sources are deemed to be completely opportunistic and will activate themselves when they have a contribution to make. Rules within DARBS facilitate looking up information on the blackboard, writing information to the blackboard and making decisions about information on the blackboard. An example of a typical DARBS rule is shown in Figure 2.36. In order to demonstrate its flexibility, DARBS has been applied to several different AI applications, including interpreting ultrasonic non-destructive evaluation (NDE) and controlling plasma processes.

2.8.15. EMBASSI

The EMBASSI project (Kirste et al. 2001, EMBASSI 2009) aims to provide a platform that will give computer-based assistance to a user in achieving his/her individual objectives, i.e., the computer will act as a mediator between users and their personal environment.

```
RULE ghost_echo_prediction_rule
IF
[
[on_partition [?centre1 is the CENTRE of the AREA == corners ~area1]
setsoflinechars]
AND
[on_partition [?centre2 is the CENTRE of the AREA == corners ~area2]
setsoflinechars]
]
THEN
[
[add [ghost echoes for centres ~centre1 and ~centre2 expected to pass thru
~[run_algorithm [ghostecho_predict [~centre1 ~centre2]] coords]]
prediction_list]
[report [ghost echoes for centres ~centre1 and ~centre2 expected to pass
thru ~coords] nil]
]
BECAUSE [~centre1 is the centre of the area]
END
```

Where:
The match variable, which is prefixed by a "?", will be looked up from the blackboard;
The insert variable, which is prefixed by a "~", will be replaced by the instantiations of that variable.

Figure 2.36: A typical DARBS rule (Nolle et al. 2001)

The ideas of human-computer interaction and human-environment interaction take focus in the EMBASSI project and effort is made to allow humans to more easily interact with their environment through the use of computers. This concept is illustrated in Figure 2.37, which shows the relationship between the user, the computer and the user's personal environment.

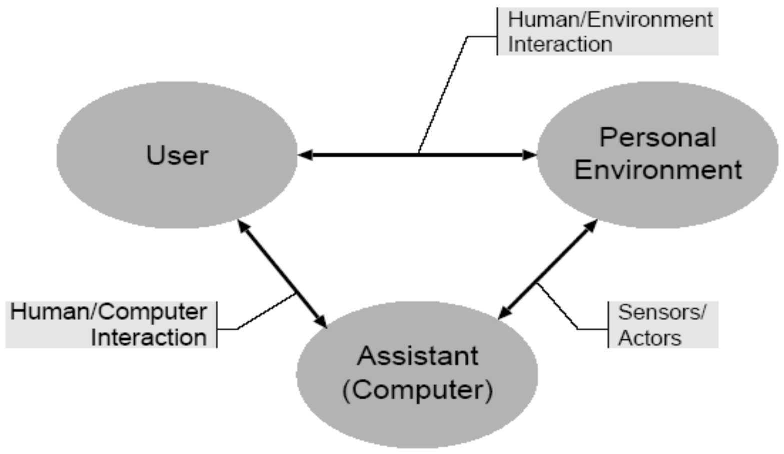


Figure 2.37: User-computer-environment relationship (Kirste et al. 2001)

Another important concept in the EMBASSI project is the idea of goal-based interaction, where the user need only specify a desired effect or goal and doesn't need to specify the actions necessary to achieve the goal. For example, a goal could be, "I want to watch the news". In response to the user’s goal, the system would then fill in the sequence of necessary actions to achieve this goal. Thus, a major function of the EMBASSI framework is the translation of user utterances into goals. The generic EMBASSI architecture used to achieve this is shown in Figure 2.38.

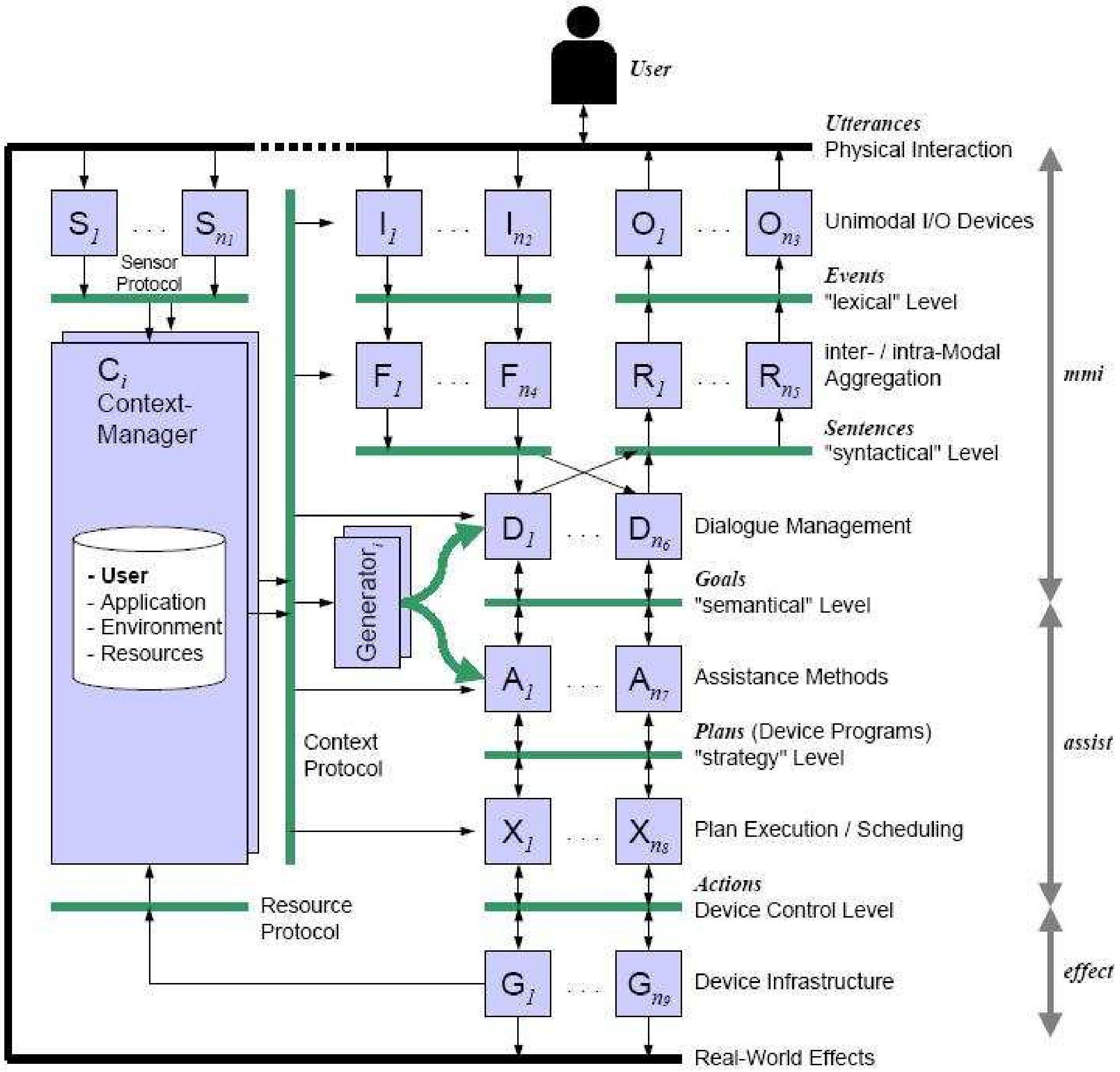


Figure 2.38: Generic architecture of EMBASSI (Kirste et al. 2001)

As shown in Figure 2.38, the MMI levels determine the goals of users from their utterance. The assistance levels are then responsible for mapping these goals to actual changes in the environment, i.e. real-world effects, such as showing the news. Below the EMBASSI protocol suite, the EMBASSI project makes use of existing standards. KQML (Knowledge Query and Manipulation Language) Agent Communication Language (ACL) (Finin et al. 1994) acts as a messaging infrastructure, whilst XML (eXtensible Mark-up Language) (W3C XML 2009) acts as the content language. A non-blackboard based model of semantic storage is implemented within EMBASSI. The platform has been tested in three main technical environments – the home, automotive and public (terminal) environments. For example, in the home environment there is the, 'living room scenario', which involves the management of home entertainment infrastructures and the control of, e.g., lighting, temperature within the room. Another scenario, this time in the car domain, is the operation of the car radio where the user could use natural language to request a suitable station, e.g., "I want a station with traditional Irish music". Many other scenarios are possible where the user can simply express a goal and leave the required technical functionality to the EMBASSI platform.

2.8.16. MIAMM

MIAMM (Multidimensional Information Access using Multiple Modalities) (Reithinger et al. 2002; MIAMM 2009) facilitates fast and natural access to multimedia databases using multimodal dialogues. A multimedia framework for designing modular multimodal dialogue systems has been created. MIAMM offers a considerable benefit to the user in that access to information systems can be made easier through the use of a flexible intelligent user interface that adapts to the context of the user query. The MIAMM platform is based upon a series of interaction scenarios that use various modalities for multimedia interaction. Integrated within the platform is a haptic and tactile device for multidimensional interaction. This enables the interface to create tactile sensations on the skin of the user and to add the sensation of weight to the interaction. The result is a more natural user interface, with haptic technology applied where the eyes and ears of the user are focused elsewhere. The MIAMM architecture is shown in Figure 2.39.

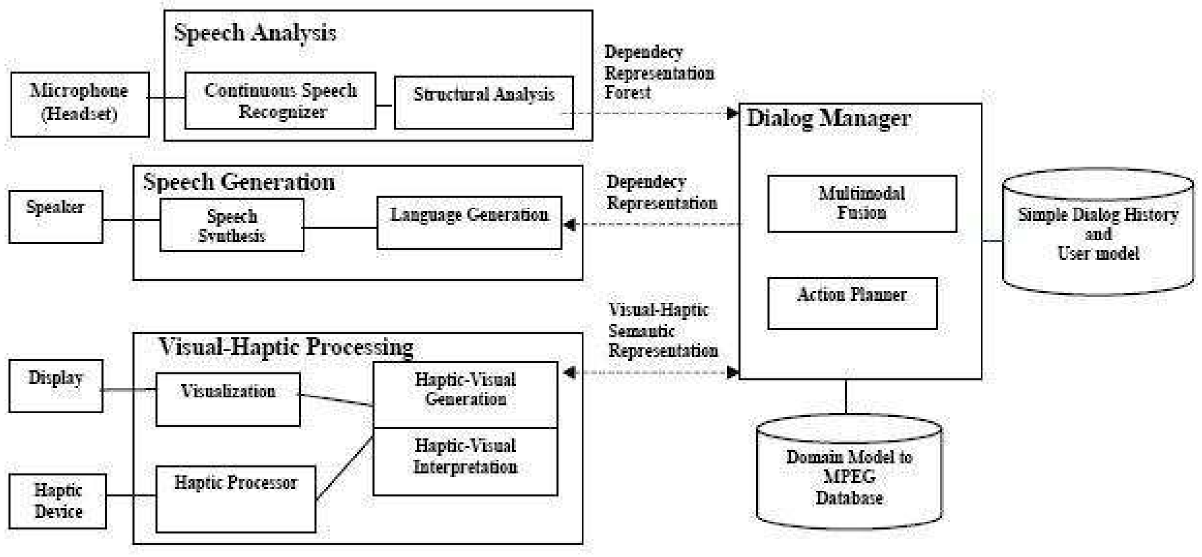


Figure 2.39: MIAMM architecture (Reithinger et al. 2002)

The exchange of information within MIAMM is facilitated through the XML-based Multi-Modal Interface Language (MMIL). MMIL comprises, amongst other components, information on gesture trajectory, speech recognition and understanding, as well as information specific to each individual user. A key objective of MMIL is to enable the incremental integration of multimodal data to provide a full understanding of the user's multimodal input, i.e., speech or gesture, and to provide the necessary information for an appropriate system response (spoken output and graphical or haptic feedback). MIAMM implements a non-blackboard based model of semantic storage. Within MIAMM, a dialogue manager combines information from the underlying application, the haptic device, the language modules and the graphical user interface. As an example, suppose the user says, "Show me the song that I was listening to this morning". Now, assuming the user has listened to some music in the morning, the utterance will be analysed and an intention based MMIL representation will be produced. MIAMM first retrieves the lists of songs from the dialogue history. The action planner then identifies displaying the list as the next system goal, passing the goal and the list to the visual-haptic agent. The interface shown in Figure 2.40 is then presented to the user. When the user has highlighted the desired track using the selection buttons on the left, he/she can select the song by simultaneously uttering, "I want this one", and clicking the selection button on the right. Now both the Speech Analysis and Visual-Haptic Processing agents send time-stamped MMIL representations to the dialogue manager. Multimodal fusion then checks time and type constraints of each structure and the action planner invokes the domain model to retrieve the relevant information from the database. Finally, the action planner sends a display order to the visual-haptic agent.

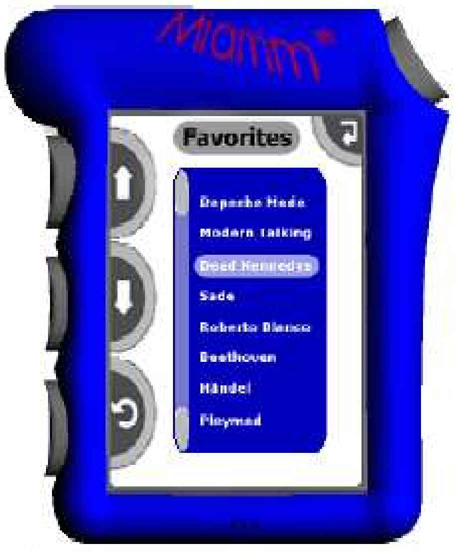


Figure 2.40: Example MIAMM hand-held device (Reithinger et al. 2002)

2.8.17. XWand

XWand (Wilson & Shafer 2003; Wilson & Pham 2003) is an intelligent wand which employs Bayesian networks to control devices in the home environment, e.g., lights, hi-fis, televisions. XWand has been designed to help speed the day of truly intelligent environments – where computational ability will reside in everyday devices, enabling the creation of powerful integrated intelligent environments. XWand addresses the problem of selecting one of several devices in an intelligent environment by adopting the notion of the computing curser and using this familiar point-and-click paradigm in the physical world. With XWand users can select and control several networked devices in a natural way. For example, users can point at a lamp and press a button on the XWand to turn it on. The XWand is shown in Figure 2.41.

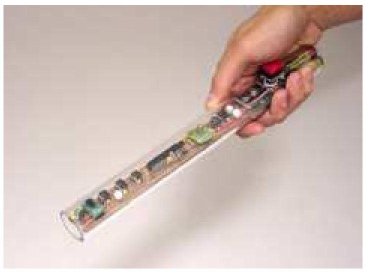


Figure 2.41: The XWand (Wilson & Shafer 2003)

In the XWand Dynamic Bayesian networks perform multimodal integration. The Dynamic Bayesian network determines the next action by combining wand, speech and world state inputs (Wilson & Shafer 2003). The technology offered by the XWand has been enhanced in the WorldCursor system (Wilson & Pham 2003). WorldCursor uses the XWand but removes the need for a geometric model, and hence the 3D position of the wand, instead using projection of a laser spot to indicate where the user is pointing, as believed by the system. A laser pointer is mounted on a motion platform, which in turn is mounted on the ceiling. The motion platform steers the laser point onto objects pointed to by the XWand. The WorldCursor motion platform is illustrated in Figure 2.42.

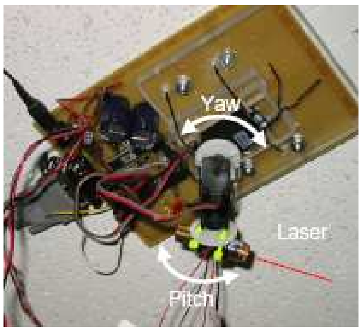


Figure 2.42: WorldCursor motion platform (Wilson & Pham 2003)

User testing of WorldCursor and the XWand has shown that users use the XWand similar to the way they use a mouse. The user seldom looks at the XWand itself, instead focusing on the laser dot (or cursor). Hence the familiar point-and-click paradigm of interaction has been successfully transferred from the desktop computing environment into the physical world.

2.8.18. COMIC/ViSoft

COMIC (COnversational Multimodal Interaction with Computers) (Foster 2004) uses models and results from cognitive psychology to make interaction with the system more intuitive. A demonstrator multimodal dialogue system, ViSoft, has been developed that helps customers to choose new designs for their bathroom. The system facilitates spoken, hand written and pen gesture input. A ‘talking head’ avatar provides system output combined with synthesised speech, deictic gestures and a simulated mouse pointer. The talking head avatar and a screenshot of the system are depicted in Figure 2.43. ViSoft first enables the user to specify the size/shape of their bathroom and position of doors/windows. The user then chooses the positioning of sanitary ware, before deciding on the bathroom tiles. When satisfied with the design of the bathroom, the user is finally given a 3D tour. A fission module, implemented in Java, chooses and coordinates output across the multimodal output channels.

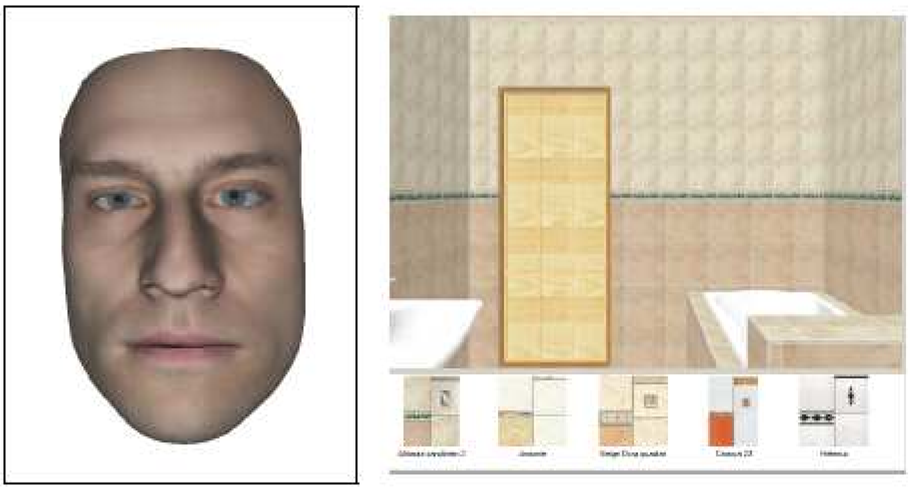
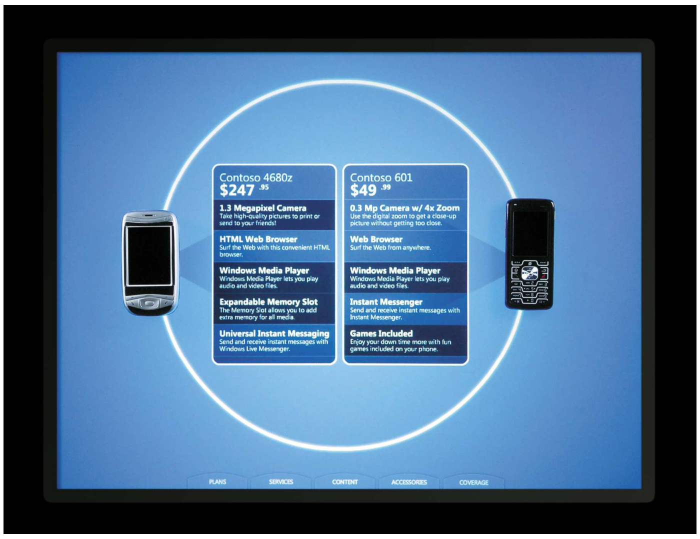


Figure 2.43: Avatar and screen shot of ViSoft (Foster 2004)

2.8.19. Microsoft Surface

Microsoft Surface (Microsoft 2009) is a recent attempt to revolutionise the way humans interface with computational devices. The physical interface to Surface is a 30 inch table-like display. Surface can recognise physical objects, such as mobile phones, that are placed on the surface and enables hands-on direct manipulation of digital content such as ring tones, images and maps.

Users can interact with Surface using touch, gesture, or by simply placing objects on it. Catering for both commercial and non-commercial end users, Surface has been tested in a number of application domains including mobile phone sales, e.g., choosing price plans and selecting ring tones, intelligent restaurant services, e.g., viewing menus and ordering food, paying bills, and digital photography, e.g., viewing, sharing and printing photos. Figure 2.44 illustrates the application of Surface in the sale of mobile phones.



Contoso 4680z
$247 .95

1.3 Megapixel Camera
Take high-quality pictures to print or send to your friends!

HTML Web Browser
Surf the Web with this convenient HTML browser.

Windows Media Player
Windows Media Player lets you play audio and video files.

Expandable Memory Slot
The Memory Slot allows you to add extra memory for all media.

Universal Instant Messaging
Send and receive instant messages with Windows Live Messenger.

Contoso 601
$49 .99

0.3 Mp Camera w/ 4x Zoom
Use the digital zoom to get a close-up picture without getting too close.

Web Browser
Surf the Web from anywhere.

Windows Media Player
Windows Media Player lets you play audio and video files.

Instant Messenger
Send and receive instant messages with Instant Messenger.

Games Included
Enjoy your down time more with fun games included on your phone.

Figure 2.44: Commercial application of Microsoft Surface (Microsoft 2009)

As shown in Figure 2.44, the customer can easily compare mobile phones and packages simply by placing the phones side by side on the surface. Figure 2.45 shows Microsoft Surface in a non-commercial setting – digital photography.

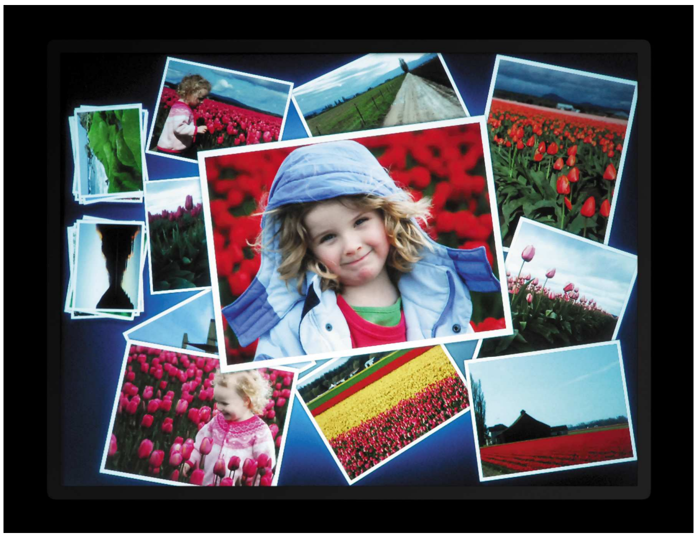


Figure 2.45: Digital photography in Microsoft Surface (Microsoft 2009)

As shown in Figure 2.45, users can slide and flick photos around in Surface just as they would in the physical world. In addition to direct interaction using touch and object recognition, the system allows multi-touch, i.e., recognition of multiple points of contact in parallel, and multi-user, e.g., collaboration between users to share photos, ring-tones.

2.8.20. Other multimodal systems

QuickSet (Johnston et al. 1997), a training simulator that enables voice and pen input, is a distributed system that consists of a collection of interacting agents that communicate using the Open Agent Architecture (Cheyer et al. 1998, OAA 2009). QuickSet supports direct manipulation by enabling complex pen gestures, such as arrows and various types of lines. The QuickSet interface enables users to manipulate an intelligent map with natural language and pen-based gestures. Semantic representation within QuickSet is in the form of typed feature structures (Carpenter 1992). MATCH (Multimodal Access To City Help) (Johnston et al. 2002) allows users to interact with a dynamic map with speech and pen input. Users can perform a variety of tasks such as circling an area of a map while asking for information about restaurants in that area. Maps are also the focus of attention in CUBRICON (Neal & Shapiro 1991) where users can point to objects on a map and ask questions such as, “Is this an air-base?”.

MATIS (Multimodal Airline Travel Information System) (Nigay & Coutaz 1995) enables users to access information about flight schedules with speech, keyboard, mouse and direct manipulation. The user can choose and freely switch between the various interaction modalities. The IHUB (Reithinger & Sonntag 2005) integration framework is intended for use in mobile multimodal dialogue systems that access the Semantic Web (Berners-Lee et al. 2001; SW 2009). IHUB, which constitutes a hub and spoke architecture, aims to allow users to perform real-time queries to the Semantic Web. The IHUB does not perform reasoning about message content, but simply validates the messages and routes them between the various modules of the system.

2.9. *Intelligent multimedia agents*

Authors use different terms to describe an intelligent multimedia agent that can engage with humans in a natural and intuitive way, using both verbal and non-verbal input/output communication, including, ‘intelligent agent’, ‘intelligent multimodal agent’, ‘conversational agent’, ‘Embodied Conservational Agent (ECA)’, ‘animated human simulation’, ‘animated presentation agent’, ‘interface agent’, ‘affective agent’ and ‘virtual human’. Whilst there are many different types of agents, e.g., talking heads, embodied, cartoon style, here the broad term, ‘agent’ shall refer to all such agents. There has been considerable research focusing on the development of agents that can engage in human-like conversations with real users. Central to the design of such agents is the means by which they accept and react to multimodal input, implement a strategy for turn-taking, and coordinate output across a range of modalities.

REA (Real Estate Agent) (Cassell et al. 2000) is an intelligent multimodal agent which acts as a salesperson in the real estate application domain. The real estate domain provides REA the opportunity to engage in both task- and socially-oriented dialogues. REA deploys computer vision techniques that enable it to understand the conversational intentions of a user. REA can respond using automated speech, facial expressions, and hand and body gestures. A user interacting with REA is depicted in Figure 2.46.

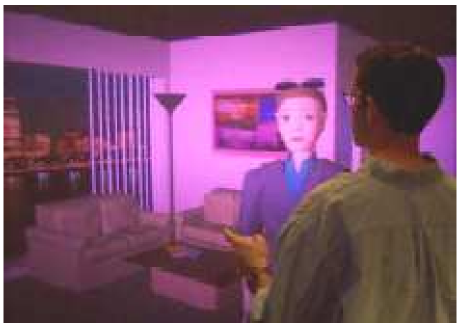


Figure 2.46: The REA agent (Cassell et al. 2000)

BEAT (Cassell et al. 2001), used for the implementation of REA, is an annotation tool which supplies input text to be spoken by an agent. Based on the same principle of Text-to-Speech (TTS) systems (Mc Tear 2004), which convert written text into speech, BEAT converts written text into verbal and non-verbal behaviours, e.g., hand gestures, head movements, facial expressions, eye-gaze. SAM (Cassell et al. 2000) acts as a peer playmate for children, telling stories and sharing experiences in a shared collaborative space. SAM can share physical objects across the real and physical world. Real-time video of the child’s play space is projected behind SAM so that he can appear to exist in the child’s actual environment. A screenshot of SAM is shown in Figure 2.47.

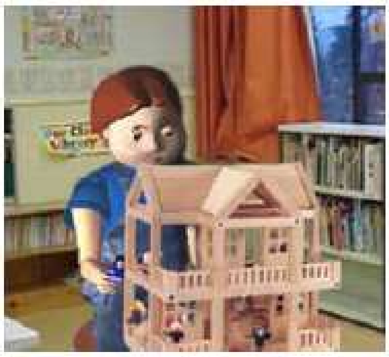


Figure 2.47: SAM (Cassell et al. 2000)

The Gandalf agent (Thórisson 1996, 1997), developed with the Ymir platform (Thórisson 1996, 1999), gives information about the planets of the solar system. The computer animated face and hand of Gandalf is shown in Figure 2.48.

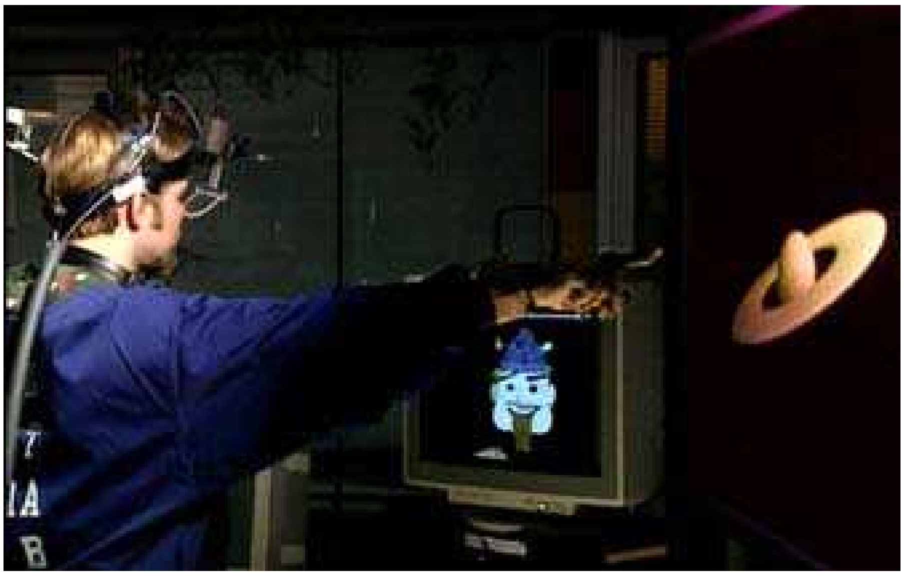


Figure 2.48: Gandalf (Thórisson 1996)

Gandalf can engage in natural, multimodal communication by coordinating output across speech, eye-gaze and gesture modalities. The agent can also sense the position of the user, determine what they are looking at, monitor their hand position, and perform automated speech recognition. A major focus in the design of Gandalf is its ability to handle turn-taking in an intelligent and human-like way.

de Rosis et al. (2003) focus on developing an expressive and believable agent, called Greta, which can communicate complex information using a combination of tightly synchronised verbal and non-verbal signals. Greta has been designed as an, ‘individual’, as opposed to a generic agent, to help encourage users to consider Greta a ‘friend’, not just an agent. Greta has been applied in the medical domain, giving information about treatments being proscribed by a doctor. Using facial expressions and behaviours, Greta is capable of performing many believable expressions as illustrated in Figure 2.49. Other agents include PPP Persona (André et al. 1996), a multipurpose agent that can present information retrieved from the Internet, Rapport (Gratch et al. 2007), which can build rapport with the user by providing non-verbal listening feedback, Steve (Rickel et al. 2001), an intelligent tutor that cohabits a number of virtual words with its student, and MAX (Multimodal Assembly eXpert) (Kopp & Wachsmuth 2004), an assembly expert who, in a virtual environment, assists users with complex assembly procedures.

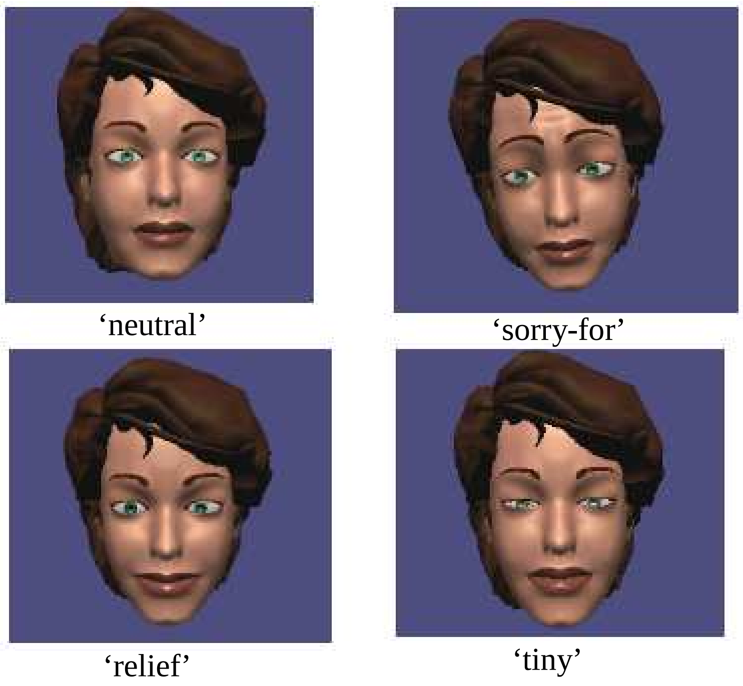


Figure 2.49: Greta’s expressions (de Rosis et al. 2003)

2.9.1. Turn-taking in intelligent multimedia agents

The problem of turn-taking is of huge importance in the design of agents. A well designed turn-taking strategy can greatly enhance the believability of an agent, and hence the naturalness of a dialogue with such an agent. By the same token, failure to deal adequately with the issue of turn-taking can have drastic effects on the naturalness of the human-agent interaction. Turn-taking is a key consideration in the design of the Gandalf agent (Thórisson 1996). Gandalf signals his intention to take a turn by using a common behaviour pattern in humans, i.e., moving eyebrows up and down quickly and glancing to the side and back. The Ymir Turn-taking Model (YTTM) (Thórisson 2002) is one of the most comprehensive computational models of multimodal turn-taking. YTTM addresses the full perception-action loop that is necessary for real-time turn-taking including multimodal perception, knowledge representation, decision-making and action generation. Turn-taking is an important task involved with dialogue management, and much research is devoted to managing turn-taking as part of a broader dialogue management strategy (López-Cózar Delgado & Araki 2005; Mc Tear 2004). Turn-taking signals are typically composed of a combination of multimodal events, such as head and gaze direction, hand position, and speech intonation. Considerable advances in the area of dialogue management have seen the granularity of turn-taking increase. Next-generation spoken dialogue systems are likely to abandon today’s principles of turn-taking completely, since there will not be clearly defined transition points where a user will stop and wait for a system response. The challenge in turn-taking is the recognition that the user wishes to give or take a turn. As dialogue systems become increasingly multimodal the dialogue management strategy need to become more competent recognising turn-taking cues, e.g., speech, eye-gaze, head movement/direction, facial expression, simultaneously across multiple modalities. This requires that dialogue management systems are capable of complex, flexible decision-making over some or all of the relevant input modalities.

Table 2.2 gives a summary of the multimodal systems discussed in this chapter. The systems are categorised into three groups: (1) multimodal platforms, (2) multimodal systems and (3) intelligent multimedia agents. A ✓ symbol is used to indicate a capability of the system, whilst a ✗ symbol indicates little or no capability. Table 2.2 shows that all multimodal platforms are considered to offer full capability in each of the categories since they provide a framework for implementing a range of different multimodal systems. Note that *F* in the semantic representation column represents frames, whilst *BB* and *D* in the semantic storage column represent blackboard-based and distributed.

2.10. *Multimodal corpora and annotation tools*

In order to develop intelligent multimodal systems it is necessary to semantically annotate multimodal input/output data that can be used to test such systems. Typically, multimodal data is collected from staged or naturally occurring situations, e.g., meetings, talk shows, Wizard-of-Oz experiments. Multimodal corpora aid study of the characteristics of human-human communication and the development of more natural human-computer interaction. Rules may also be derived from such corpora that can then be applied to decision-making within multimodal systems. The corpora can be annotated with varying levels of granularity, depending on their intended use. To assist the process of annotation various software tools have been developed. The Anvil tool (Martin & Kipp 2002) is widely used to annotate multimodal data from various application domains but, despite the existence of tools such as Anvil, annotation of multimodal data remains a difficult task.

The AMI corpus (Carletta et al. 2006), collected during staged and naturally occurring meetings, is one of the largest of its kind and is freely available for academic purposes. In Petukhova (2005), the author focuses on the detailed annotation of dialogue acts in the AMI corpus. The SACTI-2 (Simulated ASR Channel, Tourist Information) corpus (Weilhammer et al. 2005) contains annotations relating to speech and mouse click input. The data was collected during task-oriented human-human dialogues with speech and an interactive map. The annotations were performed with the Anvil tool (Martin & Kipp 2002).

Categories	System	Year	Semantic representation		Semantic Storage		Multimodal Interaction								
							Input Media				Output Media				
											Text	Audio		Visual	
			F	XML	BB	D	Text	Pointing (haptic deixis)	Speech	Vision		Speech	Non-speech audio	Graphics (static)	Video or animation
Multimodal Platforms	Collagen	1996	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ymir	1997	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	Chameleon	1998	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	Oxygen	1999	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	EMBASSI	2001		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	DARPA Galaxy Communicator	2001	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	DARBS	2001	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	JASPIS	2000		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	Psyclone	2003		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
Multimodal Systems	Waxholm	1992	✓			✓	✗	✓	✓	✗	✗	✓	✗	✓	✗
	Spoken Image/ SONAS	1994	✓		✓		✗	✗	✓	✗	✗	✓	✗	✓	✗
	Aesopworld	1996	✓			✓	✗	✗	✓	✓	✗	✓	✗	✓	✗
	InterACT	1996	✓			✓	✗	✓	✓	✓	✗	✓	✗	✓	✗
	SmartKom	2000		✓		✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
	MIAMM	2001		✓		✓	✗	✓	✓	✗	✓	✓	✗	✓	✗
	XWand	2003		✓		✓	✗	✓	✓	✗	✗	✗	✗	✗	✗
	COMIC/ViSoft	2003		✓		✓	✗	✓	✓	✓	✗	✓	✗	✓	✓
	CONFUCIUS	2003	✓			✓	✗	✗	✗	✗	✗	✓	✓	✓	✓
	TeleMorph/ TeleTuras	2004		✓		✓	✓	✓	✓	✗	✓	✗	✗	✓	✓
Intelligent Multimedia Agents	Gandalf	1997	✓		✓		✗	✗	✓	✓	✗	✓	✗	✗	✓
	SAM & REA (BEAT)	1999		✓		✓	✗	✗	✓	✓	✗	✓	✗	✗	✓
	Greta	2000		✓		✓	✗	✗	✓	✓	✗	✓	✗	✗	✓

Table 2.2: Summary of multimodal systems

The MUMIN multimodal annotation scheme (Allwood et al. 2007) caters for the annotation of gestures and facial expressions and focuses on their use in feedback, turn-taking and sequencing communicative functions. Other work is concerned with learning from multimodal corpora in order to automatically generate the multimodal behaviours of an agent (Kipp 2006). Available sources of multimodal corpora are discussed further in Rehm and André (2006) and López-Cózar Delgado & Araki (2005), whilst Carletta et al. (2006) give a comprehensive review of existing annotation schemes.

2.11. *Dialogue act recognition*

In a multimodal dialogue system, a dialogue act (Mc Tear 2004) is a functional tag which represents the communicative intention of a user's utterance or gesture. Determining the communicative intentions behind an utterance is considered an important first step in dialogue management (Webb et al. 2005). Considerable research concerns itself with the recognition of user intentions. Grosz and Sidner (1986) consider in detail the recognition of the intentions of a user. They consider the three components of discourse structure to be linguistic, intentional and attentional, where 'attentional' refers to the focus of attention of the user as a dialogue unfolds. Bunt and Keizer (2006) propose a multi-agent approach to multidimensional dialogue management, where dialogue act agents are designed to focus on tasks, feedback and social obligation management. A dialogue manager, called PARADIME, has been implemented to test the approach in a question-answering system which provides information in the medical application domain. The PARADIME architecture implements a context model, primarily concerned with linguistic, semantic, cognitive, and social context, though physical and perceptual context is also considered. The various dialogue act agents constantly monitor the context model and are automatically triggered by certain conditions.

The recognition of dialogue acts, before the advent of multimodal systems, concerned only the analysis of words spoken by the participants in a dialogue. Dialogue act recognition for language alone is by no means a simple task, but clearly it becomes more complicated when one must consider multiple modalities such as eye-gaze, gesture and facial expression, in addition to speech. Of course, multimodality can do more than make dialogue act recognition more complicated – it can make it more accurate and, in instances where serious ambiguity occurs in one modality, the use of information from other modalities can facilitate multimodal decision-making. In order that dialogue act recognition can be effectively achieved over multiple modalities there is a requirement for more advanced and innovative approaches to decision-making.

2.12. *Anaphora resolution*

Reference resolution is a key problem in the design of multimodal systems. Two types of references that regularly occur in day-to-day speech are anaphoric and deictic. Here we consider the term ‘anaphoric’ to broadly include references to preceding utterances, forward references (cataphora), and reference to objects in the physical world (exophora). Anaphoric expressions can frequently occur in a user’s interaction with a multimodal system. For example, the user could refer to the subject of a preceding utterance, e.g., “where is her office?”, or to an object in the user’s environment, e.g., “the table”. Deictic references often co-occur with pointing, e.g., when a user refers to a position on a map displayed on the screen, e.g., “is this a library?”. They can also refer to some time in the future, e.g., “I will give you the document then”. Resolving such references requires that the system has an understanding of current dialogue (contextual information), a record or past dialogue and an understanding of the current domain (domain model).

Bolt (1980) developed one of the first multimodal interfaces. His ‘Put That There’ system was one of the earliest attempts to address the issue of anaphora resolution in a multimodal system. The ‘Put That There’ interface allowed users to add, delete, and move graphical objects around a wall projection panel using speech and gesture input. In Brøndsted (1999) reference problems in the Chameleon platform (Brøndsted et al. 1998, 2001) are discussed. Brøndsted (1999) considers three linguistic reference types; (1) Endophora – covering both anaphora and cataphora, (2) deixis – depends on extralinguistic context, i.e. interpretation of the reference relies on the circumstance of the utterance, (3) cross-media (deictic) – reference to an antecedent in another communication channel, and (4) cross-user/system - reference in the user input/system output to an antecedent in the system output/user input. Brøndsted (1999) emphasises the point that, in order to deal effectively with such decisions, a system must understand not just user input but also its own output. Pineda and Garza (1997) discuss a model for multimodal resolution where the focus is on establishing the referent of an expression in one modality using contextual information from another modality. André and Rist (1994) developed a model for referring to objects using text and pictures, which is demonstrated in a multimodal presentation system (Stock & Zancanaro 2005). The model outlined in André and Rist (1994) was implemented in the WIP multimodal presentation system (Wahlster et al. 1992).

With all approaches concerned with reference resolution it is important that the system understands the current domain, e.g., to resolve queries such as, “whose office is this [→]?”, and, “how do I get from his office to that office [→]?”. It is equally important that the system understands the meaning of not just the current utterance, and/or gesture, facial expressions, but also previous dialogue acts, so that queries of the form, “what is his surname?”, and, “where is her office?”, can be more easily resolved, i.e., the system needs to maintain a dialogue history. A common means of maintaining a dialogue history is the use of a blackboard (Thórisson et al. 2005), as discussed in Section 2.3. The storage of semantics over time enables multimodal systems to retrace dialogue history in order to address the problem of reference resolution.

2.13. *Limitations of current research*

The work discussed in this chapter has considerably advanced the capabilities of multimodal systems, enabling them to engage with real users in intelligent natural and human-like interaction. Such rich interaction would only have been imaginable when Richard Bolt took the first tentative steps towards a multimodal interface with his ‘Put That There’ system in the early eighties (Bolt 1980, 1987). Yet, if progress over the next thirty years matches the speed of the previous three decades then we can expect the systems of today to quickly appear primitive. Of course, such an opinion may be optimistic since the last thirty years have seen considerable advancements in the processing power of computers, which has enabled the development of more intelligent systems. How much intelligent systems can advance over the next thirty years is dependent upon how much we can be assured that hardware technology will continue to advance at a similar rate to before, i.e., will Moore’s Law (Brock 2006) continue to remain true? Whilst this question will continue to evoke debate within the research community, one thing is certain, there will always be hardware constraints that impose a glass ceiling on the development of intelligent systems. It is therefore important that AI researchers focus on (1) removing or reducing these constraints or (2) developing more efficient and innovative intelligent systems that can operate within the current hardware limitations. What AI researchers certainly must not be focused on, or rather be distracted by, is developing software and hardware technology that already exists. In other words, they need to avoid ‘reinventing the wheel’, since this practice distracts them from other work that can add real value by advancing the capabilities of multimodal systems. This is a view shared by Thórisson (2007, p. 13) who states that, “instead of trying to build directly on systems already implemented, researchers do one of two things: they either re-implement (some of the) functionality of the former student’s software from scratch or they choose to do their research in isolation from the functionality and context that that software would have provided…the result is a state where researchers either constantly reinvent the wheel or produce their work in increased isolation.”

Much of the work discussed in this chapter could, to a certain extent, be considered to re-implement existing technology. For example, one could find similarities between the hub of Chameleon (Brøndsted et al. 1998, 2001) and the hub implemented in Galaxy Communicator

(Bayer et al. 2001), but both platforms were developed independently of each other using different tools to implement distributed processing. DARBS (Distributed Algorithmic and Rule-Based System) (Choy et al. 2004a, 2004b; Nolle et al. 2001) and Chameleon both implement a centralised blackboard but both blackboards have been developed from scratch to operate within their respective systems. Similarly, the functionality of the blackboards in Ymir is similar to that of the multiple blackboards in SmartKom, yet both platforms used different tools to develop their blackboards. Few of the systems discussed here, with the notable exceptions of some, including Psyclone (Thórisson et al. 2005) and OpenAir (Mindmakers 2009; Thórisson et al. 2005), have made their work freely available to others in the community so that it may form the basis of future research. Should this trend prevail, the constant reimplementation discussed in Thórisson (2007, p. 14) will continue indefinitely and the advancement of multimodal systems will continue to be stifled by the practice of isolated researchers constantly reinventing the wheel. Two points are key to ensuring this trend is not continued: (1) reversal of the culture within many academic institutions that forces researchers and Ph.D. students to work in isolation and subsequently focus to a large extend on developing technology that already exists, and (2) researchers focus more on building on existing tools and technology to advance towards more intelligent multimodal systems. It is the change in culture that poses the greatest challenge and overcoming this would go some way to ensuring that researchers focus more of their effort on exciting new research and less on the replication of existing work.

Further evidence of a lack of synergy in AI research can be found in the area of multimodal corpora and annotation tools. As discussed in Section 2.10, there is an abundance of existing multimodal corpora (Carletta et al. 2006; Petukhova 2005; Weilhammer et al. 2005; Kipp 2006; Rehm & André 2006) and annotation tools (Martin & Kipp 2002). However, many of these corpora are developed for the very specific needs of different applications. The corpora are therefore often difficult to utilise outside of that particular application domain. One possible way to maximise the use of existing multimodal data is to increase collaboration between researchers and academic institutions on shared projects and, in turn, build synergy within the AI research community and reduce the replication of work and academic effort. Another option is to further standardise the annotation schemes for multimodal corpora, thus ensuring that future corpora will be of maximum benefit to the research community as a whole and not just of use in one system or a small subset of systems. The latter is obviously a huge challenge since, as multimodal systems constantly advance, so too do their requirements in respect of semantic representation.

In summary, there are two key limitations of current research: (1) replication of work due to lack of collaboration between researchers who subsequently often work in isolation and reinvent the wheel and (2) lack of standardised multimodal corpora that could be applied across multiple application domains. Overall, it follows that there is a general lack of synergy in AI research. So how can this be addressed in the short term? First, it is important to utilise existing tools and technology wherever possible. For example, Psyclone (Thórisson et al. 2005) facilitates distributed processing and is free to use for academic purposes. Likewise, other tools exist for semantic representation, communication and decision-making and all these should be explored in detail before deciding to re-implement such technology. By utilising existing technology researchers can focus on more novel aspects of their work and subsequently add more value to their academic endeavours. Second, systems should be constructed in a modular way using standard tools for programming and representation that maximise their potential use by others in the field. For example, it would be better to use an XML-based approach to semantic representation than to develop a representation framework bespoke to a particular application domain. Finally, more researchers should focus on building systems or components that can form the basis of future research in the area of multimodal systems.

2.14. *Summary*

This chapter has reviewed a number of areas key to the field of multimodal systems. First, the problem of multimodal semantic fusion and synchronisation was discussed. A review of multimodal semantic representation techniques was then presented. Next, communication in multimodal systems was considered, before a discussion on four AI methods for decision-making in multimodal systems: fuzzy logic, genetic algorithms, neural networks and Bayesian networks. Distributed processing was considered and a summary of available tools for distributed processing was given. We then focused on existing multimodal systems and platforms, with discussion primarily around their individual approach to semantic storage and representation, communication and decision-making. Some intelligent multimedia agents were presented to give a flavour of progress in this area, before a discussion on the pertinent issue of turn-taking in such agents. Existing multimodal corpora and annotation tools were then reviewed, before consideration of the important challenges of dialogue act recognition and anaphora resolution, which are key to this thesis. The chapter concluded with a discussion on the limitations of current research.

Chapter 3 Bayesian Networks

This chapter provides a background discussion of Bayesian networks and their application to decision-making. First, a definition and discussion of the history of Bayesian networks is provided. The structure of Bayesian networks is then given and their ability to perform intercausal reasoning is considered. An example Bayesian network is presented before consideration of influence diagrams, which are Bayesian networks extended to include utility and decision nodes. Key problems in constructing Bayesian networks are highlighted followed by a discussion of their advantages over other approaches to decision-making. Next, the limitations of Bayesian networks are considered. Applications of Bayesian networks are then addressed and their deployment to date in multimodal systems presented. The chapter concludes with an evaluation of existing software and tools for implementing Bayesian networks.

3.1. *Definition and brief history*

Bayesian networks (Pearl 1988; Charniak 1991; Jensen 1996, 2000; Kjærulff & Madsen 2006; Jensen & Nielsen 2007; Pourret et al. 2008) are an AI technique for probabilistic reasoning under conditions of uncertainty. As observed by Charniak (1991, p. 62), Bayesian networks provide, "a convenient way to attack a multitude of problems in which one wants to come to conclusions that are not warranted logically but, rather, probabilistically". Although they have come to greater prominence in the last two decades, the origins of Bayesian networks are several centuries old. The term 'Bayesian' is derived from the surname of Thomas Bayes who, in 1763, presented his ratio formula for computing conditional probabilities (Bayes, 1763):

$$P(A|C) = \frac{P(A,C)}{P(C)} \tag{3.1}$$

In deriving this equation, Thomas Bayes gave scientific notation to the phrase, "…given that what I know is C" (Pearl 1988, p. 17). *C* in Equation 3.1 represents the context of the belief in *A*. The notation *P(A|C)* is referred to as Bayes conditionalisation and represents the probability *P* of an event *A* occurring given the knowledge or evidence *C*. Equation 3.1 was developed to form Bayes' Theorem:

$$P(A \mid C) = \frac{P(C \mid A)P(A)}{P(C)} \qquad (3.2)$$

Bayes' Theorem enables the belief of hypothesis *A* to be updated in light of new evidence *C*. Bayesian networks provide a compact graphical means of implementing Bayes' Theorem. *P(A|C)* can also be read as, "the probability of *A* being true in the context of evidence *C* being observed".

More generally, network representations have been used extensively in AI reasoning systems to encode relevancies between variables and facts, e.g., pointers, frames, inheritance hierarchies (Pearl 1988, p. 13). A major reason for their use is their ability to represent causal relations (Pearl 2000). The notion of causality has intrigued mankind for centuries. In a public lecture delivered in 1996 entitled, "The Art and Science of Cause and Effect", Judea Pearl presents a history of causality dating back to the earliest days of human development. He observes that even Adam and Eve in the Garden of Eden were well versed on causality when they gave explanations of what *caused* them to eat fruit from the tree (Pearl 2000, p. 332). Pearl illustrates the wide scope of causality when he says that, "whether you are evaluating the impact of bilingual education programs or running an experiment on how mice distinguish food from danger or speculating about why Julius Caesar crossed the Rubicon or diagnosing a patient or predicting who will win the presidential election, you are dealing with a tangled web of cause-effect considerations" (Pearl 2000, p. 331). Causation is a concept that can easily be understood by humans. As discussed in Pearl (2000, p. 1), humans often use causal utterances in situations where there exists uncertainty. For example, we say, "tonight's football match will be cancelled if that rain keeps up", or, "if we can score another point, we will win the match", when we are entirely uncertain that the game will be cancelled if it keeps raining, or that another point will mean our team will win the match. Such causal statements are an everyday occurrence in human speech and are typically used under conditions of uncertainty. It is therefore intuitive for humans to consider causes and effects, and to build Bayesian networks to represent the causal relationships between variables and events. Three other relationships: likelihood, conditioning and relevance, are considered, with causation, as the basic primitives of the language of probability (Pearl 1988). These, too, are relationships which humans can easily comprehend and can be effectively represented using the cause-effect structure of Bayesian networks. Bayesian networks are therefore an intuitive means of modelling human-like decision-making and provide a more developer-friendly method of implementing the power of probabilistic reasoning under uncertainty.

3.2. *Structure of Bayesian networks*

The Bayesian network itself consists of chance nodes (random variables, uncertain quantities) and directed edges (arcs, arrows, links) between the nodes. The nodes represent variables in the domain and directed edges represent influences between nodes in the network. The graph, consisting of the nodes and edges, represents the qualitative part of the Bayesian network. A Conditional Probability Table (CPT) specifies the quantitative part of the network. The conditional probabilities are updated dynamically when new information is input to the network, i.e., new observations or evidence. The term *prior probability* is used before any evidence is considered, whilst the term *posterior probability* applies after evidence has been added. Bayesian networks are a compact means of explicitly representing relationships, e.g., causation, dependence and independence, between variables of a domain. As observed by Pfeffer (2000, p. 27), "the graphical structure of the network reflects the causal structure of the domain". A drawback of probability theory per se is the vast amount of numbers that need to be considered in order to reach a conclusion, i.e., for *n* binary variables, we have 2^n-*1* joint probabilities (Charniak 1991). The structure of Bayesian networks enables us to encode knowledge in such a way that important, and ignorable, information is easily recognisable (Pearl 1988, p. 12). As Charniak (1991) alludes to, the conclusions of a Bayesian network may be reached with minimal computation due to the compact nature of the networks. Because graphs are easy to understand they provide, "an excellent language for communicating and discussing dependence and independence relations among problem-domain variables" (Kjærulff & Madsen 2006, p. 3). The efficiency of a Bayesian network is due to the built-in independence assumptions about variables of the problem domain. Thus, the secret to developing efficient Bayesian networks is our understanding of the (conditional) dependence and independence relationships between the variables of the domain, or nodes in our network. Bayesian networks allow us to represent these dependence and independence relations using directed links from causes to effects.

3.3. *Intercausal inference*

Bayesian networks have the intrinsic ability to perform deductive, abductive and intercausal reasoning (Kjærulff & Madsen 2006). Deductive (causal) reasoning considers the direction of causal links, edges, arcs or arrows between variables of the network, i.e., from cause to effect. For example, observing a cause increases our belief about a possible effect. Abductive reasoning is the opposite of deductive reasoning, i.e., from effect to cause, where observing an effect increases our belief about a possible cause. Hence reasoning follows, and goes against, the direction of causal links. Intercausal reasoning, sometimes referred to as intercausal inference or the explaining away effect, is a powerful property of graphical models, i.e., Bayesian networks. Intercausal inference occurs when evidence on one possible cause disconfirms, or explains away, another possible cause. For example, suppose there are two possible causes for a person not arriving at work: (1) the employee is ill, and (2) the employee has been delayed by road works. If we now get evidence that the employee was ill yesterday and that there are no road works today, then it becomes more likely that illness is the reason for the employee's non-attendance, and less likely that road works are responsible. Hence, road works as a cause for the employee not arriving at work has been explained away. Intercausal reasoning is an inherent property of Bayesian networks, whilst its implementation in a rule-based system would require the specification of numerous and complex rules.

3.4. *An example Bayesian network*

To further illustrate Bayesian networks we will consider an example network given in Pfeffer (2000). The simple Bayesian network discussed in Pfeffer (2000), shown in Figure 3.1, can be used to predict the performance of a student on a course.

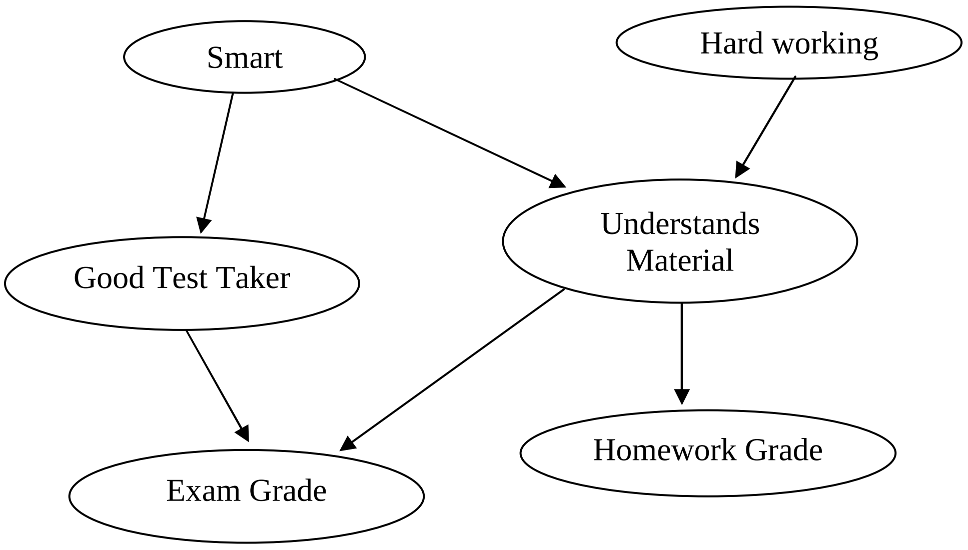


Figure 3.1: Example Bayesian network (Pfeffer 2000)

Six nodes are used in the Bayesian network in Figure 3.1: *Smart*, *Hard working*, *Good Test Taker*, *Understands Material*, *Exam Grade* and *Homework Grade*. The nodes *Smart*, *Hard Working*, *Good Test Taker* and *Understands Material* all take Boolean values of *True* or *False.* Both *Exam Grade* and *Homework Grade* have a set of grades {A, B, C, D, F} as their type. The node *Smart* is a parent of *Good Test Taker* to reflect the fact that being a good test taker depends on smartness. Causal relations within the domain are further indicated by the arrows from *Smart* and *Hard Working* nodes to the *Understands Material* node, i.e., it is believed that smartness and the fact that a student is hardworking have influence over the student's understanding of the learning material. The student's understanding of the material has an influence over both the homework grade and the exam grade. *Good Test Taker* is also a parent of *Exam Grade* to model the fact that some students may perform better in exams than others. The conditional probability functions of this example network are shown in the tables in Figure 3.2.

Smart	
True	*False*
0.5	0.5

Hard Working	
True	*False*
0.5	0.5

Smart	**Good Test Taker**	
	True	*False*
True	0.75	0.25
False	0.25	0.75

Smart	**Hard Working**	**Understands Material**	
		True	*False*
True	*True*	0.95	0.05
True	*False*	0.6	0.4
False	*True*	0.6	0.4
False	*False*	0.2	0.8

Good Test Taker	**Understands Material**	**Exam Grade**				
		A	*B*	*C*	*D*	*F*
True	*True*	0.7	0.25	0.03	0.01	0.01
True	*False*	0.3	0.4	0.2	0.05	0.05
False	*True*	0.4	0.3	0.2	0.08	0.02
False	*False*	0.05	0.2	0.3	0.3	0.15

Understands Material	**Homework Grade**				
	A	*B*	*C*	*D*	*F*
True	0.7	0.25	0.03	0.01	0.01
False	0.2	0.3	0.4	0.05	0.05

Figure 3.2: Conditional Probability Tables for student grades example (Pfeffer 2000)

The tables in Figure 3.2 represent conditional probability functions of the domain. For example, the function CPF_{HG}, Conditional Probability Function for *Homework Grade* node, specifies that, if the student understands the learning material, he/she will get a grade A with a probability of 0.7, but if the student does not understand the material then the probability that the student will get a grade A is reduced to 0.2. Note that, for each of the tables in Figure 3.2, the sum of every row is equal to unity.

3.5. *Influence diagrams*

Influence diagrams are essentially Bayesian networks extended to include decision and utility nodes, in addition to the standard chance nodes. A decision node is represented as a square and contains states that describe the choices available to the decision-maker. The utility (value) node is shown as a diamond and represents the expected utility or value of a particular decision. An influence diagram may contain multiple decision nodes, as is the case in the oil wildcatter example discussed in Hugin (2009). The influence diagram for this example is depicted in Figure 3.3.

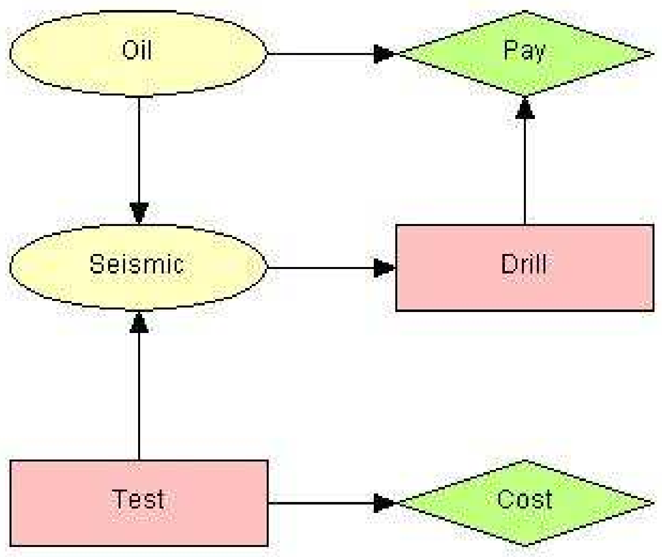


Figure 3.3: Example influence diagram (Hugin 2009)

There are two decisions in this example: whether or not to test for oil using seismic soundings and whether or not to drill for oil. The cost of testing is $10,000, whilst the cost of drilling for oil is $7,000. Note that, in the influence diagram in Figure 3.3, the arrow from *Seismic* to *Drill* does not represent a causal relation. This is because a decision node does not have a conditional probability table assigned to it. The arrow from *Seismic* to *Drill* does however indicate that, when the decision must be made on whether or not to drill, the state of *Seismic* is known. The chance node *Oil* has three states: “dry”, “wet” and “soak”. The chance node *Seismic* also has three states: “closed” (closed reflection pattern – suggesting much oil is present), “open” (open pattern – suggesting that some oil is present) and “diff” (diffuse pattern – highly unlikely that there will be any oil). The *Test* node of the influence diagram in Figure 3.3 has two states (or actions): “test” and “not”. Tables 3.1 and 3.2 show the utility tables for the *Pay* and *Cost* utility nodes respectively. Similarly, the *Drill* decision node has “drill” and “not” as its actions.

Drill = “drill”			Drill = “not”		
Oil = “dry”	Oil = “wet”	Oil = “soak”	Oil = “dry”	Oil = “wet”	Oil = “soak”
-70	50	200	0	0	0

U(Pay)

Table 3.1: Utility tables for *Drill* decision node (Hugin 2009)

Test = “test”	Test = “not”
-10	0

U(Cost)

Table 3.2: Utility tables for *Test* decision node (Hugin 2009)

Tables 3.1 and 3.2 present the potential financial gains and costs associated with the decisions to test and drill for oil. As shown, deciding to drill for oil when *Oil* = “dry” would cost $70,000, whilst drilling when *Oil* = “soak” would yield a profit of $200,000. Of course, the influence diagram takes into consideration the probabilities of the *Oil* and *Seismic* variables, i.e. *P(Oil)* and *P(Seismic | Oil, Test)*. Running the network in Figure 3.3 using the Hugin decision engine processes these probabilities, along with the potential value and cost, before recommending whether or not it is advisable to drill for oil.

3.6. *Challenges in constructing Bayesian networks*

Bayesian networks can be constructed either manually, (semi-) automatically from data, or through a combination of both approaches. Whilst it is relatively easy to quickly construct a Bayesian network for a given problem domain, ensuring that the network *correctly* represents causal dependence and independence relations within the domain can be a difficult and time-expensive task. The process of constructing a Bayesian network frequently involves several iterations of the design, implementation, analysis and testing phases. The iterative process of constructing a Bayesian network is illustrated in Figure 3.4. The design phase of construction requires the identification of variables of the problem domain, defining the relationship between the variables and performing verification of the Bayesian model, i.e., the qualitative component. This requires a detailed knowledge of the problem domain and can often require close collaboration with problem domain experts. The implementation phase involves eliciting the parameter values, i.e., the quantitative component. Eliciting the values, i.e., completing the conditional probability tables, is often a labour intensive task. It is important, therefore, that one is satisfied that the qualitative component, i.e., the graphical structure, is correct before proceeding to the quantitative part. The third phase in the Bayesian model construction process is running test cases with known outcomes. The final stage is to analyse the Bayesian network to confirm correctness. As illustrated in Figure 3.4, these four phases are iterated until the designer is satisfied that the network is correct.

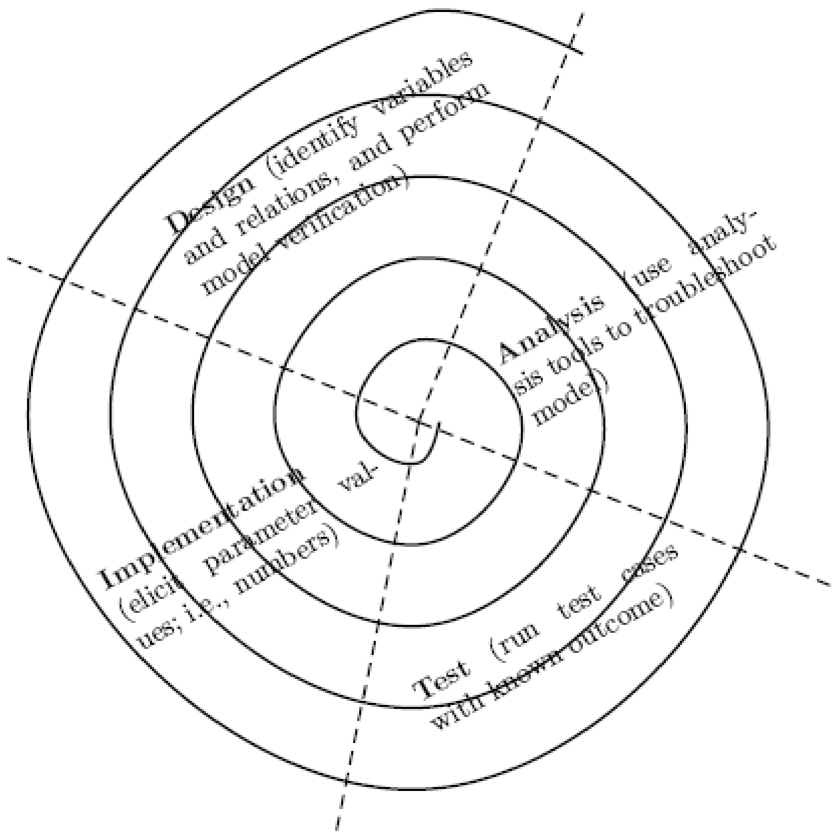

Design (identify variables and relations, and perform model verification)
Analysis (use analysis tools to troubleshoot model)
Implementation (elicit parameter values; i.e., numbers)
Test (run test cases with known outcome)

Figure 3.4: Iterative process of Bayesian network construction (Kjærulff & Madsen 2006)

Correctly identifying the variables of a given problem domain can sometimes be difficult. It is important to focus on two aspects: (1) the problem to be solved and (2) the information required to solve it. Information not related to the problem and its solution should not be captured in the Bayesian network. Kjærulff & Madsen (2006, p. 132) point out that, "defining variables corresponding to the (physical) objects of a problem domain is a common mistake made by most novices. Instead of focusing on objects of the problem domain, one needs to focus on the problem (e.g. possible diagnoses, classifications, predictions, decisions to be made) and the relevant pieces of information for solving the problem".

Another major challenge in the construction of Bayesian networks is the correct modelling of causality. Whilst the notion of causality is easily understood by humans, care is needed to ensure that the causes and effects in a problem domain are correctly identified. As discussed in Kjærulff & Madsen (2006, p. 16), "it is a common modeling mistake to let arrows point from effect to cause, leading to faulty statements of (conditional) dependence and independence and, consequently, faulty inference”. As an example, consider the Bayesian network shown in Figure 3.5. At a glance, the network in Figure 3.5 may look correct, i.e., if a person is laughing and/or smiling then the person is happy. However, the directed links from the *Laughing* and *Smiling* nodes suggest that the fact that a person is laughing or smiling causes that person to be happy. This is obviously incorrect, it is the fact that the person is happy that causes the person to laugh or smile. Reversing the links of the network in Figure 3.5 gives correct modelling of causality in this problem domain as shown in Figure 3.6. This very simple example illustrates how mistakes can easily be made when modelling causal relations in a problem domain. Careful consideration therefore needs to be given to the causes and effects before attempting to construct a Bayesian network to model a particular problem domain.

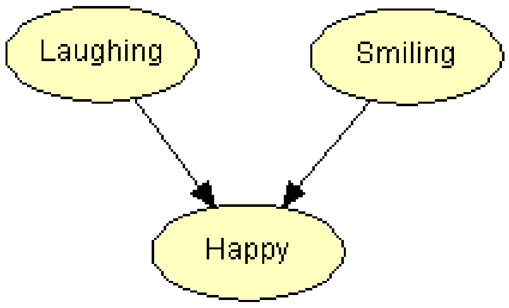


Figure 3.5: Incorrect modelling of causality

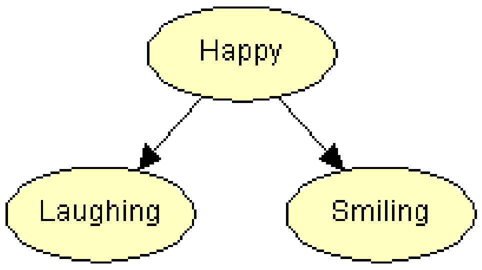


Figure 3.6: Correct modelling of causality

It should be noted that it is not essential that the links of a Bayesian network follow a causal interpretation (Kjærulff & Madsen 2006, p. 11), but doing so makes model construction much more intuitive. Implementation of the causal relations using a graphical model also helps ensure correct representation of the dependence and independence relationships existing between variables of the problem domain. The graphical structure encodes these relationships and a very compact representation of the dependence and independence relations amongst problem-domain variables is obtained. A more in-depth discussion on the subject of causal relations may be found in Pearl (2000).

3.7. *Advantages of Bayesian networks*

Bayesian networks are increasingly becoming the paradigm of choice for reasoning under uncertainty. Their increased popularity is due to a number of factors. As discussed in Kjærulff & Madsen (2006, p. v), “the graphical-based language for probabilistic networks is a powerful tool for expressing causal interactions while in the same time expressing dependence and independence relations among entities of a problem domain”. The notion of causality sits easily with the human mind and we can therefore very quickly construct useful networks that are capable of human-like reasoning under conditions of uncertainty. Representing exactly the causal dependence and independence relations within a problem domain may prove difficult. However, as Kjærulff & Madsen (2006, p. 136) observe, “it is important to bear in mind that all models are wrong, but that some might be useful”.

Due to Bayesian networks’ ability to handle causal independence, inference can be efficiently performed on models containing a very large numbers of variables. The inference that is performed by Bayesian networks is based on a deep-rooted theoretical foundation that is centuries old. Bayesian networks have a major advantage over rule-based systems, in that they can perform deductive, abductive and intercausal reasoning; the latter, according to Kjærulff & Madsen (2006, p. 4), being the property that sets Bayesian networks apart from other reasoning paradigms. It is also advantageous that many efficient algorithms exist for learning and adapting Bayesian networks from data. According to Zou and Bhanu (2005, p. 7), a Bayesian network is, “an attractive framework for statistical modeling, as it combines an intuitive graphical representation with efficient algorithms for inference and learning.” The marriage of a compact intuitive graphical representation with a powerful reasoning mechanism make Bayesian networks a popular choice for a plethora of applications.

Bayesian networks offer a lot of flexibility to the developer in modelling a problem domain. As observed by Kjærulff & Madsen (2006, p. vi), “probabilistic networks are “white boxes” in the sense that the model components (variables, links, probability and utility parameters) are open to interpretation, which makes it possible to perform a whole range of different analyses of the networks (e.g., conflict analysis, explanation analysis, sensitivity analysis, and value of information analysis)”. A key benefit of Bayesian networks, as observed by Zou and Bhanu (2005), is their ability to be extended in order to handle time series data, e.g., dynamic Bayesian networks. Another advantage of Bayesian networks over other approaches to decision-making is their capability to handle missing data. When a Bayesian network has been constructed it will always run with or without data, or evidence, being added. As new information becomes available it can be added on the fly to the network and the accuracy of the conclusion reached by the network can be improved. The compact graphical nature of Bayesian networks also renders them useful in communicating ideas about a problem domain amongst different members of a development team, e.g., knowledge engineers and problem domain experts.

3.8. *Limitations of Bayesian networks*

Whilst Bayesian networks are suitable for, and have been successfully applied in, a wide variety of application domains there are problem domains where they are not ideal and where other modelling paradigms may be a better choice. A problem domain where it is particularly difficult to define the variables would be less suitable for Bayesian networks. For example, as discussed in Kjærulff & Madsen (2006, p. 119), in the medical application domain the set of possible symptoms for a particular illness are normally well-defined, e.g., sore throat, headache, fever, but the variables relevant in modelling a person's like or dislike for a painting are likely to be harder to define. Problem domains where it is hard to identity the causal relations are less suitable to the deployment of Bayesian networks. Similarly, where there is no uncertainty about the cause-effect relationships, e.g., the conditional probabilities are deterministic, i.e., they take the value 1 or 0, a better approach for decision-making probably exists.

For some problems of pattern recognition, e.g. of fingerprints, there may be no clearly defined mechanism that controls the layout of the pattern. It would therefore be difficult to build Bayesian networks that would be of significant use in this problem area. Finally, in order that the time and effort spent in constructing a Bayesian network is justified, it is important that the problem solving is repetitive in nature. Examples of problems that need to be solved repeatedly include deciding whether or not to offer car insurance to a driver or deciding if rainfall is likely, whilst the decision on the best location for a new national sports stadium is not likely to be needed more than once. It is worth noting that the majority of real-world decisions are repetitive in nature. Whilst there are situations where Bayesian networks may not be the most suitable choice for decision-making, the proliferation of applications that they have been applied to highlights their importance.

3.9. *Applications of Bayesian networks*

The greatest testament to any technology is the extent to which it is used. There are numerous practical applications of Bayesian decision-making across a wide range of different and diverse areas. Microsoft's Lumiere project (Horvitz et al. 1998; Lumiere 1998) developed an architecture for reasoning about the goals and needs of software users. In Lumiere, Bayesian networks model relationships between the goals and needs of a user and observations about the current program state. The user's intentions and needs are inferred based on the current context, previous actions and queries. Additionally, the system also computes the likelihood that the user would like help completing their current task. The technology developed in Lumiere was applied in the design of the Bayesian help system in the Office 1997 and Office 2003 product suites. Microsoft also used Bayesian networks in the design of Microsoft Pregnancy and Child Care (Haddawy 1999), which provides online heath information to parents. This involved the construction of Bayesian networks for common symptoms in children. The appropriate model is then chosen at run-time based on the primary complaint.

The VISTA project (Horvitz & Barry 1995) focused on providing online decision support for space shuttle flight controllers. A key aspect of this work was the management of the time-critical, complex information that is displayed to the flight controller. Bayesian networks are used to interpret live telemetry and determine the likelihood of problems in the propulsion systems in the space shuttle. A list of problems ordered by criticality and likelihood are presented. The level of detail displayed is controlled by a model of time criticality, enabling the flight controller to focus on the most important information at any given time. Lockheed Martin's Marine Systems have developed an Autonomous Control Logic (ACL) system for use in an Unmanned Underwater Vehicle (UUV) (Haddawy 1999). The ACL architecture applies both rule-based and Bayesian decision-making to guide the UUV. The rule-based component is concerned with real-time response, whilst the Bayesian model-based component focuses on diagnosis, analysis and decision-making about unexpected events. A Bayesian network models both the capabilities of the vehicle and the uncertainty on the current state of these capabilities, before deciding on the best possible response to the event.

Pathfinder (Heckerman et al. 1992) is an expert system for providing advice to surgical pathologists to assist the diagnosis of lymph-node diseases. Pathfinder is, "one of a growing number of normative expert systems that use probability and decision theory to acquire, represent, manipulate, and explain uncertain medical knowledge" (Heckerman et al. 1992, p.1). The Pathfinder technology evolved into the commercialised Intellipath group of systems. The Intellipath modules present competing diagnoses of possible diseases based on histological features that are input to the system. The user can then identify the features that best distinguish between competing diagnoses, whilst considering the cost and potential benefits of each observation or test. Another example of Bayesian networks being applied in medical diagnosis is discussed in Milho & Fred (2000) which presents a Web supported development tool for medical diagnostic applications. More information on the application of abductive inference models to medical diagnosis can be found Peng & Reggia (1990), whilst Browne et al. (2006) discuss the application of Bayesian network approaches to predict Protein-Protein Interactions (PPI) in biological systems. Bayesian network have also been applied in the areas of machine vision (Levitt et al. 1990), story understanding (Charniak & Goldman 1989, 1991), economic forecasting (Abramson 1991) and risk analysis (Agena 2009). Hugin (2009) gives case studies of where Bayesian networks have been applied in many different areas, including business intelligence, earthquake risk management, crime control planning, food production design, customer support operations and mobile robotics.

3.10. *Bayesian networks in multimodal systems*

Bayesian networks are becoming utilised increasingly in the field of multimodal systems. XWand (Wilson & Shafer 2003), as discussed in Chapter 2, Section 2.8.17, is a wireless sensor package enabling natural interaction within intelligent environments. XWand implements a dynamic Bayesian network for action selection within an intelligent space - focussing on the home environment. XWand can be used to turn a lamp off and on, control a media player, and act as a mouse - controlling the windows curser. XWand uses a dynamic Bayesian network (see Figure 3.7) to perform multimodal integration. The network in Figure 3.7 makes decisions based on sensors, referent, i.e., what the user is believed to be referring to, and a command that corresponds to the referent.

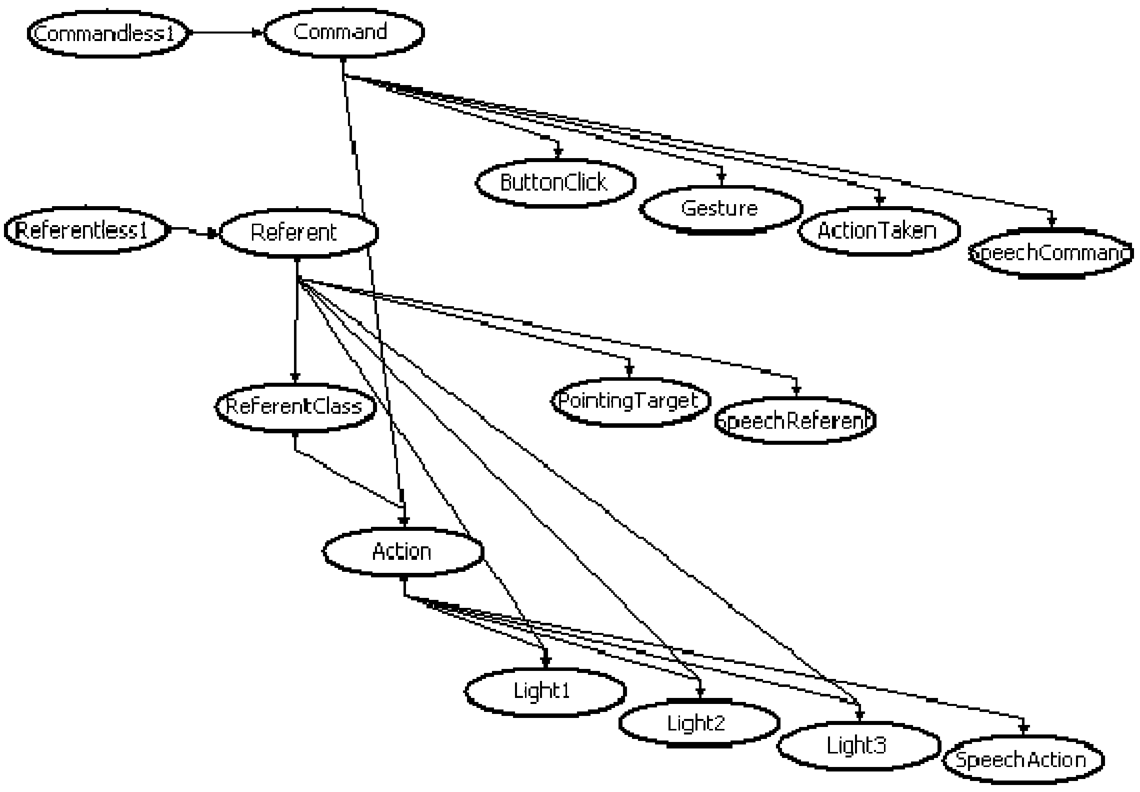


Figure 3.7: Dynamic Bayesian network in XWAND (XWAND 2009)

The Bayesian network in Figure 3.7 enables the referent to be identified in two ways: (1) it is determined by where the wand is pointing, and (2) it may be identified using speech recognition events. There are three ways to issue a command: (1) performing a wand gesture, (2) clicking a button or (3) issuing a spoken command. It is therefore possible for the same action to be specified in different ways, e.g., speech, gesture, pointing, clicking. For example, as discussed in Wilson & Shafer (2003), all the following are possible ways of turning on a lamp:

- Uttering “turn on the desk lamp”
- Pointing at the desk lamp and saying “turn on”
- Pointing at the lamp and performing the “turn on” gesture with the wand
- Uttering “desk lamp” and performing the “turn on” gesture

The mechanism also ensures that spurious speech recognition results are ignored, e.g., “volume up” while the wand is pointing at the desk lamp. As an example of how the Bayesian network in Figure 3.7 works in practice, suppose that the wand is pointing at a light. This causes the *PointingTarget* variable to be set to *Light1*. The Action node assigns equal probability to its two possible states: *TurnOnLight* and *TurnOffLight*. If the user says “turn on”, the speech node is then set to *TurnOn*; the probabilities of the *Light1* node are dynamically updated, i.e., the probability of *TurnLightOn* drastically increases whilst the probability of *TurnLightOff* is decreased. Based on the new probability distribution the system then decides to turn the light on.

Greta (de Rosis et al. 2003), as discussed in Chapter 2, Section 2.9, is an Embodied Conversational Agent (ECA) that can engage in natural and believable conversation with both real users and other agents. Greta denotes both a real user and another agent as an Interlocutor (*I*). Greta implements Bayesian networks for computing probabilities of all possible gaze states of the agent. Bayesian networks have been developed to represent the triggering of emotional states for the agent such as ‘envy’ and ‘happy-for’. More specifically, Bayesian networks are used to represent the uncertainty in the agent’s belief about the possibility of achieving certain goals and the utility assigned to achieving these goals. Bayesian networks model the relationships between the beliefs in Greta’s mind by using nodes to represent goal achievement. Figure 3.8 shows the Bayesian network for triggering ‘envy’. In the triggering of ‘envy’ the goal is to not have less power than others (or to dominate others). As stated in de Rosis et al. (2003, p. 90), “the Agent’s belief about the probability of achieving this goal is influenced by her belief that some desirable event occurred to some other agent *I* and that the same event cannot occur to itself because the two events are ‘exclusive’: when some desirable event occurs to *I*, envy towards *I* may then increase, in Greta”. The Bayesian network depicted in Figure 3.8 consists of two sub-networks which represent the state of Greta’s mind at time *T* and time *T+1*. Considering the left sub-network first, we see that Greta (*G*) believes that the Interlocutor (*I*) will gain more power over her (represented by *Bel G (MPow I G i)*) if *I* is in possession of an object i *(Bel G (Has I i))* that Greta wants to get (*Goal G (Has G i)*) and if she is unable to get the same object herself (*Bel G n(Has G i)*). The sub-network on the right models the state of Greta's mind at time *T+1*. As shown at time *T+1*, the Interlocutor (*I*) saying he/she has the winning lottery ticket increases Greta's belief that *I* has the winning ticket (indicated by the directed edge from the *Say I G "I've got the winning ticket"* node to the *p (Bel G (Has I i)* node). This, coupled with Greta's desire to dominate others (*(Bel G Ach (Domin G I)*) increases the likelihood that Greta will be jealous of *I* (*Feel G (Envy I)*). As discussed in de Rosis (2003), the intensity of the 'envy' emotion is dependent upon whether or not Greta is a 'dominant' agent. A counteracting event may cause the intensity of the emotion to decrease. For example, if Greta learns in the time interval *T+2* that the prize for the winning ticket is very small, then Greta's envy will decrease.

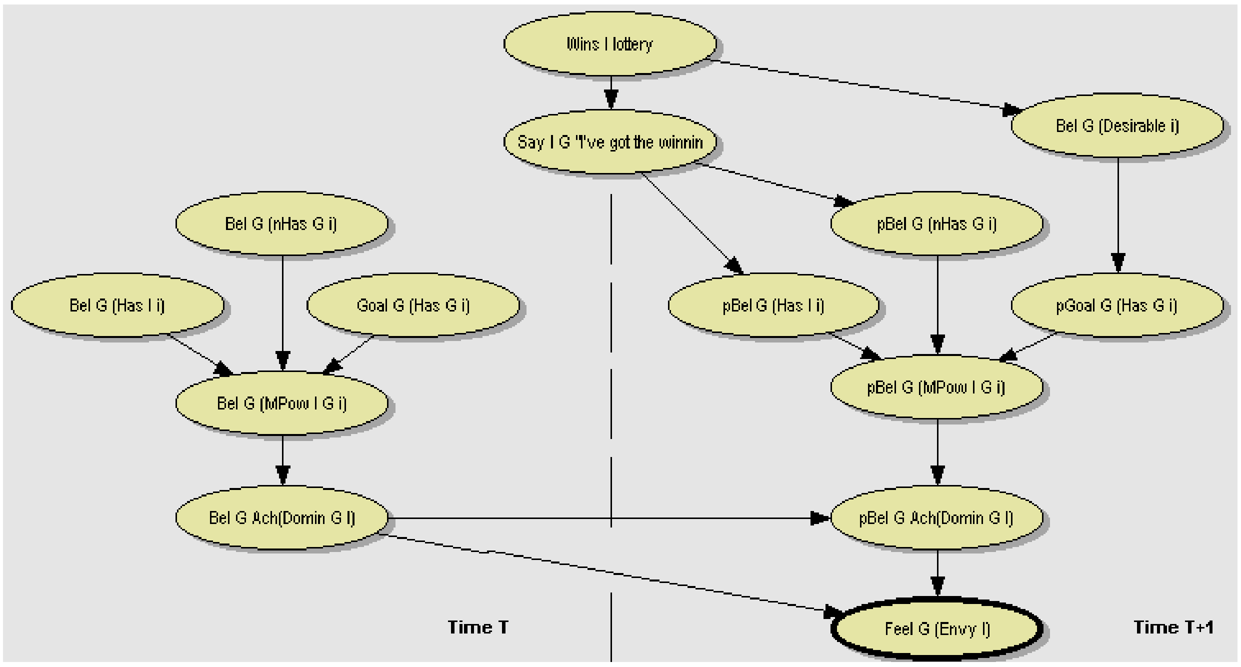


Figure 3.8: Bayesian network for triggering 'envy' (de Rosis et al. 2003)

In Vybornova et al. (2007), Bayesian networks are applied to multimodal fusion using contextual information. An initial application of the technology is an intelligent diary that assists elderly people who live alone. The intelligent diary aims to help the elderly, "perform their daily activities, prolong their safety, security and personal autonomy, and support social cohesion" (Vybornova et al. 2007, p. 61). In order to allow more natural interaction with human users the research focuses on the interpretation of human behaviour and the recognition of the user's intentions. This requires both low level (signal) and high level (semantic) multimodal fusion. Vybornova et al. (2007, p. 61) observe that, "everything said or done is meaningful only in its particular context", and therefore, to perform semantic fusion, information is taken from at least three contexts: (1) domain context, (2) linguistic context and (3) visual context. Bayesian networks are utilised for analysing and combining the modalities. Robust contextual fusion is achieved by applying probabilistic weighting of the multimodal data streams. This enables recognition of user intention, prediction of human behaviour and interpretation and reasoning about the user's cognitive status.

A Bayesian network approach to fusion within a multimodal automated surveillance system is discussed in Zou and Bhanu (2005, p. 4) who explain that automated surveillance, "addresses real-time observation of people, vehicles and other moving objects within a complicated environment, leading to a description of their actions and interactions". Zou and Bhanu (2005) compare two approaches to multimodal fusion in the human detection and tracking application domain: time-delay neural networks and Bayesian networks. Two signal modalities, visual and audio are fused in order to detect a person walking in a scene. The fusion mechanism is motivated by their investigation of the relationship between step sounds and visual motions. These relationships are subsequently modelled with Bayesian networks.

IM2 (Interactive Multimodal Information Management) (IM2 2009) aims to develop innovative technologies that support multimodal human-computer interaction. This objective is pursued through research in computer vision, multimedia indexing, speech understanding and multi-channel fusion. Multimodal input modalities include speech, pen, gesture and head/body movements, whilst multimedia system output includes speech, sound, animation, images and 3D graphics. One of the main applications of the work developed by IM2 is smart meeting rooms. Bayesian networks have been proposed by IM2 researchers as a means of addressing problems associated with dynamic data fusion.

Cohen-Rose and Christiansen (2002) discuss a storytelling system (called Guide) that assists the user by answering queries about where to eat and drink. Guide's interface combines speech, text, graphics, mouse-based and pen-based pointing and positional input (e.g. GPS). Guide is inspired by the science fiction novel, "The Hitchhikers Guide to the Galaxy" (Adams 1979), which is based on the existence of an electronic guide to absolutely everything. Cohen-Rose and Christiansen (2002) suggest that the Internet could be viewed as being similar to the fictional guide envisaged in Adams (1979) and discuss the limitations of existing search engines. The authors explain how Guide uses Bayesian decision-making to present more relevant information from a variety of sources in a contextual setting.

3.11. *Tools for implementing Bayesian networks*

This section reviews tools for performing Bayesian decision-making. Such tools enable the implementation of Bayesian networks using a Graphical User Interface (GUI), an Application Programming Interface (API), or both. The use of a GUI enables networks to be constructed using the familiar notion of cause and effect, whilst the use of an API is necessary to access the decision-making capabilities of the networks from within a software program.

3.11.1. MSBNx

MSBNx (Kadie et al. 2001, MSBNx 2009) is a Windows application for creating and evaluating Bayesian networks. It is available at no cost for non-commercial use and can be downloaded from MSBNx (2009). Components of this Microsoft application can be integrated into programs, enabling them to perform inference and decision-making under uncertainty. In addition to the components provided by MSBNx, developers and researchers can create their own add-in components that can be used within MSBNx. The package itself includes an add-in for editing and evaluating Hidden Markov Models. Bayesian networks are encoded in an XML-based format and the application will run on any Windows operating system from Windows 98 to XP. The MSBNx Model Diagram Window is shown in Figure 3.9.

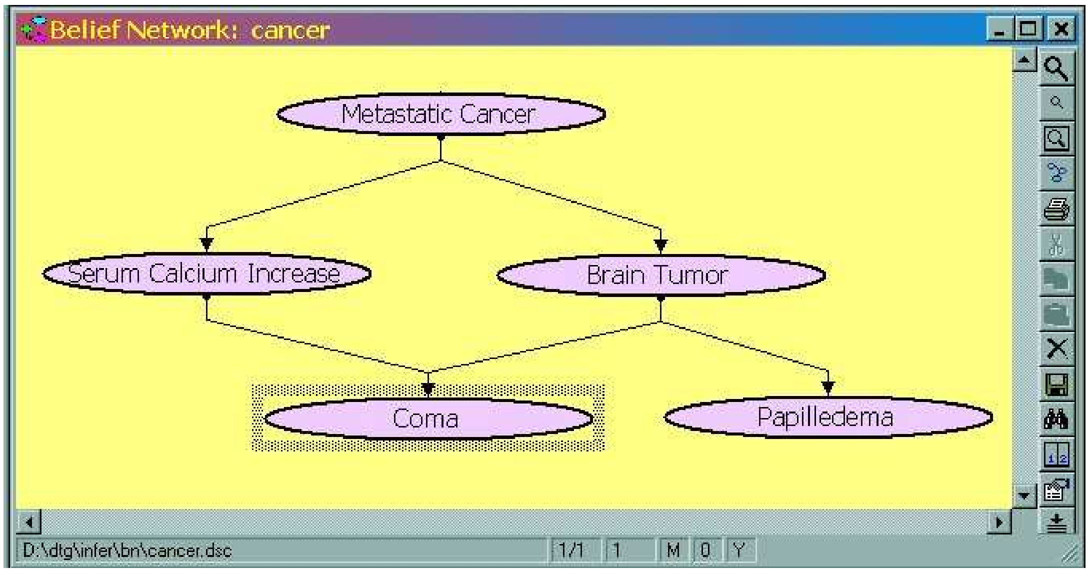


Figure 3.9: Model Diagram Window in MSDNx Editor (Kadie et al. 2001)

MSBN3, an ActiveX DLL, is the most important component of MSBNx. MSBN3 provides developers with a powerful COM-based API for creating and evaluating Bayesian networks. The API is particularly suited for use with COM-friendly programming languages such as Visual Basic and JScript. If Bayesian networks have been developed with machine learning tools, MSBNx can edit and evaluate the results. The WinMine Toolkit (WinMine 2009), also developed by Microsoft, can create Bayesian networks with machine learning and statistical methods. Models created by WinMine can be loaded, edited and evaluated with MSBNx.

3.11.2. GeNIe

The GeNIe GUI (Genie 2009) is shown in Figure 3.10. GeNIe is the graphical user interface to SMILE (Structural Modelling, Inference, and Learning Engine) (Genie 2009). GeNIe supports all major file types, including Bayesian networks developed using the Hugin (Hugin 2009) and Netica (Norsys 2009) software tools. GeNIe is implemented in Visual C++ and only runs under the Windows family of operating systems (98, NT, 2000, XP).

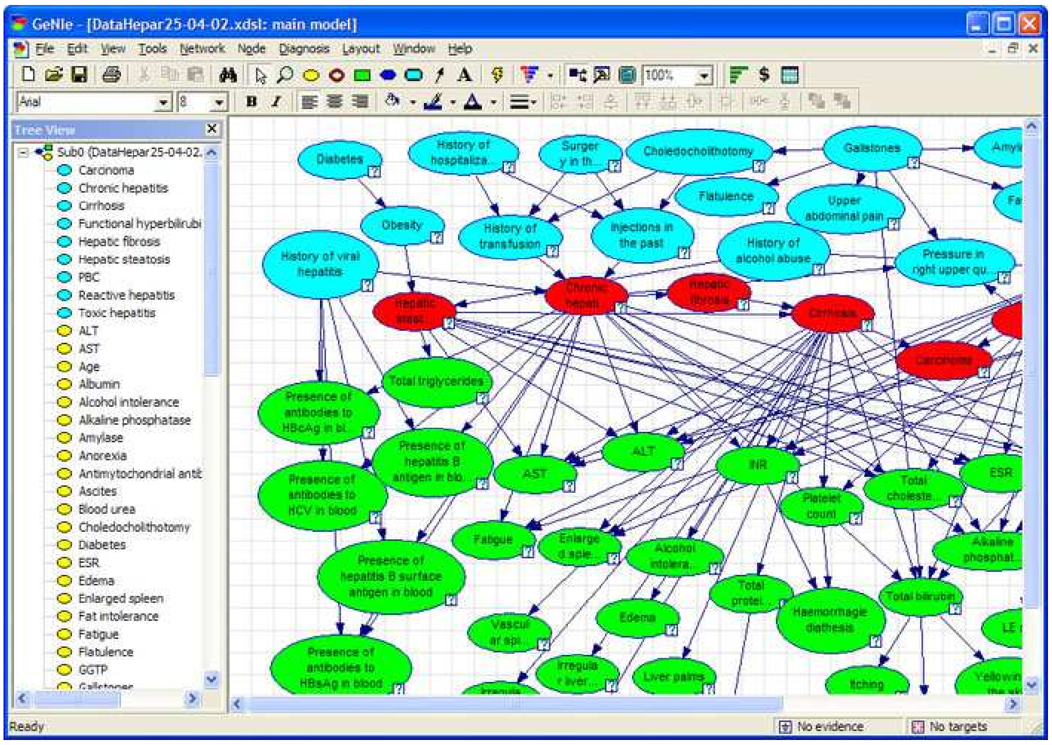


Figure 3.10: GeNIe GUI (Genie 2009)

GeNIe can learn the structure of Bayesian networks from data. It also contains a background knowledge editor which enables the developer to force or forbid arcs between variables and assign variables to temporal tiers. Background knowledge can be saved and previous knowledge loaded to influence the learning process. GeNIe also enables the parameters of an existing network to be learned from a data file. SMILE (Genie 2009) is a platform independent library of functions that can be used by programmers and developers to implement Bayesian networks and influence diagrams. SMILE is implemented in C++ and defines functions for creating, editing, saving and loading graphical models for probabilistic reasoning and decision-making under uncertainty. The SMILE library acts as a set of tools that can be used by the application program, which has full control over the model building and reasoning process.

3.11.3. Netica

Netica (Norsys 2009) enables problem solving with Bayesian networks and influence diagrams. A demo version can be downloaded at no cost from Norsys (2009). The demo version has full functionality, but is limited in model size. The complete application enables the building, editing and learning of Bayesian networks and influence diagrams. Netica compiles belief networks into junction trees of cliques to enable fast probabilistic reasoning. Netica's GUI is shown in Figure 3.11.

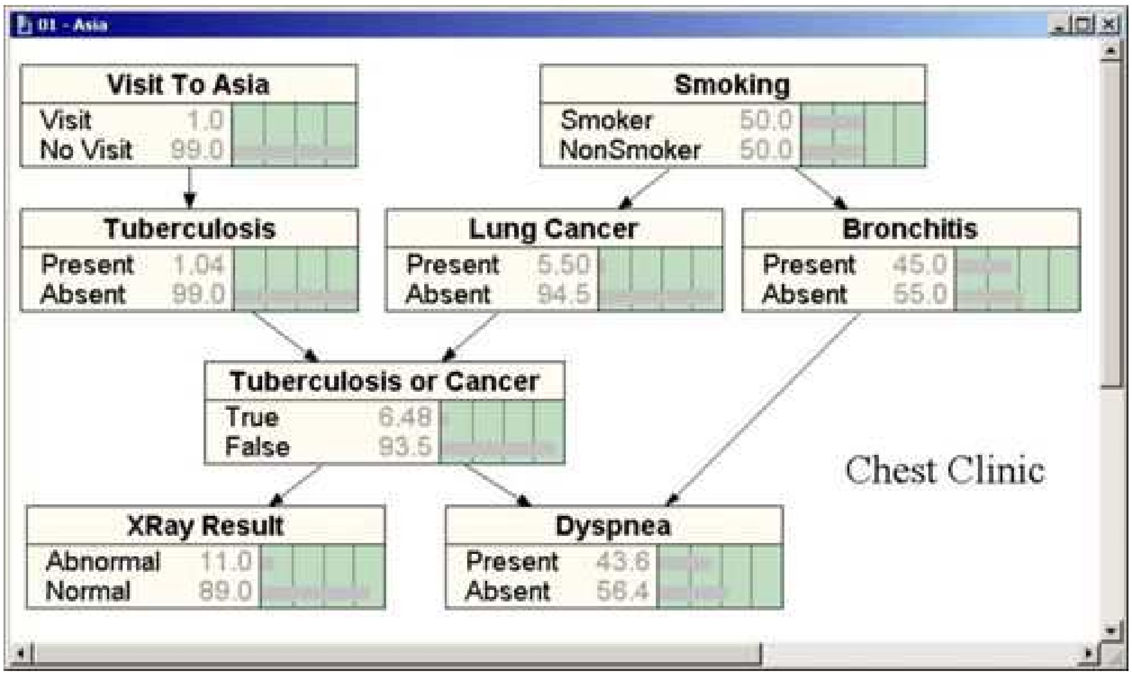


Figure 3.11: Netica GUI (Norsys 2009)

Netica also offers the Netica Programmers Library, or the Netica API, to enable programmers to embed the functionality of Netica within their own applications. The API is available in C, C++, C#, Java, Visual Basic, Matlab and CLisp. Versions of the API are available for Windows, Solaris and Macintosh. The interface for each of these operating systems is identical, so code is fully portable across all of the platforms. To facilitate learning from data (EM and gradient descent learning) Netica can be connected to a database or a Microsoft Excel spreadsheet.

3.11.4. Elvira

Elvira (Elvira 2009) (see Figure 3.12) is a tool for constructing and evaluating Bayesian networks and influence diagrams. Elvira is the product of a joint program of research and development by a consortium of Spanish researchers. The Elvira program is written in Java and can therefore run on different operating systems, including Windows, Linux and Solaris. Figure 3.12 shows the Elvira GUI in edit mode for a simple Bayesian network containing 2 nodes: *Disease* and *Test*. The program is switched to Inference mode, i.e., run mode, by selecting *Inference* from the drop down menu on the top toolbar. This causes the Bayesian network to be compiled (see Figure 3.13) and, in this example, computes the values of the *Test* node based on the values of the *Disease* node. As shown in Figure 3.13, in inference mode Elvira shows the probability of each value using both a number and a pair of horizontal bars. Elvira offers exact and approximate algorithms for both discrete and continuous variables. Explanation methods and decision-making algorithms are also available, as is the ability to learn from a database.

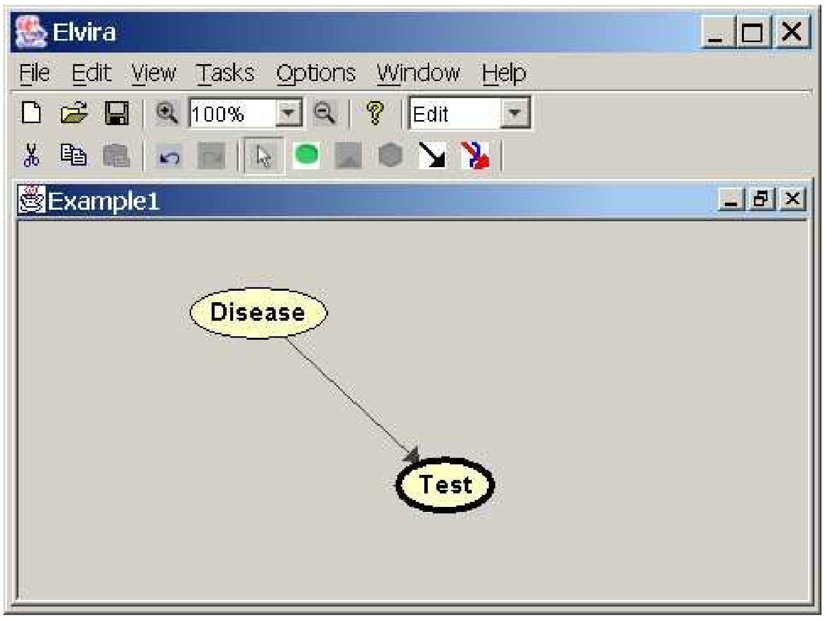


Figure 3.12: Elvira's main screen (Elvira 2009)

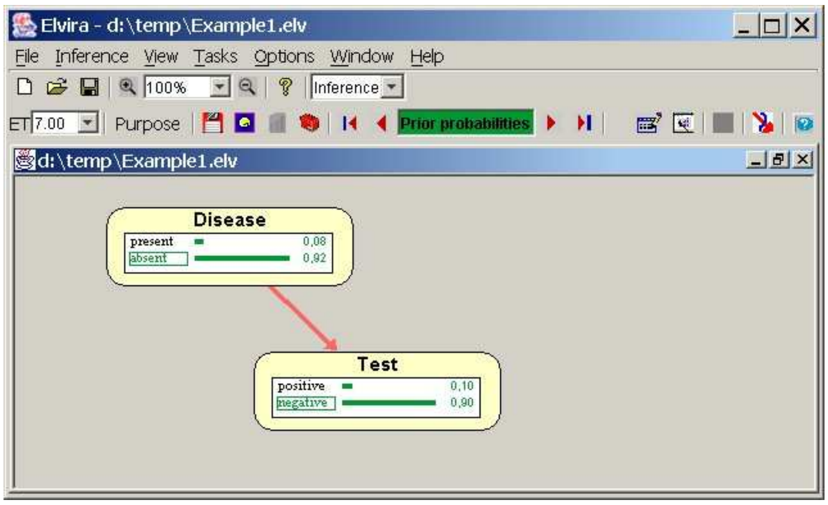


Figure 3.13: Elvira in inference mode (Elvira, 2009)

3.11.5. Hugin

Hugin (HUGIN 2009; Jensen 1996) offers both a Graphical User Interface (GUI) and an Application Programming Interface (API) for constructing and running Bayesian networks. The Hugin GUI (Graphical User Interface) is a tool for creating and implementing Bayesian networks and influence diagrams for decision-making. The GUI provides an intuitive interface to the Hugin decision engine. APIs are available in C, C++ and Java. There is also an ActiveX server for use with Visual Basic for Applications. Hugin is compatible with the Windows 2000/XP/Vista, MAC, Linux and Solaris operating systems. Hugin can also learn the structure of a Bayesian network from a database of cases. When the Hugin GUI is started the main window is displayed, as shown in Figure 3.14.

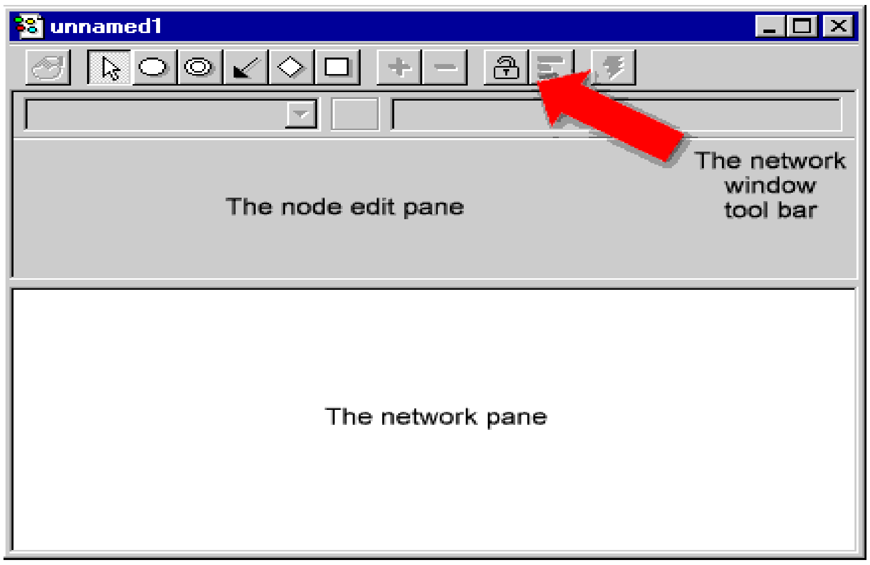


Figure 3.14: Main window of Hugin GUI (Hugin 2009)

As shown in Figure 3.14, the main window contains a toolbar, a node edit pane and a document pane. The GUI automatically starts up in edit mode, enabling developers to immediately start constructing their Bayesian networks. Nodes can be added to the network by selecting the *Discrete chance tool* from the toolbar and clicking anywhere on the network pane. Links are added between nodes by selecting the *Link tool* button from the toolbar and dragging a link from the influencing node to the influenced node. Figure 3.15 shows a simple example of a Bayesian network implemented in the Hugin GUI. Note from Figure 3.15, that the nodes *Diet* and *Exercise* both have influence over the *Weight Loss* node, as indicated by the causal arrows. *Diet* and *Exercise* nodes are therefore influencing nodes, while *Weight Loss* is the influenced node.

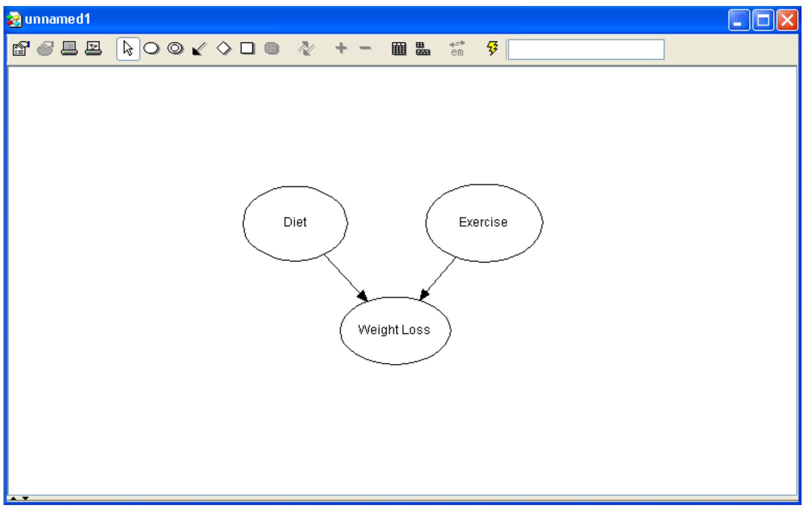


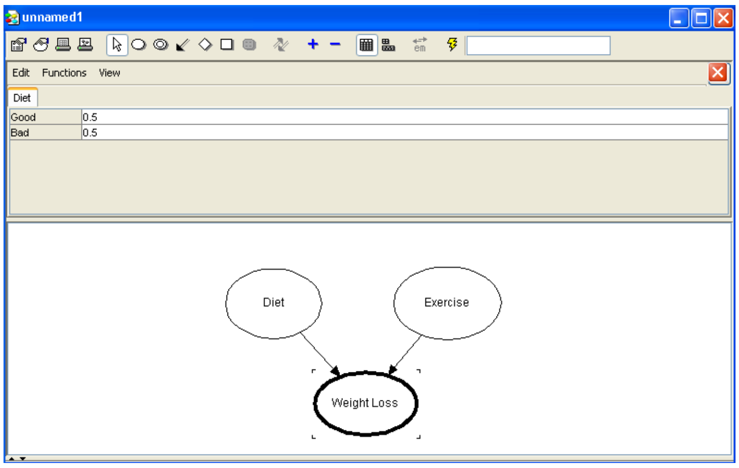
Figure 3.15: Simple example of a Bayesian network in Hugin

The next step in this example is to specify the states for each of the nodes. This requires that we open the tables pane by clicking on the *Show node tables* button. To specify the states of *Diet*, we must do the following:

- Select the *Diet* node by left clicking inside the node. This causes the table for *Diet* to appear in the tables pane, as shown in Figure 3.16.
- We can now rename the "State 1" and "State 2" states to more meaningful names such as "Good" and "Bad".
- We now enter values into the Conditional Probability Table (CPT) for the *Diet* node. Note that, by default, the Hugin GUI has given all entries in the CPT a value of 1.

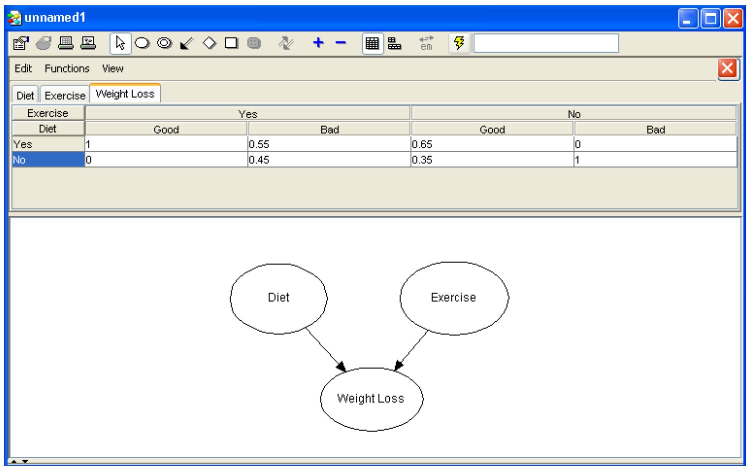
The previous procedure would then be repeated for the *Exercise* and *Weight Loss* nodes. To continue this example we will assign 0.5 to the "Good" and "Bad" states of the *Diet* node, and to the "Yes" and "No" states of the *Exercise* node. We then assign appropriate values to the states of *Weight Loss* as illustrated in Figure 3.17. Note from Figure 3.17, that the CPT of *Weight Loss* is larger since it accounts for the parent nodes of *Diet* and *Exercise*. Pressing the *Switch to Run Mode* button on the toolbar causes the network to be compiled. This involves checking for errors, for example, making sure that the probabilities of each state has a sum of 1. Figure 3.18 shows the weight loss example in run mode.

We now add evidence to the network. To do this we can double-click on a state in the tree structure to assert 100% belief, or we can right click and select, 'Enter Likelihood …', which enables us to enter a value between 0 and 100. To assert the belief that *Diet* is definitely good we double-click on the "Good" state causing it to turn red, indicating that evidence has been added.



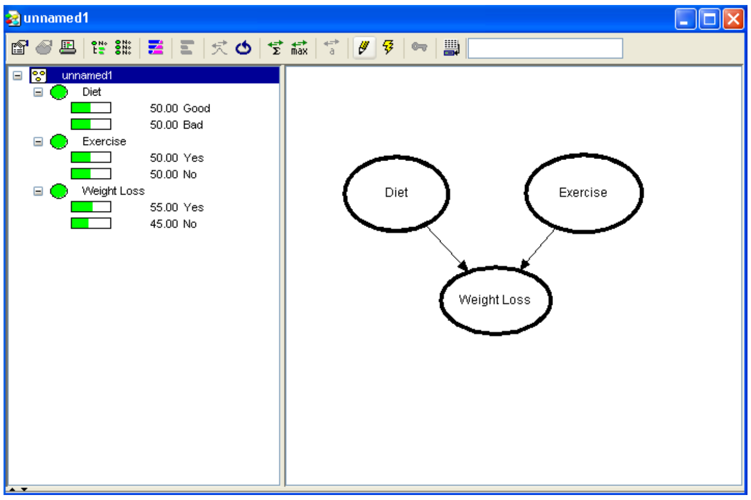
Diet	
Good	0.5
Bad	0.5

Figure 3.16: View of table for *Diet* node



Exercise	Yes		No	
Diet	Good	Bad	Good	Bad
Yes	1	0.55	0.65	0
No	0	0.45	0.35	1

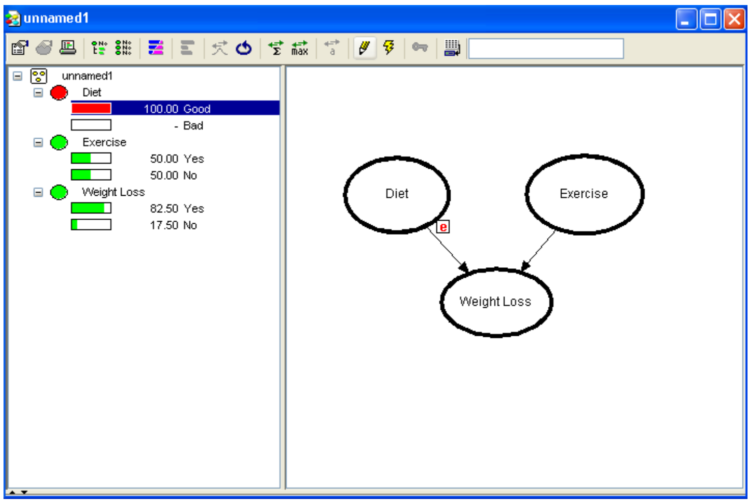
Figure 3.17: CPT of *Weight Loss* node



```
unnamed1
  Diet
      50.00 Good
      50.00 Bad
  Exercise
      50.00 Yes
      50.00 No
  Weight Loss
      55.00 Yes
      45.00 No
```

Figure 3.18: Example of run mode

Note from Figure 3.19, that a red coloured letter ‘e’ appears beside the *Diet* node in the Bayesian network, indicating that evidence has been added to the node.



```
unnamed1
  Diet
     100.00 Good
          - Bad
  Exercise
      50.00 Yes
      50.00 No
  Weight Loss
      82.50 Yes
      17.50 No
```

Figure 3.19: Evidence added to the *Diet* node

As shown in Figure 3.19, when evidence has been added that *Diet* is “Good”, the probability of weight loss has risen from 55% to 82.5%. If we now add evidence that we are certain that the person does regular exercise, the probability of weight loss rises to 100% (see Figure 3.20).

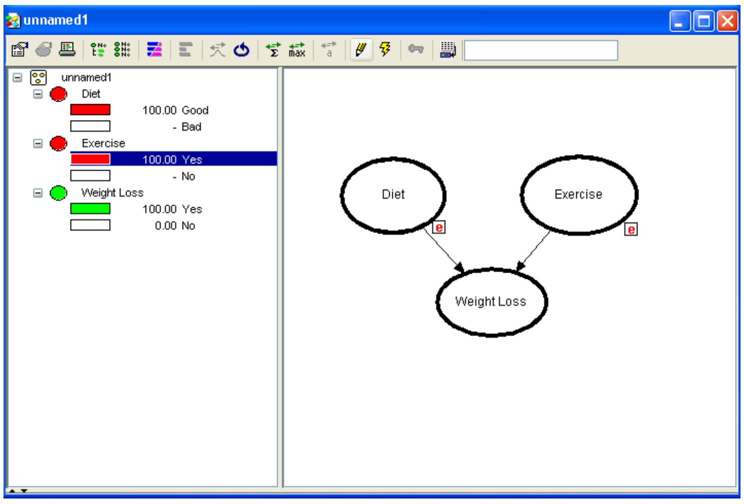
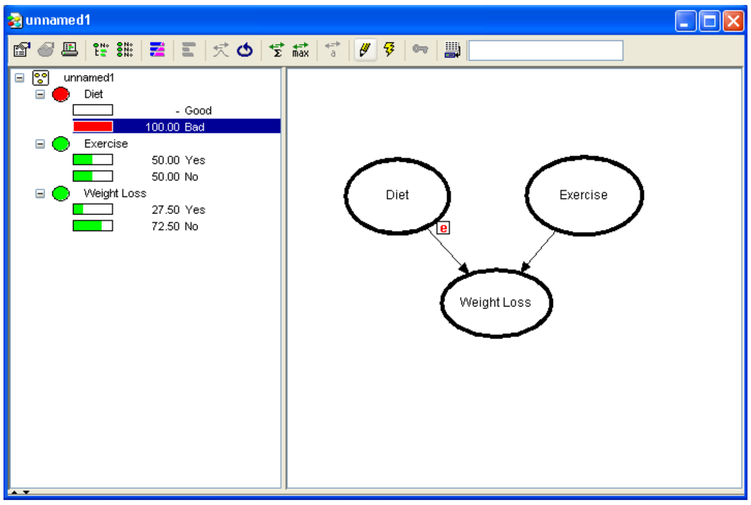


Figure 3.20: Evidence of a good diet and exercise

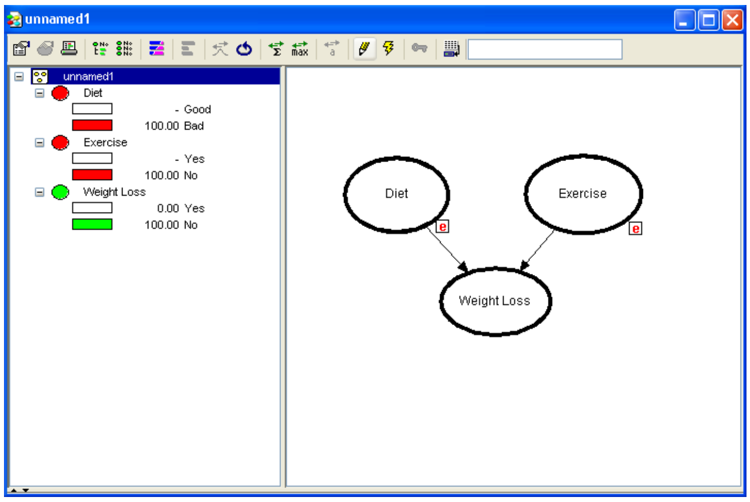
To continue the example we will remove the evidence from the *Exercise* node, by right clicking and selecting ‘Retract Evidence’, and change the evidence on the *Diet* node to indicate the certainty that the diet is definitely bad. This, as expected, causes the probability that there will be weight loss to reduce drastically. This is illustrated in Figure 3.21, where the probability of weight loss has become 27.5%. If we now assert the belief that there is a bad diet and no exercise, then the probability of weight loss changes to 0%, as shown in Figure 3.22. Therefore we are absolutely certain that a bad diet and no exercise will not result in any weight loss.

Thus, the Hugin software tools enable the creation of Bayesian networks to model real world scenarios. The simple weight loss example discussed here did not involve any decisions, but it could be easily changed to do so. For example, the *Weight Loss* node could be replaced with a *Join a Gym* or a *Make changes to Lifestyle* decision node. When our Bayesian network has been constructed, we can enter evidence into the model and view the results dynamically. Hugin automatically performs all the mathematical calculations and computes new probabilities whenever additional information has been added. The Hugin API is implemented in the form of a library written in the C, C++ and Java programming languages. The API can be used like any other library and can be linked to applications, enabling them to implement Bayesian decision-making. The Hugin API encloses a high performance inference engine that, when given descriptions of causal relationships, can perform fast and accurate reasoning. Full documentation is provided for the C, C++ and Java versions of the API (Hugin 2009). The Hugin API allows a flexible approach to error handling. When errors occur, the API informs the application program and lets it decide on an appropriate action. Thus the developer is given maximum freedom regarding how to deal with errors. Errors are communicated to the user of the API using the *h_error_t* enumeration type. All of the API functions return a value. Functions that don't return anything, i.e., void return type, have a return type *h_status_t*. In such cases, if a zero is returned then the function has run successfully, while a non-zero result indicates failure, i.e., an error has occurred.



```
unnamed1
  Diet
          - Good
     100.00 Bad
  Exercise
      50.00 Yes
      50.00 No
  Weight Loss
      27.50 Yes
      72.50 No
```

Figure 3.21: Evidence of a bad diet added



```
unnamed1
  Diet
          - Good
     100.00 Bad
  Exercise
          - Yes
     100.00 No
  Weight Loss
       0.00 Yes
     100.00 No
```

Figure 3.22: Evidence of bad diet and no exercise added

Within the Hugin API, all objects are represented as opaque pointers. An opaque pointer points to data that is not further defined. Opaque pointers enable the references to data to be manipulated, without requiring knowledge of the structure of the actual data. The Hugin API enables several data types including integer, string and boolean. Functions exist for creating new domains, inserting nodes and adding directed edges between nodes. Functions are also provided for removing parents from a node, replacing one parent with another parent, reversing a directed edge between two nodes and numerous other operations such as retrieving the current parents and children of a node and specifying the number of states associated with a node.

Hugin can facilitate two types of learning: structural learning and parametric learning. Structural learning is performed using the PC algorithm and can be accessed via the *File* menu and the structural learning icon. The PC (Peter-Clark) algorithm (Spirtes et al. 2000) is an extension of the IC (Inductive Causation) algorithm (Pearl 2000, p. 50) that tests for conditional independence, or partial correlation, between variables. Therefore, in structural learning, Hugin learns the dependencies between variables in the data and, based on these relations, determines the structure of the Bayesian network. Consider the segment of the data file presented in Figure 3.23.

```
X,B,D,A,S,L,T
no,no,no,no,no,no,no
yes,yes,yes,no,yes,yes,no
yes,yes,yes,no,yes,yes,no
no,no,yes,no,no,no,no
no,no,no,no,no,no,no
no,yes,yes,no,yes,no,no
no,yes,no,no,N/A,no,no
no,no,no,yes,yes,no,no
no,yes,yes,no,yes,no,no
no,no,no,no,no,no,no
no,yes,no,no,N/A,no,no
no,no,no,no,yes,no,no
no,no,no,no,no,no,no
no,yes,yes,no,yes,no,no
no,no,no,no,no,no,no
```

Figure 3.23: Data file for structural learning (Hugin 2009)

Running the structural learning algorithm on the data file illustrated in Figure 3.23 enables Hugin to learn the structure of a Bayesian network that represents the cause-effect relationships embedded in the data, as shown in Figure 3.24.

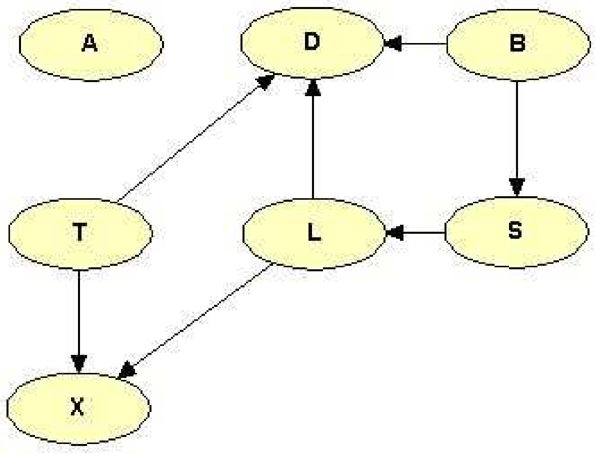
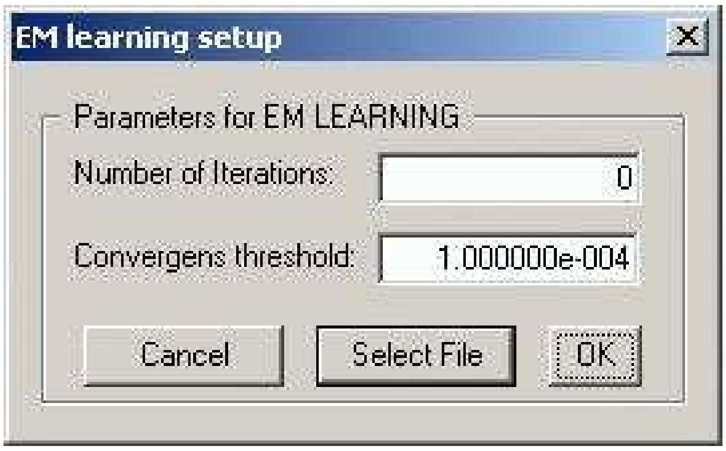


Figure 3.24: The Bayesian network learned (Hugin 2009)

The previous example discusses only learning the structure of a Bayesian network. To learn the parameters of a network, or Conditional Probability Tables (CPTs), parametric learning is used. There are two types of parametric learning supported by Hugin: Adaptive learning and EM (Estimation-Maximum) learning. Adaptive learning can adapt the CPTs of a Bayesian network to a new dataset. Experience tables are used to perform adaptation. Experience nodes can be added to some or all of the discrete chance nodes in a Bayesian network. The adaptation process involves entering evidence, propagating the evidence through the Bayesian network and updating (or adapting) the CPTs and experience tables. Following adaptation the experience nodes can be deleted and the current values of the CPTs will then form the new conditional distribution probabilities of the nodes in the Bayesian network. EM learning uses data stored in a database to generate CPTs in a Bayesian network. The EM learning facility is accessed via the 'EM Learning' icon. Clicking on this icon opens the EM Learning window shown in Figure 3.25.


EM learning setup

Parameters for EM LEARNING

Number of Iterations: 0

Convergens threshold: 1.000000e-004

Figure 3.25: EM Learning window (Hugin 2009)

Selecting the data file and clicking *OK* runs the EM algorithm and computes new conditional distribution probabilities for each of the nodes based on the case set given in the data file.

3.11.6. Additional Bayesian modelling software

The Bayes Net Toolbox (BNT) (Murphy 2009) is an open source Matlab package for developing probabilistic graphical models for use in statistics, machine learning and engineering. Although BNT is marketed as an 'open-source' package, it can be argued that it is not truly open-source due to its reliance on Matlab. BNT was initially designed for use with Bayesian networks (hence the name Bayes Net), but it has since been extended to deal with influence diagrams. Bayesian networks are represented within BNT as a structure containing the graph as well as the Conditional Probability Distributions (CPDs). One of the main advantages of BNT is the wide variety of inference algorithms that it offers. It also offers multiple implementations of the same algorithm, e.g. Matlab and C versions. Bayesian and constraint-based structure learning are both supported in BNT. Several methods of parameter learning are also supported, including EM (Estimation-Maximum), and additional methods of structure and parameter learning can be easily added.

BUGS (Bayesian inference Using Gibbs Sampling) (BUGS 2009) can perform Bayesian analysis of complex statistical models using Markov Chain Monte Carlo (MCMC) Methods (Neal 1993). Since its development began in 1989, several versions of BUGS have been released. WinBUGS 1.4.1, released in September 2004, aims to make practical MCMC methods available for use in probabilistic inference. Although WinBUGS does not provide an API, it is possible to call WinBUGS from other programs. The package allows graphical representations of Bayesian models through the use of its DoodleBUGS facility. JavaBayes (CMU 2009) is a set of software tools for creating and manipulating Bayesian networks using Java. JavaBayes offers a graphical Interface, an inference engine, a collection of parsers and is freely available under the GNU General Public License. JavaBayes can be run both as an application and as an applet within a HTML document. A more comprehensive list of available Bayesian network software can be found in Murphy (2009).

3.11.7. Summary

This chapter has discussed a definition and brief history of Bayesian networks. This was followed by a discussion on the structure of Bayesian networks and on their ability to perform intercausal reasoning. An example Bayesian network was presented, before influence diagrams were discussed. Consideration was then given to the challenges, advantages and limitations of Bayesian networks. Previous applications of Bayesian networks were reviewed, with particular focus on their use in multimodal systems. Finally, a review of existing software and tools for implementing Bayesian networks was presented.

Chapter 4 Bayesian Decision-making in Multimodal Fusion and Synchronisation

Decision-making in multimodal systems is a complex task (Thórisson 2002), involving the representation and understanding of input and output semantics, distributed processing and maintenance of dialogue history along with domain-specific information, e.g., the number of movies currently showing, the coordinates of an office. Decision-making in such systems is becoming increasingly complex as advances in technology enable a much wider range of modalities to be captured and generated. The hub of a multimodal distributed platform must be capable of processing information relating to the various input/output modalities. The hub is primarily concerned with three key problems: (1) Semantic storage - often using a blackboard, (2) dialogue management – often involving fusion and synchronisation, and (3) decision-making. It must also act as a conduit between the various components of the system and the outside world and it must deploy an appropriate decision-making mechanism that enables the interaction between the system and user to be as intelligent and natural as possible. Decision-making must consider the current context and domain, the dialogue history and the beliefs associated with the various modalities.

This chapter presents a Bayesian approach to multimodal decision-making in a distributed platform hub. First, a generic architecture for a multimodal platform hub is presented. Then a discussion on the key problems and the nature of decision-making within multimodal systems is considered, with decisions categorised into two areas: (1) synchronisation of multimodal data and (2) multimodal data fusion. The problem of synchronisation is only partially addressed. The focus here is on decision-making with respect to multimodal semantic fusion. Semantic representation and ambiguity resolution are also considered in the context of decision-making. Features of a multimodal system that aid decision-making are discussed including distributed processing, dialogue history, domain-specific information and learning. A list of necessary and sufficient criteria required for a multimodal distributed platform hub is then presented. Finally, the rationale for a Bayesian approach to multimodal decision-making is proposed with a discussion on its advantages.

4.1. *Generic architecture of a multimodal distributed platform hub*

A typical architecture of a multimodal distributed platform hub is presented in Figure 4.1. The key functions, dialogue management, semantic representation and storage, decision-making and domain knowledge, of the platform hub are represented by separate modules in the conceptual architecture in Figure 4.1.

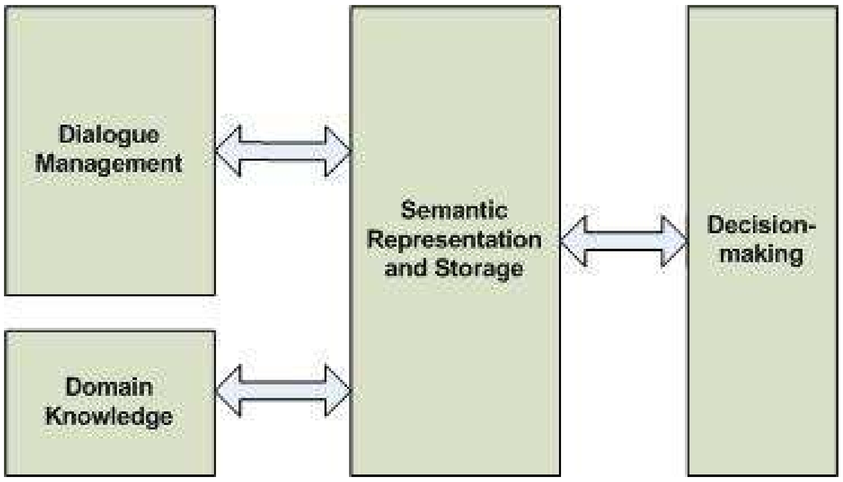


Figure 4.1: Generic architecture of a multimodal distributed platform hub

The *Dialogue Management* module of Figure 4.1 is responsible for coordinating the dialogue between the user and the multimodal system, and the communication between its internal modules. The *Decision-making* module is a crucial component of a multimodal system. The decision-making mechanism would typically use dialogue history and domain-specific information to make intelligent decisions that support multimodal interaction with the user. Examples of domain-specific information are the titles of movies currently showing in a cinema, the location and occupant of an office and the number of emergency exits in an auditorium. Examples of context information are the current speaker in a multimodal dialogue, the fact that a car is moving or stationary, and the current intentional state of a user. Multimodal semantics is usually stored in a shared space and a full dialogue history is maintained in this shared space to support future decision-making during a multimodal dialogue. Maintenance of dialogue history is the primary function of the *Semantic Representation and Storage (SRS)* module depicted in Figure 4.1. The *SRS* module is usually implemented in the form of a blackboard, as discussed in Chapter 2, Section 2.3. Multimodal semantics stored in the *SRS* module is processed by input and output processing modules such as NLP, eye-gaze tracking and image processing modules. Contextual knowledge is also stored in the *SRS* module. Information on the current context is used in conjunction with domain-specific information from the *Domain Knowledge* module to support intelligent multimodal decision-making. The generic architecture depicted in Figure 4.1 could take a number of alternative forms. For example, since decision-making is normally the responsibility of the *Dialogue Management* module, the *Decision-making* module may not be explicitly represented. It is also possible that the functionality of the distributed platform hub may be spread across different machines. Whatever the exact setup of the hub, it will always need to have mechanisms in place to support the key functionalities of dialogue management, domain knowledge retention and retrieval, semantic representation and storage, and decision-making.

4.2. *Decision-making in multimodal systems*

Although much has been achieved in the development of intelligent multimodal systems in recent years, many challenges still remain. Whilst recent research has resulted in systems capable of multimodal communication, this communication is very much on the computer's terms. The user must learn to use the system and the communication is constrained to suit the application. If we are to achieve truly human-like communication with computers, then the user must be able to dictate the terms of communication, i.e., the system must learn to meet the needs of the user instead of the user learning to use the system. In order to realise such systems we must investigate new, more intelligent, methods of representing multimodal input/output, communication and decision-making in multimodal systems.

Humans use a vast array of modalities to interact with each other including speech, gesture, facial expression, eye-gaze and touch. In order to achieve truly natural human-computer interaction, multimodal systems must be able to process these modalities in an intelligent and complementary manner. Such systems should be flexible, enabling the user to have appropriate control over the interaction modality. They must adapt to the changing needs of user interaction, switching from one modality to another as required. Communication must not be restricted to a particular modality, but should be facilitated using a variety of interaction modalities. Multimodal systems must also facilitate communication using a combination of modalities in parallel, e.g., speech and gesture, speech and gaze.

4.3. *Semantic representation and understanding*

Various approaches to semantic representation were discussed in Chapter 2, Section 2.2. Representing and understanding the semantics of multimodal input and output is an important task that must be performed in multimodal systems. Whilst the method of representing and understanding semantic content varies from system to system, the basic principle of representing information, using either frames (Minsky 1975) or XML, is prevalent within the majority of approaches. The marked-up semantics contains contextual information that is crucial to the decision-making process such as the current context, the current speaker, the module that produced the semantics, the module that should receive the semantics, the time the input was received, the time the output semantics was generated, the time at which the input/output becomes invalid (time to live), the confidence relating to multimodal recognition and the confidence associated with a decision or conclusion.

4.3.1. Frame-based semantic representation

An example semantic representation frame of multimodal input is shown in Figure 4.2. The example semantics given in Figure 4.2 contains frame-based semantic information sent from a posture recogniser to a dialogue manager of an intelligent in-car information presentation system. The first slot in the *POSTURE* frame is called *CONTEXT:*. Context information is important in enabling multimodal systems to behave differently depending on the current context. In this example, the value of the *CONTEXT:* slot is *CarMoving* and this information can be used by the in-car information presentation system to adapt its multimodal output accordingly, e.g., audio output only instead of an animated agent or graphical display.

```
[POSTURE
CONTEXT: CarMoving
FROM: PostureRecogniser
TO: DialogueManager
INPUT TYPE: posture
INTENTION: warning
HYPOTHESES [
HYPOTHESIS 1 [
POSTURE: tired
CONFIDENCE: 56.04%
  ]
HYPOTHESIS 2 [
POSTURE: angry
CONFIDENCE: 43.96%
  ]
 ]
TIMESTAMP: 011237432
TIMETOLIVE: 011239432
]
```

Figure 4.2: Example semantic representation of multimodal input

The second and third slots of the frame in Figure 4.2, *FROM:* and *TO:*, contain the module that produce the semantics and the module(s) which receive it. In this case, the semantics is produced by the *PostureRecogniser* module and is being sent to the *DialogueManager* module. The fourth slot of the example frame is *INPUT TYPE:* which in this case is simply posture. The *INTENTION:* slot is used here to indicate the purpose of the recognised input, i.e., to warn that the driver of the vehicle looks tired or angry. The sixth slot of the frame in Figure 4.2 is called *HYPOTHESES:* which contains one or more hypothesis about the mental state of the driver. In this case, there are two hypotheses: (1) that the driver is tired and (2) that the driver is angry. Note that each hypothesis slot also contains a *CONFIDENCE:* slot that identifies the confidence associated with each hypothesis. The *TIMESTAMP:* slot contains the time at which the input was detected. In this example, the format of the timestamp is a continuous string containing hour, minute, second, thousandth of a second, i.e., 011237432 represents 1:12 am and 37.432 seconds. Note that any format of timestamp can be used, provided it is understandable by the system and of a sufficient level of accuracy. Some applications may not need to be accurate to one thousandth of a second and, in these cases, a simpler timestamp would suffice. The final slot in the example frame of Figure 4.2 is *TIMETOLIVE:* and this contains the time at which the information contained in the frame becomes invalid. In this example, the input is valid for 2 seconds, after which time it may be discarded by the system.

Whilst much work is focused on representing the semantic content at the input of a multimodal system, representing the semantics of output is equally important. As observed by Wahlster (2003, p. 12), for a system to understand the semantics of its own output there should be, "no presentation without representation". Adherence to this principle is critical if a multimodal system is to handle commands such as, "show me a list of similar recipes to this one", "can you compare the features of this mobile phone to the previous two that I looked at?", and, "can I book two tickets to see the second movie you showed me?". These are examples of only a few requests that would become impossible to process if the system does not understand and keep a record of previous input/output.

4.3.2. XML-based semantic representation

Figure 4.3 shows an example semantic representation of multimodal output marked up in XML.

```
<output>
   <id>4454-1211-8754-3342</id>
   <from>DialogueManager</from>
   <to>PresentationPlanner</to>
   <text>The following movies are now showing:</text>
   <list>
   <item>
     <title>The Whole Nine Yards</title>
     <no>1</no>
   </item>
   <item>
      <title>The Green Mile</title>
      <no>2</no>
   </item>
   <item>
     <title>The Life of David Gale</title>
     <no>3</no>
   </item>
   </list>
   <speech>Which movie would you like to reserve?</speech>
   <timestamp>153421569</timestamp>
</output>
```

Figure 4.3: Example semantics for multimodal output presentation

Figure 4.3 contains a segment of XML-based semantic representation sent from the dialogue manager to the presentation planning module of a cinema ticket reservation system. As with the example frame in Figure 4.2, the semantics encodes the sending and receiving modules, only this time using the <*from*> and <*to*> XML tags. Additionally, in this example an <*id*> tag is used to delimit an identification number for the segment. The semantic representation contains information relating to two output modalities: text and speech. The <*text*> tag contains the text to be presented on screen. The <*list*> tag is used to identify the items to appear in a list on the display. Each item in the list is delimited by the <*item*> tag and within this tag are the <*title*> and <*no*> tags, which contain the title of the film and its order in the presented list. The <speech> tag contains text that the presentation planner can forward to a text-to-speech module. In this example, not all the information needed by the presentation planning module is contained in the semantic representation. For example, there is no information on the font size of the text, the colour of the background screen or the exact positioning of the films list. Obviously this information is important but, in this example, it is being obtained from another source by the *PresentationPlanner*. Semantic representations should only contain information that is strictly necessary to reduce the processing time and effort in the sending and receiving module and to minimise the strain on system resources. If information is already available in, for example, a domain model or semantic storage then it is not necessary to include this information in the semantics.

4.4. *Multimodal data fusion*

Multimodal data fusion requires several problems to be addressed including establishing criteria for fusing the information chunks, determining the abstraction level at which the fusion will be done and what to do if there is contradiction between the different information chunks. Often temporal information (timestamps) becomes important in the fusion process, e.g., to fuse the speech segment, “whose office is this?”, with the corresponding deictic gesture. As an example, consider the following dialogue between a user and an intelligent agent:

1 U: Whose office is this [→][2]*?*
2 S: That is Paul’s office.

The semantics of the speech input of turn 1 can be encoded in the segment of XML mark-up shown in Figure 4.4. The <*speech*> tag of the semantic representation shown in Figure 4.4 is used to delimit four tags containing information on the speech input: (1) the <*stype*> tag contains the speech type *query-partial* which tells the multimodal system that the speech is one

[2] [→] is used here to indicate a deictic gesture.

part of a multimodal query, (2) <*category*> contains the text *who* which gives more information on the meaning of the speech input, (3) the <*subject*> tag identifies the subject of the query and (4) <*stimestamp*> contains a timestamp for the speech segment.

```
<speech>
<stype>query-partial</stype>
<category>who</category>
<subject>office</subject>
<stimestamp>10345</stimestamp>
</speech>
```

Figure 4.4: XML semantic representation of "Whose office is this?"

The corresponding gesture input of turn 1 can be encoded, this time using a frame-based approach, as presented in Figure 4.5.

```
[GESTURE
GTYPE: pointing
COORDINATES: 1155, 2234
GTIMESTAMP: 10312]
```

Figure 4.5: Frame-based semantic representation of deictic gesture

Here, the information on the gesture input is marked up in the *GTYPE:*, *COORDINATES:* and *GTIMESTAMP:* slots. The timestamps are important so that the pointing gesture can be fused with the corresponding speech input. The value of *GTIMESTAMP:* would be particularly important if a gesture recognition module recognises another deictic gesture input several milliseconds after the first deictic gesture. The temporal information can then be used to discard the least likely gesture input or to assign probabilities to each of the two possible gesture hypotheses.

It is important to appreciate that multimodal input processing modules, e.g., for speech/images, may take different amounts of time to analyse various input data. This can mean that the marked up information will arrive in the wrong order. It is therefore common that timestamps are assigned to the individual multimodal information chunks. These timestamps can then be used to determine the exact order of several potentially corresponding inputs, to decide whether a separate information chunk corresponds to the current or different input and to discard input not relevant to the current situation, e.g., a third pointing gesture with the speech input, "check room availability and pricing at these two hotels". As an example, consider the XML semantic representation segments shown in Figure 4.6. The marked-up speech segment of Figure 4.6 (a) has been generated by a speech understanding component after analysing the utterance, "Please check room availability at these two hotels". The gesture recogniser has recognised three deictic gestures in close proximity to the speech input and the semantics of each of these is shown in Figure 4.6 (b) - (d) respectively. Clearly, one of these deictic gestures is not related to the spoken utterance represented in Figure 4.6 (a) and may be unintentional or related to a later utterance. In this example, the deictic gesture represented in Figure 4.6 (d) may be discarded since it was detected over four seconds after the deictic gesture marked-up in Figure 4.6 (c). The deictic gestures represented by Figure 4.6 (b) and (c) are both detected within 2 seconds of the speech input and are therefore deemed more likely to have been related to the speech utterance. The system could also use domain-specific information to discard the erroneous gesture if, for example, there is no hotel at or near the coordinates given in the semantics.

```
<speech>
  <type>booking-query</type>
  <category>hotel</category>

<subject>availability</subject>
  <noOfHotels>2</noOfHotels>
 <timestamp>101453232</timestamp>
</speech>
```

(a)

```
<gesture>
  <type>deictic</type>
  <coordinates>
      <x>700</x>
      <y>893</y>
  </coordinates>
 <timestamp>101453231</timestamp>
</gesture>
```

(b)

```
<gesture>
  <type>deictic</type>
  <coordinates>
      <x>1232</x>
      <y>543</y>
  </coordinates>
 <timestamp>101454561</timestamp>
</gesture>
```

(c)

```
<gesture>
  <type>deictic</type>
  <coordinates>
      <x>1454</x>
      <y>678</y>
  </coordinates>
 <timestamp>101458683</timestamp>
</gesture>
```

(d)

Figure 4.6: Semantic representations for ‘hotel availability’ example

As discussed in Chapter 2, Section 2.1, decisions on the synchronisation of modalities are often required at the output of a multimodal system where, for example, the system may need to synchronise the movement of a pointing laser with corresponding speech output. A decision may also need to be made on what is the best modality to use at the output, i.e., language or vision? For example, the directions from one office to another may be best presented visually using a laser, whilst a response to a user’s query may be better presented using speech output. Another example could be when the driver of a car asks an in-car intelligent information system for directions to the nearest petrol station. Here the system could respond by presenting a map to the driver or by dictating directions using speech output. The system response in this case would depend on whether or not the car was moving. That is, if the car is stopped in a lay-by the response could be given via the map. If however the car is moving, i.e., the driver’s eyes are pre-occupied on the road, then the system would respond using speech output.

Multimodal semantic fusion, as discussed in Chapter 2, Section 2.1, can be performed at a number of levels. Whilst the level of fusion that is necessary depends on the application, fusion is a key problem that must be addressed in multimodal decision-making. It is important that the correct level of fusion is chosen for a particular application. It would be pointless performing low level fusion of signals if this is not a requirement of the system. For example, if an intelligent space recognises simple commands such as, “turn the heating on”, “turn off the television”, “draw the curtains” and “dim the lights”, a low level analysis of the intonation of the speaker’s voice is not necessary. It would be equally unhelpful if high level semantic fusion was being applied when a high level interpretation is not important to the multimodal system. For, example, a high level interpretation of a user’s facial expressions and body language is not necessary if the system only needs to know the user’s head orientation and gaze direction within an intelligent space. It is often the case that best results are achieved when a combination of low level (signal) and high level (semantic) fusion is performed. That is, the first stage of the fusion process combines low level multimodal events such as speech and lip movement and the second stage of the fusion process extracts the high level meaning of the multimodal combinations.

4.5. *Multimodal ambiguity resolution*

Ambiguity does not necessarily always occur in multimodal systems but, when it does, it presents a difficult challenge that needs to be addressed. Where ambiguity occurs in one input modality, e.g. speech, information from other input modalities, e.g., gesture, eye-gaze, facial expression and touch, may be used to resolve the ambiguity. An example of ambiguity at the input could be when a user’s deictic gesture is accidentally logged as input. Consider the following example dialogue:

1 User: Show me the route from this office [→] to that [→] office.

2 User: [→]

3 System: This is the route from Sheila’s office to Tom’s office.

In this example, the user has pointed three times but has only referred to two offices. The third deictic gesture of turn 2 was unintentional and has been detected as input by the multimodal system. Here, synchronisation information in the semantic representation, e.g., timestamps, as discussed in Section 2.2, can be used to determine which two offices the user is referring to. The third deictic gesture can then be discarded if it has occurred considerably later than the second referent in the user’s utterance. Another example of input ambiguity is in an industrial environment where a control technician points at two computer consoles saying, “copy all files from the ‘process control’ folder of this computer to a new folder called ‘check data’ on that computer." In this example, synchronisation of the visual and audio input is needed to determine exactly which two computers the control technician is referring to. Ambiguity could also occur in an intelligent space or smart room when a person says, "turn that on". If there is more than one device in the room that can be turned on, ambiguity could arise in determining which device is the referent. Here, recognition of an accompanying deictic gesture could be used to determine which device the user is referring to. If no gesture input is received, then the system may need to ask the user to clarify which device he/she wants to turn on. Only three examples of ambiguity were given in this section, however there are many ways in which ambiguity can occur during decision-making in multimodal systems. Resolving ambiguity is thus a key problem for the decision-making component of a multimodal platform hub.

4.6. *Uncertainty*

Representing and dealing with uncertainty, as discussed in Chapter 2, Section 2.5.1, is a key problem in multimodal systems. Everyday decisions are seldom taken with 100% certainty that they are correct. During the course of a dialogue humans continuously make judgements about the mental state of other dialogue participants and anticipate the future actions of others. Decisions on when to speak, when to listen and where to look are taken all the time. Such decisions are never taken with absolute certainty. When humans make assumptions about the mental state of another person they adapt their dialogue strategy and plan future actions based on these beliefs. Additionally, when new information becomes available, people can dynamically adapt their dialogue strategy appropriately.

Given the uncertainty that frequently exists in multimodal dialogues between human users, it would be naive to assume that a multimodal system could take dialogue management decisions with absolute certainty. Regardless of how many multimodal inputs are considered, or how these inputs are weighted and analysed, there will always be a degree of uncertainty. Beliefs held by a multimodal system will often have confidence scores associated with them, which are subject to change if new evidence becomes available. The ability of Bayesian networks to perform intercausal reasoning enables the strengths of the beliefs in competing hypotheses to be reduced when new evidence is observed supporting a particular hypothesis. This is a desirable property for the decision-making component of a multimodal system, since it makes decision-making easier through the reduction of uncertainty. As an example, assume that the beliefs listed in Table 4.1 are held by an intelligent travel agent system and that, at this juncture in the multimodal dialogue, the intelligent travel agent system needs to narrow down the possible holiday destinations to recommend to the user. Also assume that the system can only select a certain category, e.g., hot destinations, if the confidence associated with the corresponding belief in Table 4.1 is greater than 65% and at least 20% greater than its competing hypothesis.

Hypotheses		Confidence
1.	User wants to book a holiday for two people	100%
2.	User wants a hot destination	53%
3.	Sunshine or heat is not important	47%

Table 4.1: Example hypotheses held by an ‘intelligent travel agent’ system

Next assume that input from the speech recognition, facial expression and gaze tracking modules causes the confidence associated with hypothesis 2 (user wants a hot destination) to rise from 53% to 66%. The system is still not in a position to decide to show holidays from the hot destination category since the belief in hypothesis 2 is not 20% greater than the competing hypothesis 3 (sunshine or heat is not important). However, an intelligent system should be able to determine that, if there is increased evidence that a user prefers a hot destination, then it is less likely that sunshine or heat is not important. It would be helpful if there was some mechanism that the intelligent travel agent system could use to lower the confidences of competing hypotheses when the belief in a certain hypothesis increases and vice versa. This exact capability is an inherent property of Bayesian networks, i.e., intercausal reasoning. If Bayesian networks were applied to decision-making in the intelligent travel agent system, obtaining evidence on one hypothesis would explain away competing hypotheses.

To conclude this example, assume now that Bayesian networks are being used in the decision-making component of the intelligent travel agent system. By performing intercausal reasoning, when the belief in hypothesis 2 is increased from 53% to 66%, the belief in hypothesis 3 is decreased from 47% to 34%. The system is now in a position to display more information on hot destinations, since the belief in hypothesis 2 is at least 20% greater than the belief in hypothesis 3. This is just one example of how the use of Bayesian networks and, in particular, their ability to perform intercausal reasoning has reduced the uncertainty in decision-making within a multimodal system. The probabilistic nature of Bayesian networks enable them to easily represent and dynamically adapt the beliefs associated with the semantics of multimodal data.

4.7. *Missing data*

Missing data is also a potential cause of ambiguity in multimodal decision-making. The decision-making mechanism must therefore be able to handle missing information. For example, if a multimodal system allows the user to move a file to the *Recycle Bin* using speech, hand gestures, facial expressions, touch and mouse input, then the user should be able to do this using just one modality, a combination of modalities, or all of the available modalities. The absence of one or more of these modalities should not create a problem. Equally the presence of all of these modalities should not make the decision more difficult. The aim in multimodal decision-making is always to reduce ambiguity using different modalities. Careful decision-making design is needed to ensure that ambiguity is reduced, not increased, by the presence of multiple modalities.

As an analogy, consider an investor who seeks financial advice as to whether or not he/she should buy shares in a company in times of economic uncertainty. If the investor goes to just one financial advisor, then the decision may be easier to make. However, the decision being easier is no guarantee that the decision will be correct. Conversely, if the investor goes to five different financial advisors with each making recommendations with varying degrees of certainty, the decision becomes more complex. It is arguable, however, that the latter option is better since the multiple inputs to the decision allow for a more balanced, intelligent decision to be made. The same is true for decision-making in multimodal systems. The presence of multiple modalities can make the decision more complex but, by considering all of the available modalities, the system can come to a more intelligent conclusion. In order to ensure that ambiguity is reduced, and not increased, the decision-making mechanism must be able to assign appropriate weighting to the relevance of each modality and dynamically adjust the weighting at run-time. Consider an intelligent car safety system that monitors the posture, head position, eye-gaze and facial expression of a driver with the aim of warning the driver should he/she show signs of tiredness. Table 4.2 presents some of the beliefs held by the system:

Hypotheses		Confidence
1.	Driver is tired based on posture recognition	23%
2.	Driver is not tired based on posture recognition	77%
3.	Driver is tired based on head tracking	71%
4.	Driver is not tired based on head tracking	29%
5.	Driver is tired based on eye-gaze tracking	67%
6.	Driver is not tired based on eye-gaze tracking	33%
7.	Driver is tired based on facial expression	12%
8.	Driver is not tired based on facial expression	88%

Table 4.2: Example hypotheses held by an ‘intelligent car safety’ system

Here, if we are to assume that a hypothesis with a confidence greater than 65% is deemed true, the following four hypotheses are all true:

- Driver is not tired based on posture recognition.
- Driver is tired based on head tracking.
- Driver is tired based on eye-gaze tracking.

- Driver is not tired based on facial expression.

We now have two overall competing beliefs held by the system: (1) the driver is tired and (2) the driver is not tired. The intelligent car safety system now needs some way of deciding whether or not the driver is actually tired. What is necessary in this example is some means of weighting the significance of the posture, head, eye-gaze and face recognition modules. This can easily be done using a conditional probability table (CPT) of a Bayesian network. The overall belief in a driver being tired or not could be represented by a single node in the network, e.g. called *DriverTired,* that is influenced by *Posture*, *Head*, *Eye-gaze* and *FacialExpression* nodes. The CPT of the *DriverTired* node would appropriately weight the inputs to ensure that an intelligent conclusion could be reached as to the tiredness of the driver.

To continue this example further, let's assume that there is no input to the *FacialExpression* node because glare from the sun has distorted the system's recognition of the driver's facial expressions. Now assume that the intelligent car safety system implements a rigid rule-based method of decision-making and uses the following rule to decide if the driver is tired:

IF the belief that the driver is tired based on posture recognition is greater than 55%
AND the belief that the driver is tired based on head tracking is greater than 55%
AND the belief that the driver is tired based on eye-gaze tracking is greater than 50%
AND the belief that the driver is tired based on facial expression is greater than 70%
THEN the driver is tired

Here, the absence of the facial expression input will mean that the decision on the driver's tiredness cannot be made. Of course, the previous rule could easily be adapted to make the facial expression input optional but this would reduce the intelligence of the system. The inclusion of the semantics of facial expressions in the rule suggests it is important and therefore excluding it from the decision, under any circumstances, is not ideal and would only serve to reduce the accuracy of the system. A better approach would be to implement a Bayesian network that considers all available inputs at all times in the decision-making process and, where evidence is observed to support or disconfirm a particular hypothesis, adjust the beliefs of that hypothesis accordingly, i.e., update the values of the states on that node. Where no evidence is observed to support a particular hypothesis, as is the case in the example above, the system does not update the belief in that hypothesis but continues to recognise its, albeit limited, influence within the Bayesian network and on the decision as to the tiredness of the driver. Missing data can be handled by a multimodal system using Bayesian networks for decision-making. Where evidence is observed on the node of a Bayesian network, all nodes in the network are updated. It is not an essential requirement that all, or indeed any, nodes of a Bayesian network are updated before a conclusion can be reached.

4.8. *Aids to decision-making in multimodal systems*

This section considers features of a multimodal system that aid the decision-making process. This includes a discussion on distributed processing, dialogue history, context knowledge, domain information and learning.

4.8.1. Distributed processing

Decision-making in any situation often requires the decision-maker to process information from a variety of sources arriving at different times. This is particularly true in an intelligent multimodal system which needs to process information from various input modalities, e.g., speech recognition, face recognition, gesture recognition and haptic modules. The multimodal information from the different sources will invariably arrive at different times, i.e., haptic input via a touch-screen will arrive before speech input. It is therefore important that the multimodal system has mechanisms in place to deal with distributed processing. For example, consider a cinema ticket reservation system. Suppose that the system enables user input using speech, eye-gaze and mouse input. The system uses the eye-gaze input to aid decision-making where mouse input is not detected and ambiguity or uncertainty arises in the understanding of the speech input. Assume that the speech input is processed in a speech recognition module running on a medium specification Linux machine, whilst a much faster, more powerful Windows computer is used to host the gaze-tracking module. The processing of mouse input, where present, is conducted on the local Windows PC, which is of relatively low specification in comparison to the other two computers. The remaining modules of the system are also running on the local PC. Hence, three separate computers, all with different hardware specifications, are used to implement the cinema ticket reservation system.

In this example, both Windows PCs are present in the same building, whilst the Linux machine is located in another building. It should be obvious to the reader why the ability to perform distributed processing is an essential requirement of the cinema ticket reservation system. Because the system is distributed across three machines and two buildings, there needs to be some mechanism in place to process the inputs from both the speech recognition and gaze-tracking modules as they arrive in the main application on the local PC. The distributed nature of the system discussed in this example would also leave timestamps, as discussed in Section 2.2, important to the correct interpretation of the different inputs. The varying processing speeds of the three computers and the time taken to process the different multimodal inputs will mean that the inputs from the recognition modules will all arrive at different times and not necessarily in the correct order. It would therefore be important to know the exact time that each input was detected. It should also be noted that distributed processing can be advantageous, and often a requirement, for a multimodal system with its modules running on a single machine.

To continue this example further, assume that during the development stage the cinema ticket reservation system is distributed across seven computers, again all with different hardware specifications. There are now three speech recognition modules and three gaze-tracking modules and each of these recognition modules is running on a separate machine. The remaining modules of the system are located on the local PC. The three speech recognition modules are each running a different speech recognition algorithm and are being monitored for speed and accuracy. The speed and accuracy of the gaze-tracking modules are also being monitored. The purpose of the current phase of development is to determine which speech recognition and gaze-tracking modules to implement in the final version of the cinema ticket reservation system. Here, not only temporal information, but also the source of the information and the confidence associated with the recognition results needs to be captured in order that the fastest and most accurate recognition modules can be identified. All this information can be contained in the semantic representation sent from the recognition modules. A possible frame-based semantic representation for the speech recognition information is shown in Figure 4.7, whilst Figure 4.8 gives an XML segment that represents the semantics of the gaze input.

```
[SPEECH
FROM: SpeechRecogniser2
INPUT TYPE: speech
INTENTION: film_selection
HYPOTHESIS1 [
SPEECH: “the first film”
CONFIDENCE: 76.76%
]
HYPOTHESIS2 [
SPEECH: “the third film”
CONFIDENCE: 23.24%
]
TIMESTAMP: 0112374323
]
```

Figure 4.7: Frame-based semantic representation of speech recognition result

```
<eye-gaze>
<from>GazeRecogniser3</from>
<inputType>film_selection</inputType>
<coordinates>1234,900 </coordinates>
<timestamp>0112301234</timestamp>
</eye-gaze>
```

Figure 4.8: XML-based semantic representation of gaze input semantics

The semantic frame in Figure 4.7 has two slots, *HYPOTHESIS1:* and *HYPOTHESIS2:*, that represent the beliefs that the user uttered, "the first film", and, "the third film", respectively. Since the gaze of the user will continuously change as he/she is speaking, the exact timing of the eye-gaze input in relation to the speech input is crucial in this example. Another piece of information that may be beneficial is the amount of time the user's gaze is fixed on a particular part of the screen. This information could be contained in a *<duration>* tag and could be used to identify, and where appropriate discard, short and long eye-gaze fixations. The coordinates of the corresponding eye-gaze input are also marked up in the XML segment in Figure 4.8. For the purposes of identifying the most effective speech recognition and eye-gaze modules, a *FROM:* slot is included in Figure 4.7 and a *<from>* tag is contained in the semantics in Figure 4.8.

4.8.2. Dialogue history, context and domain information

During the course of a dialogue, humans automatically keep track of what they and the other dialogue participants say and do. They then use this information to dictate their future speech, actions and dialogue acts. It is also important that context-specific information is maintained by the multimodal system. The system needs to have the capability of behaving differently depending on the current context. For example, an in-car multimodal presentation system may need to switch from video to text output if bandwidth becomes limited and the car is not moving or the system may opt to use speech output only when a car is moving. Sometimes a change in context can reduce or increase the significance of a multimodal input. For example, if the environment suddenly becomes noisy, speech input may be less important and its accompanying visual input, e.g. lip movement, may become more important. Depending on the complexity of the decision-making domain a multimodal system may need to handle one, some or many different contexts. For example, numerous different contexts need to be considered to handle turn-taking between an intelligent agent and a human user (e.g. *GiveTurn, TakeTurn, Speaking, Listening*), whilst only two contexts might suffice in an intelligent in-car information presentation system (e.g. *CarMoving, CarStopped*). In order to perform intelligent context-aware decision-making, multimodal systems need to constantly monitor the current context and dynamically adapt its behaviour based on the changing context.

4.8.3. Learning

The intelligence of a multimodal system can be greatly enhanced if it has the ability to learn from past experience. It is impossible for humans to prepare themselves for every eventuality in real life situations and dialogues. It is equally impossible for decision engineers to design a dialogue strategy that will be prepared for every possible combination of multimodal input. Whilst people cannot prepare themselves for every situation they will face, they do have the ability to learn from past experiences so that they may know what to do if they encounter a similar situation in the future. Therefore, whilst the ability to learn from data and past experience is not an essential requirement of a multimodal system it does allow for more intelligent human-like decision-making. Bayesian networks, as discussed in Chapter 3, Section 3.11.5, can support both structural learning and parametric learning. Structural learning learns dependencies between variables in a set of data. This can be useful if data has been collected for a particular application domain which contains typical outputs, decisions or conclusions for certain combination of inputs or evidence. As an example, suppose 100 users are monitored interacting with a multimodal system in a Wizard-of-Oz[3] experiment and that the data is collected and stored in a case file. The case file contains the states of five inputs (*A*, *B*, *C*, *D*, and *E*) and the corresponding conclusions (*X*, *Y* and *Z*). Assume that the inputs *A*-*E* are captured by the following multimodal processing modules:

- *A* - Speech recognition module.
- *B* - Lip reading module.
- *C* - Facial expression recognition module that tries to ascertain users' intention based on their facial expressions.
- *D* - eye-gaze tracking module that detects where the user what part of a computer screen the user is looking at.
- *E* - Posture recogniser that monitors posture and body language of the user and makes judgments on the user's emotional state.

X, *Y*, and *Z* in this example are variable each with a number of states that represent conclusions on the intentional state of the user. Now, suppose we have a data file populated during the Wizard-of-Oz experiment, a segment of which is shown in Figure 4.9. Structural learning can take the data file shown in Figure 4.9 and learn the structure of a Bayesian network that represents the causal dependencies implicit in the data. Parametric learning is used to learn the parameters, or Conditional Probability Tables (CPTs), of a Bayesian network. This can involve the adaptation of an existing Bayesian network to a new data set (adaptive learning) or generating the CPTs of a Bayesian network from a database, i.e., Estimation-Maximum (EM) learning.

[3] A Wizard-of-Oz experiment is one where a person, i.e., the wizard, simulates the behaviour of an intelligent system.

```
A, B, C, D, E, X, Y, Z
TRUE, FALSE, command, A2, neutral, TRUE, FALSE, FALSE
TRUE, TRUE, question, B3, happy, FALSE, FALSE, TRUE
FALSE, TRUE, comment, C24, happy, FALSE, TRUE, TRUE
TRUE, FALSE, comment, A18, angry, TRUE, TRUE, FALSE
TRUE, FALSE, question, D11, sad, TRUE, FALSE, FALSE
FALSE, TRUE, question, A17, neutral, TRUE, TRUE, TRUE
FALSE, TRUE, undefined, A5, neutral, TRUE, FALSE, FALSE
TRUE, FALSE, comment, G9, happy, FALSE, FALSE, FALSE
TRUE, TRUE, comment, L3, frustrated, TRUE, FALSE, TRUE
FALSE, TRUE, question, L6, neutral, TRUE, FALSE, FALSE
TRUE, FALSE, undefined, L1, happy, FALSE, TRUE, TRUE
FALSE, FALSE, undefined, X23, happy, FALSE, FALSE, FALSE
TRUE, TRUE, comment, Y24, neutral, TRUE, FALSE, FALSE
FALSE, FALSE, question, X23, neutral, FALSE, TRUE, TRUE
TRUE, TRUE, question, C24, frustrated, FALSE, FALSE, FALSE
TRUE, FALSE, undefined, S21, happy, TRUE, FALSE, FALSE
FALSE, TRUE, comment, Z23, frustrated, TRUE, TRUE, TRUE
```

Figure 4.9: Segment of data file for structural learning of a Bayesian network

4.9. *Key example problems in multimodal decision-making*

This section discusses multimodal processing example problems that highlight the benefits of a Bayesian approach to decision-making within multimodal systems.

4.9.1. Anaphora resolution

Consider the following dialogue:

1 A: Can you tell me how to get to Mary's office?

2 B: Yes, go down that [→] corridor and take the 3rd door on the left.

3 A: And how do I get from her office to the school office?

4 B: The school office is directly opposite Mary's office. The one with the red door. But it is currently closed for lunch and will not be open until 2:00pm.

Here, because *B* has kept a record of dialogue history, he/she knows that 'her' in turn 3 refers to Mary. This kind of decision-making is easy for humans but, in order to replicate this in a computer, a multimodal system needs mechanisms in place to keep track of dialogue history. If we were to replace *B* in the previous dialogue with an intelligent agent, we would need to ensure that a dialogue history is maintained that keeps track of the name and gender of the last person mentioned. We would also need domain-specific information such as the current position of *A*, the location of Mary's office and the school office, the colour of the school office door, the current time and the opening hours of the school office. This simple example gives an indication of the amount of domain-specific information and dialogue history that must be maintained to support decision-making in a multimodal system. The multimodal dialogue history needs to be stored as fused semantic representations, often on a blackboard or whiteboard, within the multimodal system. As the complexity and scope of the dialogue increases, so too does the amount of dialogue information that must be maintained.

4.9.2. Domain knowledge awareness

Often partial frames or information chunks are encoded by a multimodal system during the course of a dialogue. Consider an intelligent bus ticket reservation system. Assume that the semantics of Figure 4.10 is created in response to the following utterance:

1 User: I want to book a bus from Dromore to Dublin on Sunday 21st of September.

```
[BookingPartial
FROM: SpeechRecogniser
TO: DialogueManager
INPUT TYPE: Speech
INTENTION: BookingRequest
DEPART: Dromore
DESTINATION: Dublin
DATE: 21/09/08
TIME:
TIMESTAMP: 011237432
]
```

Figure 4.10: Partial frame for intelligent bus ticket reservation system

Note that the frame in Figure 4.10 is only partially complete. The *TIME:* slot is empty since the user did not mention a departure time. Also assume that, after querying the domain model to check the correctness of the recognised source and departure locations, the system realises that there are several different places called Dromore. Turns 2 to 4 below are then necessary to resolve this ambiguity by asking the user which Dromore he/she is referring to and to confirm the departure time.

2 System: Which Dromore are you departing from?

3 User: Dromore, County Tyrone.

4 System: And, at what time would you like to leave?

5 User: The first bus in the morning.

After turn 3 the semantics of the input will be interpreted by the system and the domain model will be queried to check that there is a place called Dromore in County Tyrone that is serviced by the bus. The system will also need to query the domain model after analysing the semantics of turn 5 to determine the earliest bus from Dromore to Dublin on the date specified. After the disambiguation steps taken in turns 2 to 5, the semantic frame in Figure 4.10 will be updated with the correct information. Essentially this step will involve the fusion of several semantic frames containing the required information to book the bus ticket. In this example, domain-specific information is crucial to resolving the ambiguity present in the dialogue. Careful consideration is needed in the design of a multimodal platform to ensure that it can fully utilise the valuable information contained in the semantic representation. The hub of a multimodal platform must process the semantic content from the various sources of multimodal input, where appropriate route the semantics to other system modules, generate semantics for multimodal output and coordinate multimodal presentation.

4.9.3. Multimodal presentation

Consider a multimedia presentation system for monitoring the driver of a car for signs of tiredness. Assume that the semantics of the following modalities are available to support the decision-making:

- Facial expression
- Eye gaze
- Head movement
- Posture

The system also considers driver behaviour, i.e., steering and braking. A Bayesian network for this example is presented in Figure 4.11.

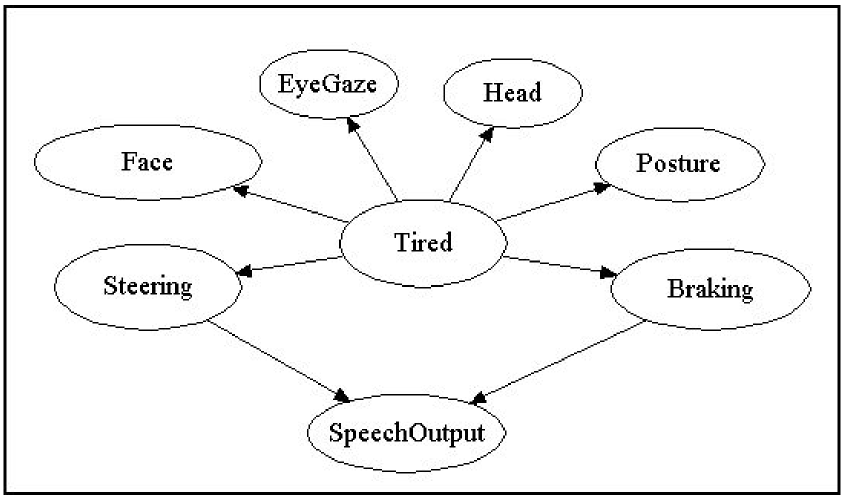


Figure 4.11: Bayesian network for multimodal presentation

As shown in Figure 4.11, four nodes represent the beliefs relating to the multimodal inputs, i.e., *Face*, *EyeGaze*, *Head* and *Posture*. The *Steering* and *Braking* nodes represent driver behaviour.

CPTs elicit the parameters of the Bayesian network. An example CPT for the *Face* node is given in Table 4.3, whilst Table 4.4 shows a CPT for the *SpeechOutput* node. As illustrated by Table 4.4, the *SpeechOutput* node recommends simulated speech output based on the driver's behaviour, represented by the *Steering* and *Braking* nodes.

Face		
Tired	OK	Tired
Tired	0.5	0.5
Normal	0.5	0.5

Table 4.3: CPT of *Face* node

SpeechOutput				
Braking	Normal		Abrupt	
Steering	Normal	Abrupt	Normal	Abrupt
None	0.3333	0.3333	0.3333	0.3333
FancyBreak?	0.3333	0.3333	0.3333	0.3333
Warning	0.3333	0.3333	0.3333	0.3333

Table 4.4: CPT of *SpeechOutput* node

4.9.4. Turn-taking

Consider the problem of coordination of turn-taking in an intelligent agent. In this example, the multimodal system processes the semantics of speech, gaze and posture multimodal inputs. Figure 4.12 presents a Bayesian network for this example.

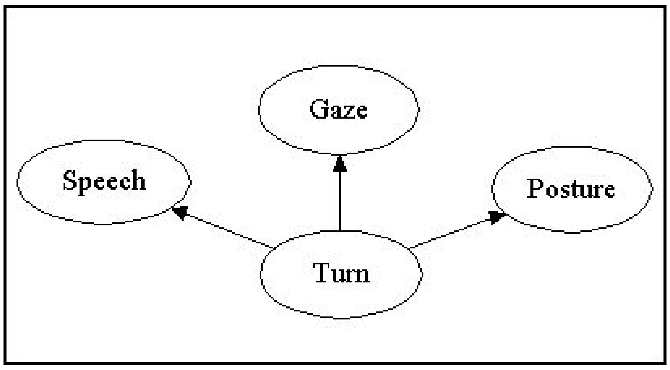


Figure 4.12: Bayesian network for turn-taking

As indicated by the directed edges in the Bayesian network, the *Turn* node has influence over the *Speech*, *Gaze* and *Posture* nodes. Each of the nodes in the Bayesian network in Figure 4.12 has the states *Give* and *Take* as shown in the CPTs for the Bayesian network given in Tables 4.5 - 4.8.

Speech		
Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

Table 4.5: CPT of *Speech* node

Gaze		
Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

Table 4.6: CPT of *Gaze* node

Posture		
Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

Table 4.7: CPT of *Posture* node

Turn	
Give	0.5
Take	0.5

Table 4.8: CPT of *Turn* node

In the *Speech*, *Gaze* and *Posture* nodes the states *Give* and *Take* represent the belief that the user wishes to give or take the turn. The *Give* and *Take* states of the *Turn* node recommend whether or not the intelligent agent should give or take the next turn in the dialogue.

4.9.5. Dialogue act recognition

Consider a multimodal system that supports the decision-making of an intelligent agent through dialogue act recognition. The system considers the semantics associated with speech, voice intonation and the recognition of eyebrow and mouth movements, before making decisions on the dialogue acts being performed by the user. A Bayesian network for this example is shown in Figure 4.13. The directed edges between the nodes of the Bayesian network in Figure 4.13 indicated that the *DialogueAct* node has influence over the *Speech*, *Intonation*, *Eyebrows* and *Mouth* nodes. This influence is also evident in the CPTs for the Bayesian network. Tables 4.9 – 4.11 gives the CPTs for the *Intonation*, *Eyebrows* and *Mouth* nodes. Tables 4.12 and 4.13 give the CPTs for the *Speech* and *DialogueAct* nodes respectively.

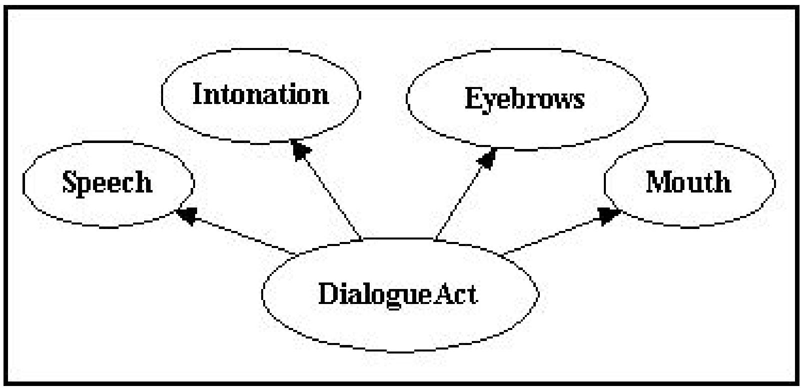


Figure 4.13: Bayesian network for dialogue act recognition

Intonation		
Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

Table 4.9: CPT of *Intonation* node

Eyebrows		
Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

Table 4.10: CPT of *Eyebrows* node

Mouth		
Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

Table 4.11: CPT of *Mouth* node

Speech					
DialogueAct	Greeting	Comment	Request	Accept	Reject
Greeting	0.80	0.05	0.05	0.05	0.05
Comment	0.05	0.80	0.05	0.05	0.05
Request	0.05	0.05	0.80	0.05	0.05
Accept	0.05	0.05	0.05	0.80	0.05
Reject	0.05	0.05	0.05	0.05	0.80

Table 4.12: CPT of *Speech* node

DialogueAct	
Greeting	0.2
Comment	0.2
Request	0.2
Accept	0.2
Reject	0.2

Table 4.13: CPT of *DialogueAct* node

4.9.6. Parametric learning

Suppose the data file shown in Figure 4.14 has been used to learn the structure of a Bayesian network. However, following analysis of the Bayesian network, it is felt that its performance could be improved if a larger set of data is considered in the learning process. A Wizard-of-Oz experiment is conducted monitoring 1000 users interacting with the system.

```
A, B, C, D, ES
happy, neutral, relaxed, happy, happy
neutral, neutral, happy, relaxed, neutral
open, happy, happy, neutral, happy
defensive, open, confused, happy, neutral, confused
defensive, defensive, confused, defensive, defensive
open, neutral, happy closed, open
happy, neutral, relaxed, happy, happy
confused, relaxed, neutral, neutral, neutral
happy, relaxed, neutral, happy, neutral
happy, happy, happy, relaxed, happy
happy, relaxed, relaxed, happy, neutral
open, happy, happy, neutral, happy
defensive, happy, confused, defensive, defensive
open, open, neutral, neutral, neutral
happy, happy, relaxed, happy, happy
open, closed, open, closed, closed
happy, happy, happy, relaxed, happy
relaxed, neutral, neutral, relaxed, relaxed
```

Figure 4.14: Segment of data file for structural learning of a Bayesian network

Parametric (adaptive) learning is performed to learn the parameters, i.e., the CPTs, of the Bayesian network. As a result, the existing Bayesian network is adapted to the new, much larger, data set. This adapted Bayesian network can now be analysed to determine if the increase in learning data has improved the accuracy of its conclusions. Parametric learning was discussed in greater detail in Chapter 3, Section 3.11.5.

This section has discussed six key example problems in multimodal decision-making, including anaphora resolution, domain knowledge awareness, multimodal presentation, turn-taking, dialogue act recognition and parametric learning.

4.10. *Requirements criteria for a multimodal distributed platform hub*

Having considered key problems in decision-making within a multimodal system, a set of necessary and sufficient criteria for the decision-making mechanism in a multimodal hub can now be drafted. These criteria list the core requirements for the hub of a multimodal distributed platform. The criteria are categorised into the following two categories:

- Essential criteria
- Desirable criteria

Essential criteria (denoted by E) must be met in order that the hub is capable of performing and/or coordinating the type of decision-making commonly required within a multimodal system. Desirable criteria (denoted by D) are not essential but would enhance the effectiveness of the decision-making mechanism. Essential criteria for a multimodal distributed platform hub are summarised in Table 4.14.

4.11. *Bayesian decision-making in multimodal fusion and synchronisation*

In this section the rationale for a Bayesian approach to decision-making within a multimodal distributed platform hub is detailed and how this approach addresses a number of key problems in multimodal decision-making discussed.

4.11.1. Rationale

There are a number of properties of Bayesian networks that leave them particularly suited to decision-making over multimodal data. First, intercausal reasoning, or the explaining away effect, can greatly simplify decision-making in multimodal systems by disconfirming, or explaining away, other hypotheses in the light of new evidence supporting a particular hypothesis. As discussed in Chapter 3, Section 3.3, intercausal reasoning is an intrinsic property of Bayesian networks. An example of intercausal reasoning is where evidence supporting the hypothesis that a person wants to take the next dialogue turn decreases the belief in the competing hypothesis that the person wants to give the turn to another dialogue participant, i.e., the competing hypothesis is explained away. Another example is where a multimodal 'building data' system detects three deictic gestures in close proximity to a user utterance, "show me route from that office to this office". If timestamp information increases the belief that the user intentionally referred to two particular offices using the first two deictic gestures, then the belief that the third deictic gesture was intentional will subsequently decrease. The ability to automatically perform intercausal inference is a key contributor to the reasoning power of Bayesian networks.

Criterion	**Capability**
E1	The decision-making mechanism must be able to operate over semantic representations of both multimodal input and output.
E2	The hub must be able to fuse semantics at both input and output of a multimodal system.
E3	There should be, “no presentation without representation” (Wahlster 2003, p. 12).
E4	The decision-making mechanism should be able to dynamically update the beliefs associated with multimodal input and output at run-time.
E5	The hub should be capable of distributed processing in recognition of the inherently distributed nature of multimodal systems.
E6	Multimodal dialogue history should be stored for use in decision-making.
E7	The decision-making process should consider the current context when making decisions.
E8	The decision-making mechanism should be capable of resolving ambiguity in one modality using information from other modalities.
E9	Domain-specific information should be available to enable intelligent interaction with human users.
E10	Missing data should not create a problem for the decision-making process.
E11	The decision-making mechanism must be able to make decisions on the optimum combination of output modalities in a multimodal system.
E12	It should be possible to learn a decision-making strategy based on sample data for a particular problem domain.
D1	The hub should operate as multi-platform.
D2	The hub should be able to learn and adapt the decision-making based on previous experience.
D3	The decision-making mechanism should have the ability to learn from real data.

Table 4.14: Requirements criteria for a multimodal distributed platform hub

Second, as discussed in Chapter 3, Section 3.3, the compact graphical nature of Bayesian networks is advantageous whilst attempting to model a large and complex multimodal decision-making domain consisting of many random variables. As an example, consider the case where there are several discrete random variables representing the probabilities of beliefs associated with various multimodal inputs. Here, if we were to specify the joint probability distribution, its size would grow exponentially with the number of variables, i.e., one probability would be needed for every possible configuration of the variables. Bayesian network provide a compact representation of such a complex domain by using a graphical structure to encode dependence and independence relations between the random variables.

Third, there are inherent cause-effect relationships in multimodal decision-making. For example, if a person is observed shaking his/her head, then this causes us to believe that the person disagrees with what is being said, whilst facial expressions can influence our belief about a person's mental state. Similarly, our knowledge of past events and dialogue history may cause us to adapt our future actions and dialogue strategy. In order to engage in natural human-like communication, the ability to model causation in multimodal systems is desirable. Bayesian networks can explicitly represent cause-effect relationships within any decision-making domain. Furthermore, Bayesian networks are an intuitive graphical means of representing causality within a domain. As discussed in Chapter 3, Section 3.1, humans frequently consider causation in their everyday lives and this is evident in the choice of words humans use in situations where uncertainty exists. Phrases such as, "John will be late for the meeting because of the harsh driving conditions", "if Mary does not call today, then she must be satisfied that the issue is resolved", and, "there was definitely someone at home since the lights and TV were on", are all examples of causation being used in speech under uncertain conditions, i.e., the speaker cannot be certain that John will be late, that Mary's issue is resolved or that there was anyone at home. Hence, causation is a phenomenon that humans deal with frequently during the course of a dialogue. It is therefore appropriate that Bayesian networks be used to model the cause-effect relations that arise in multimodal decision-making. The fact that causation sits easily with people's reasoning processes simplifies the construction of Bayesian networks that model the causal dependencies between variables of a problem domain.

Fourth, decision-making within multimodal systems frequently involves the resolution of uncertainty and ambiguity. The interpretation of multimodal input and the weightings assigned to multimodal output are most naturally handled using confidence or probability scores. The careful weighting of all available inputs enables Bayesian networks to deal with the complexity of decision-making within multimodal systems. The more modalities that are considered, the more complex the decision-making becomes. In order that one or more modality may be used to resolve ambiguity and uncertainty arising in another modality, a flexible and intuitive means of representing the beliefs associated with modalities is needed. It is difficult for people to make absolute certain judgements about the emotional states of others, just as it may be difficult to be 100% certain that a person has pointed to a particular office and not an adjacent office. Even when humans are almost completely certain about something, they are reluctant to express certainty. For example, we frequently choose to say we are, “nearly sure”, or, “almost certain”, or, “99.9% certain”. Where uncertainty is present, however small the uncertainty may be, it is important that it is represented. Probabilities, i.e., percentages, are an intuitive means of representing uncertainty. As discussed in Section 4.6, Bayesian networks are proficient at dealing with the beliefs assigned to various multimodal inputs. Furthermore, the probabilistic nature of Bayesian networks renders them useful for representing competing hypotheses on the semantics of multimodal input. For example, a speech recogniser may believe a user has said, “the first film”, with a probability of 46%, “the third film”, with a probability of 32%, and, “the fourth film”, with a probability of 22%. These competing hypotheses can be easily represented in a Bayesian network, which can use additional multimodal information, e.g., mouse or eye-gaze input, to overcome the uncertainty regarding the user’s intention.

Fifth, missing information does not create a problem for a Bayesian network. There is no requirement to update all, or indeed any, nodes in a Bayesian network. Acquiring more information on the variables of a problem domain does lead to more intelligent decision-making, but missing data will not prevent the Bayesian network from running and reaching a conclusion. Missing data is common in multimodal systems, since often the multimodal inputs are optional. It is also possible that certain inputs may only be considered if there is uncertainty or ambiguity present. For example, consider a multimodal system for downloading music from the Web. If the speech recognition module believes with a high degree of certainty that the user has said, “download the first song in the list”, and there are no competing hypotheses with a confidence score above a certain threshold, then the system may not consider eye-gaze or mouse input. Here, the only data, or evidence, applied to the Bayesian network would be that relating to the speech input. Of course, there would still be nodes relating to the eye-gaze and mouse input but, in the absence of any evidence on these nodes, they would have minimal influence on the conclusions reached by the Bayesian network.

Finally, Bayesian networks possess the ability to learn and update their conditional probability tables based on previous experience. The conditional probability tables (CPTs), used to specify the quantitative part of a Bayesian network are updated dynamically at run-time when new evidence is propagated through the network. Additionally, both structural and parametric learning can derive or refine a Bayesian network from a data set. The ability to learn from data is particularly advantageous when attempting to develop Bayesian networks to model the causal relationships between variables of a new decision-making domain. If data has been collected for a new application domain a Bayesian network can learn the cause-effect relationships between the variables in the data. The learning capability of Bayesian networks was discussed in Chapter 3, Section 3.11.5.

To summarise, Bayesian networks are deemed particularly suited to multimodal decision-making for the following reasons:

- They can automatically perform intercausal reasoning which is advantageous when modelling complex multimodal problem domains.
- They constitute a compact, intuitive means of representing large and complex decision-making domains.
- Their graphical structure is an intuitive way to represent the cause-effect relations that are inherently present in multimodal decision-making.
- Probabilities, and hence Bayesian networks, provide a flexible and intuitive means of representing uncertainty and ambiguity, thereby meeting the essential criteria E4 and E8 in Table 4.14.
- Missing data does not create a problem. There is no requirement to add evidence on the node of a Bayesian network in order that the network can be run and produce useful conclusions (criterion E10 in Table 4.14).
- Their ability to learn from past experience and data. Bayesian networks dynamically adapt their CPTs at run-time as new evidence is propagated through the network. Bayesian networks can also learn from data through, for example, structural and parametric learning (desirable criterion D2 in Table 4.14).

4.12. *Summary*

This chapter presented a Bayesian approach to decision-making within a multimodal distributed platform hub. Key problems within multimodal systems were highlighted before the characteristics of multimodal decision-making were discussed. Distributed processing, dialogue history, context/domain-specific information and learning where considered with regard to their role in aiding multimodal decision-making. Essential and desirable criteria for a multimodal distributed platform hub were then presented. Finally, the motivation and advantages of applying Bayesian networks to multimodal decision-making were discussed. In summary, this chapter presented the thesis that Bayesian networks fulfil the requirements associated with decision-making over multimodal data within a multimodal distributed platform hub. The next chapter discusses the implementation of a multimodal distributed platform hub called MediaHub.

Chapter 5 Implementation of MediaHub

This chapter discusses the implementation of MediaHub, a multimodal distributed platform hub for Bayesian decision-making over multimodal input/output data. First, we present the architecture of MediaHub and then its key modules are discussed in detail. A discussion follows on semantic representation and storage, before Psyclone (Thórisson et al. 2005), which facilitates distributed processing in MediaHub, is described. Next, five decision-making layers in MediaHub are outlined: (1) *psySpec* and contexts, (2) message types, (3) document type definitions (DTDs), (4) Bayesian networks and (5) rule-based. The role of Hugin (Jensen 1996) in implementing Bayesian networks for decision-making in MediaHub is then discussed. Multimodal decision-making in MediaHub is then demonstrated through six worked examples investigating key problems in various application domains.

5.1. *Constructionist Design Methodology*

The Constructionist Design Methodology (CDM) (Thórisson et al. 2004), discussed in Chapter 2, Section 2.7.11, was used in designing MediaHub. As the development of MediaHub did not involve a large team not all aspects of CDM were directly relevant. The key steps of CDM that were particularly relevant are listed below:

1. Define the project's goal, i.e., implement Bayesian decision-making in a multimodal distributed platform hub.
2. Define the project's scope, i.e., the key problems and application domains discussed in Chapter 4.
3. Modularisation – MediaHub is constructed using modules that communicate through *MediaHub Whiteboard*.
4. Test the system against scenarios, i.e., MediaHub is tested against a number of decision-making scenarios that illustrate its capabilities in multimodal decision-making.
5. Iterate – Steps 2 to 4 were repeated until the desired functionality was achieved.
6. Early testing of system modules – all MediaHub modules were tested at an early stage in their implementation.

7. Build all modules to their full specification – all MediaHub modules were iteratively developed to full specification.
8. Tune the system – MediaHub was then tested with all its modules running.

The step that was not relevant was step 6 in Chapter 2, Section 2.7.11, 'Assign modules to suitable team members (based on their strengths and areas of interest)'. This step was not necessary since MediaHub was developed by a single researcher.

5.2. *Architecture of MediaHub*

MediaHub, developed in the Java programming language, takes as input marked up multimodal data in XML format. These XML segments represent potential output of recognition modules, e.g., speech, haptic, gaze and facial expression. Figure 5.1 shows the architecture of MediaHub, consisting of the following key modules:

- Dialogue Manager
- MediaHub Whiteboard
- Decision-Making Module
- Domain Model
- MediaHub psySpec

MediaHub's architecture closely resembles the generic architecture of a multimodal distributed platform hub given in Figure 4.1, Chapter 4. As shown in Figure 5.1, MediaHub utilises Psyclone for distributed processing and tracking the current context. Psyclone, discussed in Chapter 2, Section 2.7.10, is a message-based middleware that enables large distributed systems to be developed. Bayesian decision-making is performed by the Hugin decision engine, discussed in Chapter 3, Section 3.11.5, which is accessed through a Hugin API (Hugin 2009). Input/output recognition modules are not implemented, only the XML representation of the input/output is generated/interpreted. Some additional processing is conducted for testing purposes, e.g., a terminal window displays the coordinates of recognised offices, names of recognised individuals and coordinates for laser output.

5.2.1. Dialogue Manager

The *Dialogue Manager*, in conjunction with *MediaHub Whiteboard*, coordinates the following: (1) interaction between MediaHub and other system modules, (2) fusion and synchronisation of multimodal input/output and (3) communication between the modules of MediaHub. Each of these functions, with examples, will now be considered.

Interfacing to MediaHub

Assumed output from various input modules of a multimodal system marked up in XML format is encapsulated within messages that are posted to *MediaHub Whiteboard*. Dot-delimited message types specify the content of messages passed within MediaHub.

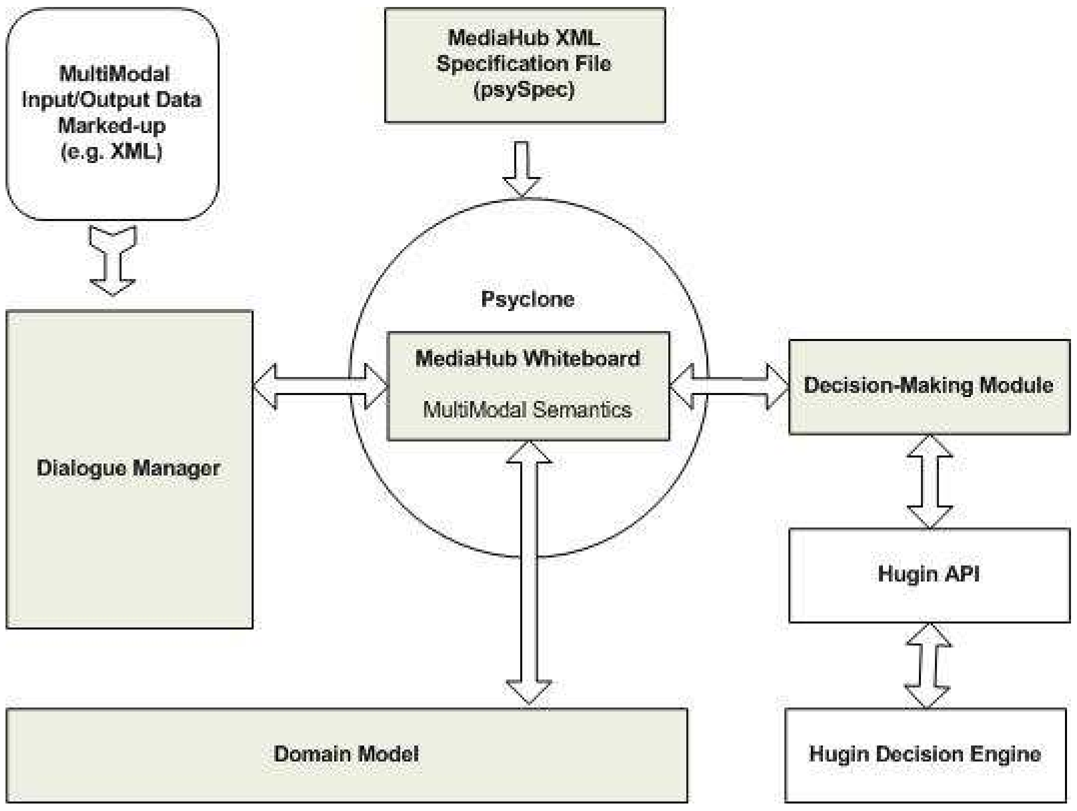


Figure 5.1: Architecture of MediaHub

All message types pertaining to input/output are automatically routed by Psyclone's whiteboard to the *Dialogue Manager*. Upon receiving input, the *Dialogue Manager* then decides, again based on the message type, how to process the input. The majority of input messages are processed and repackaged as new messages, with new message types, and posted back to *MediaHub Whiteboard*, where they are routed to the *Decision-Making Module*. When output messages are received the *Dialogue Manager* must decide which output modules should receive the output.

Semantic fusion

The *Dialogue Manager* coordinates the fusion of multimodal input/output. For example, fusion of speech input with its corresponding deictic gesture input or fusing the selection of a menu item with corresponding speech output. The problem of synchronisation is not fully addressed in the current implementation of MediaHub. The processing of multimodal input involves invoking a JDOM (Java Document Object Model) parser to retrieve only the relevant information from the semantic representation XML mark-up. Document Type Definitions (DTDs) determine when all the required information has been received for a particular scenario, based on message type, and to ensure correctness of the XML data received. One such DTD is given in Figure 5.2.

```
<!-- speech and gesture can be in any order-->
<!ELEMENT multimodal ((speech, gesture)|
(gesture, speech))>
<!ELEMENT speech (stype, category, subject,
stimestamp)>
<!ELEMENT stype (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT stimestamp (#PCDATA)>
<!ELEMENT gesture (gtype, coordinates,
gtimestamp)>
<!ELEMENT gtype (#PCDATA)>
<!ELEMENT coordinates (x, y)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT gtimestamp (#PCDATA)>
```

Figure 5.2: MediaHub example Document Type Definition (DTD)

The DTD in Figure 5.2 ensures that both speech and corresponding gesture input are received before proceeding with processing. Effectively, the DTD acts as a delay mechanism and the *Dialogue Manger* will not proceed to the next stage of processing until the XML mark-up contains all the required information as specified in the DTD. Message types invoke the correct DTD to validate an XML segment. Note that the DTDs can also specify optional information that may appear in the XML segment. A subset of MediaHub's DTDs is given in Appendix A.

Communication between MediaHub modules

The *Dialogue Manager*, as illustrated in Figure 5.1, communicates directly with *MediaHub Whiteboard*. All communication is achieved by exchanging semantic representations through *MediaHub Whiteboard*. Any messages posted to *MediaHub Whiteboard* with the text "input" or "output" in the message type are automatically routed to the *Dialogue Manager* which must then decide what future processing is required. Often this involves extracting the relevant information from the XML mark-up for the current situation and repackaging it in another message, with a new message type, which is posted back to *MediaHub Whiteboard*. It is usually necessary to acquire domain-specific information. As an example, consider the following dialogue segment:

1 U: Whose office is this [→][4]?

2 S: That is Paul's office.

3 U: Ok. Whose office is that [→]?

4 S: That's Sheila's office.

Here, in order to respond to turns 1 and 3 MediaHub must determine which office the user is pointing to. The XML representation of turns 1 and 3 will contain both the speech segment and the coordinates of the pointing gesture. These coordinates will then facilitate querying the *Domain Model* in order to determine whose offices are at those locations, i.e., Paul's and Sheila's office.

5.2.2. MediaHub Whiteboard

MediaHub Whiteboard has two primary functions: (1) communication and (2) semantic storage. A publish-subscribe mechanism for communication is achieved by means of the *MediaHub Whiteboard*, implemented with Psyclone's patent-pending Whiteboards™ (Thórisson et al. 2005). During processing, input/output semantics is stored on *MediaHub Whiteboard* in XML format. Modules subscribe to dot-delimited message types. Some examples of message types in MediaHub are listed below:

building.query.office.occupant.speech.input

building.query.office.occupant.gesture.pointing.input

building.query.office.occupant.repdoc

movies.gesture.pointing.input

The MediaHub messages types are listed in Appendix B. Modules can subscribe to message types within the XML specification file (*psySpec*). A *psySpec* is an XML configuration file read by Psyclone at invocation which defines the operation of all system modules. In addition to being triggered by message types defined in MediaHub's *psySpec*, a module may also retrieve information dynamically at run-time. When a message is posted to *MediaHub Whiteboard*, a copy of the message is automatically delivered to all modules subscribing to that message type. A copy of the message remains on *MediaHub Whiteboard* to facilitate future processing. Semantic storage is another key function of *MediaHub Whiteboard.* All previous messages are stored on *MediaHub Whiteboard* and can be retrieved later for decision-making. It is possible to query *MediaHub Whiteboard* and retrieve the last message of a certain type, i.e., a dialogue

[4] [→] is used here to indicate a deictic gesture.

history is maintained. MediaHub can also access the last *X* messages of a certain type and parse the XML content to assist the decision-making process.

5.2.3. Domain Model

The *Domain Model* contains data specific to a given application domain, e.g., building data, cinema ticket reservation, in-car safety. This data is stored in an XML format which can be parsed by a JDOM parser for XML. Figure 5.3 shows a segment of an XML file stored in the *Domain Model*.

```
<Offices>
      <Office>
            <ID>MG221</ID>
      <Person>
                   <FirstName>Paul</FirstName>
                   <Surname>McKevitt</Surname>
              <Gender>Male</Gender>
      </Person>

   <Coordinates>
         <From>
                  <X>1100</X>
                  <Y>2150</Y>
         </From>
            <To>
                  <X>1311</X>
                  <Y>2323</Y>
            </To>
      </Coordinates>
      </Office>
      <Office>
            <ID>MG203</ID>
            <Person>
            <FirstName>Sheila</FirstName>
                  <Surname>McCarthy</Surname>
            <Gender>Female</Gender>
            </Person>
        <Coordinates>
            <From>
                  <X>1400</X>
                  <Y>5300</Y>
            </From>
            <To>
                  <X>1525</X>
                  <Y>5500</Y>
            </To>
      </Coordinates>
</Office>
```

Figure 5.3: Segment of XML file containing data on offices

The XML segment in Figure 5.3 contains domain-specific information for the ‘building data’ domain. It contains the ID (room number), occupant name, gender of occupant and the coordinates for each of the offices in the building. Without such information it would be impossible to answer queries such as, “Whose office is that [→]?”, or, “How do I get from this office [→] to that office [→]?”. Upon receiving the coordinates of a pointing gesture, a JDOM parser can query the *Domain Model* to determine the office being referred to. The *Domain Model* code shown in Figure 5.4 checks the coordinates received in the XML mark-up of a deictic gesture against data in the *Domain Model*. When the correct office has been identified in the *Domain Model* the office ID, first name and gender of the occupant is extracted for use in the current dialogue, e.g., “That’s Paul’s office”. The *Domain Model* is accessed via the *DomainModel* Class.

```
if(intX >= xFrom && intX <= xTo && intY >= yFrom && intY <= yTo){

//retrieve the office number and occupant name

String strOfficeNo = ((Element)offices.get(x1))
                         .getChild("ID").getText();

String strOccupantName = ((Element)offices.get(x1)).getChild("Person")
                           .getChild("FirstName").getText();

String strOccupantGender = ((Element)offices.get(x1)).getChild("Person")
                           .getChild("Gender").getText();
```

Figure 5.4: Segment of *Domain Model* code

5.2.4. Decision-Making Module

A key component of MediaHub is the *Decision-Making Module*. The *Decision-Making Module* manages Bayesian decision-making by accessing the Hugin Decision Engine via the Hugin API, as shown in Figure 5.1. The *Decision-Making Module* deploys, where necessary, appropriate Bayesian networks and supplies these networks with data contained in the XML segments. The decision on which Bayesian network to access is determined by the message type. Where necessary, data relating to dialogue history is accessed via the *History* class. When a network is accessed by the *Decision-Making Module* the results or conclusions are interpreted with simple rules within it and a message is posted to *MediaHub Whiteboard*. All messages posted to *MediaHub Whiteboard* from the *Decision-Making Module* are automatically delivered to the *Dialogue Manager*.

The *Decision-Making Module* has at its disposal a collection of Bayesian networks developed with the Hugin GUI. For each decision-making scenario there exists a collection of Bayesian networks that can be deployed depending on context and message type. The Bayesian networks are accessed by the Hugin API for Java. The Java API for the Hugin Decision Engine offers a comprehensive array of methods for creating and accessing Bayesian networks. All of the Bayesian networks utilised by MediaHub were developed with the Hugin GUI and are opened, supplied with evidence (or input), and run by the *Decision-Making Module* in MediaHub.

5.3. *Semantic representation and storage*

MediaHub generates and interprets semantic representations of multimodal input/output data to support the fusion and synchronisation of multimodal data. MediaHub's *Dialogue Manager* receives marked up multimodal semantics in XML format which is parsed for data to support decision-making. The accuracy and completeness of XML semantics is checked by Document Type Definitions (DTDs), as discussed in Section 5.2.1. XML was chosen due to its compatibility with Java, its portability and the fact that it is easily extensible. Portability is important so that MediaHub can be integrated with existing multimodal systems that are deployed on different operating systems. The extensibility of XML affords flexibility in dealing with the varied and complex nature of multimodal semantics. Additionally, XML is a standard mark-up language used extensively for semantic representation within multimodal systems. XML is therefore deemed a practical choice for MediaHub which aims to be easily integrated with existing multimodal systems.

Multimodal systems frequently use a shared space, or *blackboard*, to maintain a record of dialogue history. The blackboard keeps track of all interactions over time so that semantic information on dialogue history may be accessed to perform more intelligent decision-making. MediaHub has a *whiteboard*, as discussed in Section 5.2.2, to maintain a history of all messages passed within MediaHub. Psyclone's whiteboards enable heterogeneous systems, hosted on different computers, to be connected together. The whiteboards in Psyclone effectively act as publish/subscribe servers. Information is both posted to, and dispatched from, the whiteboard to all modules subscribed to that type of information. The semantics of all multimodal input/output data is stored on *MediaHub Whiteboard* and is accessible at later stages of a multimodal dialogue, i.e., dialogue history is maintained on *MediaHub Whiteboard*.

5.4. *Distributed processing with Psyclone*

The nature of multimodal systems means that inputs to the decision-making process will typically arrive at different times from various distributed recognition and interpretation modules. The hub of a multimodal system must be capable of performing distributed processing, i.e., receiving input from the various system modules and routing this information to the appropriate destination modules within the system. Psyclone facilitates distributed processing in MediaHub. The architecture of Psyclone is shown in Figure 5.5. When Psyclone is invoked, it first reads the *psySpec* as shown by step (1) in Figure 5.5. Then, any internal or external modules are invoked, such as speech recognition (2) and computer graphics (3).

Psyclone then sets up appropriate subscription mechanisms for the modules and can be configured to automatically invoke other Psyclone servers as indicated by step (4). Step (4), a powerful feature of Psyclone, was not utilised in the current implementation of MediaHub. Psyclone is invoked with an executable file stored in MediaHub's working directory. Deploying the *psyclone.exe* file launches Psyclone which automatically initialises MediaHub's modules, as shown in Figure 5.6. Messages posted to *MediaHub Whiteboard* are automatically routed to the appropriate modules based on a dot-delimited message type. *OpenAIR* (Mindmakers 2009; Thórisson et al. 2005), implemented within Psyclone, is a communication protocol based on a publish-subscribe system architecture and is the protocol for communication within MediaHub.

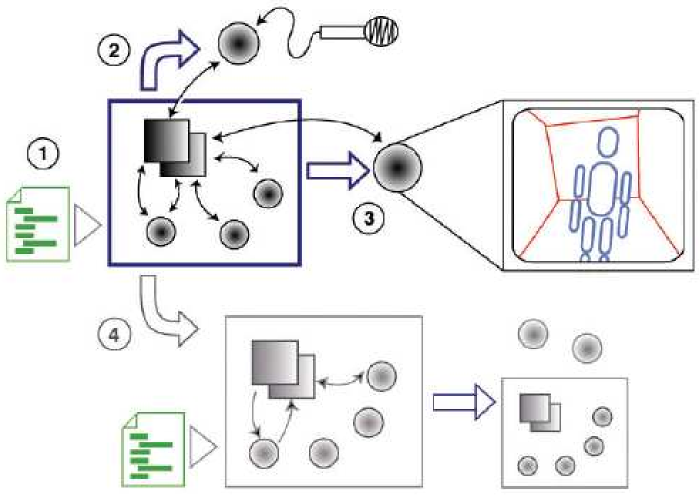
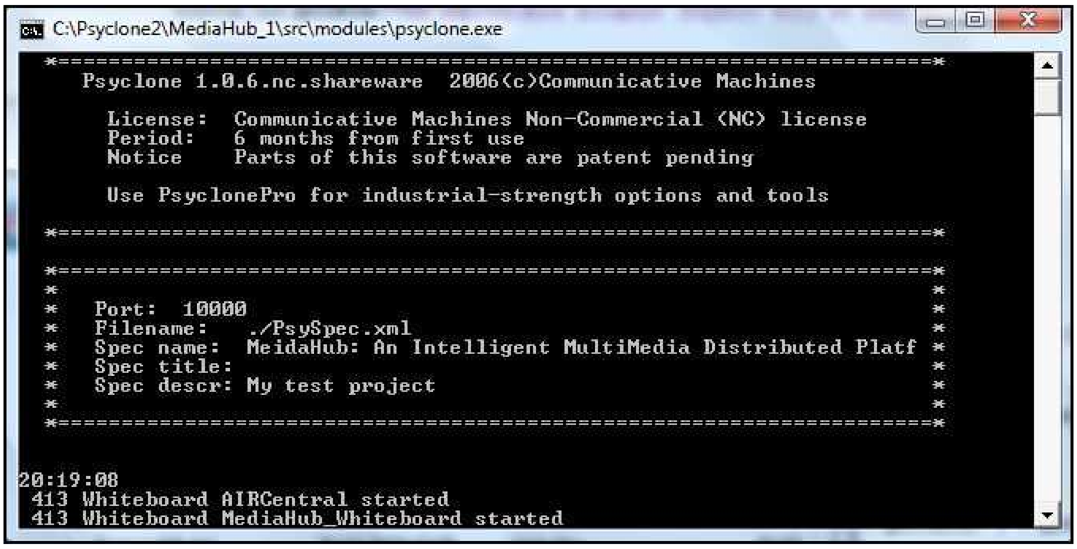


Figure 5.5: Architecture of Psyclone (Thórisson et al. 2005)



```
C:\Psyclone2\MediaHub_1\src\modules\psyclone.exe
 *===================================================================*
    Psyclone 1.0.6.nc.shareware  2006(c)Communicative Machines

      License:  Communicative Machines Non-Commercial (NC) license
      Period:   6 months from first use
      Notice    Parts of this software are patent pending

      Use PsyclonePro for industrial-strength options and tools

 *===================================================================*

 *===================================================================*
 *                                                                   *
 *   Port:  10000                                                    *
 *   Filename:    ./PsySpec.xml                                      *
 *   Spec name:  MeidaHub: An Intelligent MultiMedia Distributed Platf *
 *   Spec title:                                                     *
 *   Spec descr: My test project                                     *
 *                                                                   *
 *===================================================================*


20:19:08
 413 Whiteboard AIRCentral started
 413 Whiteboard MediaHub_Whiteboard started
```

Figure 5.6: Psyclone running in command window

5.4.1. MediaHub's psySpec

Psyclone has a central XML specification file (*psySpec*) for defining the setup of all system modules. The functionality of Psyclone's *psySpec* was discussed in Section 5.2.2. Although the *psySpec* can set a number of advanced configuration options, MediaHub's *psySpec* primarily starts *MediaHub Whiteboard* and registers modules to receive, or be triggered by, messages of a certain type. A module is subscribed to messages of a certain type with the *type* attribute of the *<trigger>* tag in the *psySpec*. A segment of MediaHub's *psySpec* is shown in Figure 5.7.

```
<module name="DomainModel">
<description>Used to access domain-specific information</description>
 <executable name="DomainModel" consoleoutput="yes">
 <sys ostype="Win32">
 java -cp .;JavaOpenAIR.jar DomainModel psyclone=%host%:%port%
  name=%name%
 </sys>
 </executable>
 <spec>
    <triggers from="any" allowselftriggering="no">
       <trigger type="MediaHub.shutdown"/>
       <trigger type="building.query.office.occupant.intdoc"/>
       <trigger type="cinema.request.reservation.intdoc"/>
```

Figure 5.7: Segment of MediaHub's psySpec.XML file

As shown in Figure 5.7, the *Domain Model* is registered to be triggered by messages of certain types with the *<trigger>* tag. Also included in the *psySpec* configuration of the *Domain Model* is the operating system type and a Java command to automatically invoke the module. Note that the host value is typically *localhost* and the default port is *10000* if not specified. The *from* attribute of the *<triggers>* tag defines the module that can send a message to the *Domain Model*. In this case, a message from any module can trigger the *Domain Model*, provided it is of a message type listed in the *psySpec*. The *allowselftriggering* tag here stops the *Domain Model* from being triggered by messages it has posted itself to *MediaHub Whiteboard*.

5.4.2. JavaAIRPlugs

Note that it is possible to override the settings specified in the *psySpec*. For example, a module can be registered to receive messages of a certain type at run time with a *JavaAIRPlug* connected to Psyclone. The Java code which makes a connection to Psyclone with a JavaAirPlug is shown in Figure 5.8.

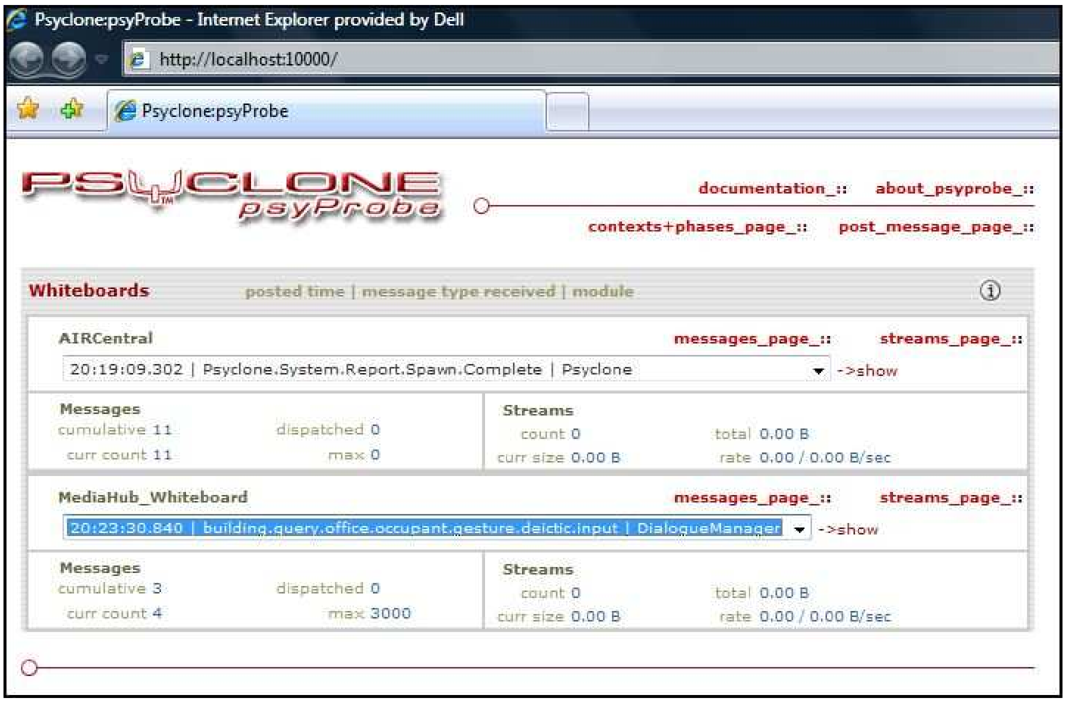
```
 plugDMM = new JavaAIRPlug("DMM", host, port);

if (!plugDMM.init()) {
  System.out.println("Could not connect to the Server on " + host
  + " on port " + port + "...");
   System.exit(0);
        }

 System.out.println("Connected to the Server on " + host +
               " on port " + port + "...");
}
```

Figure 5.8: Java code for establishing a connection to Psyclone

Once the connection to Psyclone has been established, the *JavaAIRPlug* then posts messages to, and retrieve messages from, *MediaHub Whiteboard*.

5.4.3. psyProbe

Psyclone enables developers to 'see inside the system' at run-time through a web-based interface called a *psyProbe*. A *psyProbe* is a built-in monitoring system that enables developers to monitor all activities of the system. Figure 5.9 shows the *psyProbe* viewing the messages posted to *MediaHub Whiteboard*.



```
Psyclone:psyProbe - Internet Explorer provided by Dell
http://localhost:10000/
Psyclone:psyProbe

PSYCLONE psyProbe          documentation_::   about_psyprobe_::
                     contexts+phases_page_::   post_message_page_::

Whiteboards        posted time | message type received | module

AIRCentral                                messages_page_::   streams_page_::
20:19:09.302 | Psyclone.System.Report.Spawn.Complete | Psyclone   ->show
Messages                           Streams
cumulative 11    dispatched 0        count 0        total 0.00 B
 curr count 11        max 0        curr size 0.00 B  rate 0.00 / 0.00 B/sec

MediaHub_Whiteboard                       messages_page_::   streams_page_::
20:23:30.840 | building.query.office.occupant.gesture.deictic.input | DialogueManager   ->show
Messages                           Streams
cumulative 3     dispatched 0        count 0        total 0.00 B
 curr count 4      max 3000        curr size 0.00 B  rate 0.00 / 0.00 B/sec
```

Figure 5.9: Viewing messages on MediaHub Whiteboard with *psyProbe*

psyProbe defaults to port 10000 of localhost and is usually accessed by browsing *http://localhost:10000*. A Psyclone server running on another machine can be accessed over the network by browsing *http://machine:10000*, where *machine* is the name of the computer running Psyclone. *psyProbe* can facilitate viewing time-stamped information on *MediaHub Whiteboard* and the content of individual messages with a standard web browser.

5.4.4. Psyclone contexts

Contexts in Psyclone are globally announced system states that manage the runtime behaviour of a system's modules. MediaHub's modules are context-driven in that they are only active when a certain context is active. MediaHub's modules primarily operate in the default context of *Psyclone.System.Ready*. Each of the modules in Psyclone is assigned at least one context and the module will not run until one of its contexts becomes true. Contexts enable individual modules to change their behaviour to meet the overall requirements of the system. Modules can be configured to perform different tasks in different contexts, thus reducing the number of separate modules that a system will need to implement.

5.5. *Decision-making layers in MediaHub*

The granularity of the decision-making necessary in a multimodal system varies considerably across different application domains. MediaHub implements a five layer approach in order to increase the resolution of its decision-making. The five layers of decision-making are illustrated in Figure 5.10.

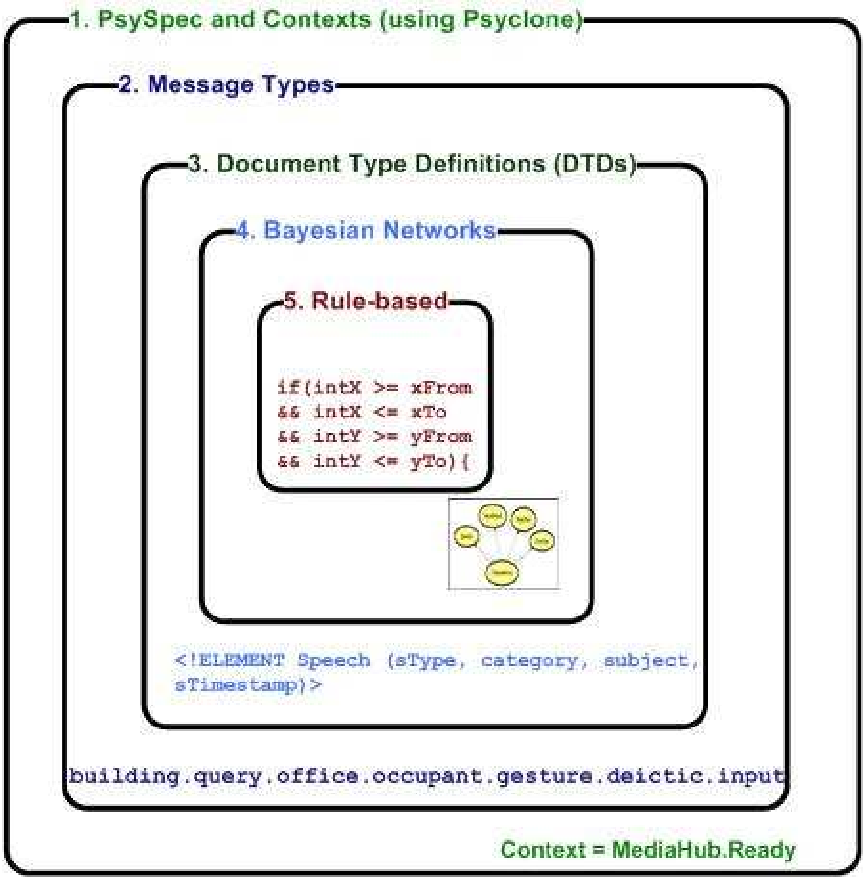

1. PsySpec and Contexts (using Psyclone)
2. Message Types
3. Document Type Definitions (DTDs)
4. Bayesian Networks
5. Rule-based
if(intX >= xFrom
&& intX <= xTo
&& intY >= yFrom
&& intY <= yTo){
<!ELEMENT Speech (sType, category, subject, sTimestamp)>
building.query.office.occupant.gesture.deictic.input
Context = MediaHub.Ready

Figure 5.10: MediaHub’s five decision-making layers

Layer 1 represents the decisions undertaken when MediaHub initialises. This type of decision-making is performed when Psyclone reads the *psySpec.XML* configuration file, as discussed in Section 5.4.1. Decisions in Layer 1 relate to determining what modules in MediaHub should receive which message types when the messages are posted to *MediaHub Whiteboard*. The *psySpec* can also set the context in which a module becomes active.

The second layer of decision-making analyses message types of received messages to determine their purpose and apply appropriate processing. Message types in MediaHub are dot-delimited strings that indicate the purpose of the message. The second layer is the most commonly used layer of decision-making in MediaHub since it facilitates all decisions across all application domains, regardless of whether or not Bayesian decision-making is applied. A collection of MediaHub message types is given in Appendix B.

The third layer of decision-making shown in Figure 5.10 is that performed with Document Type Definitions (DTDs). DTDs, as discussed in Section 5.2.1, are used by MediaHub modules to check: (1) that the XML received is syntactically correct and (2) that all the required information, e.g., semantics of speech, deictic gesture, eye-gaze, has been received at a particular stage in the decision-making.

The fourth layer of decision-making in MediaHub is the Bayesian networks layer. This is the layer concerned with the resolution of uncertainty and ambiguity in multimodal decision-making. It is this layer that deploys Bayesian networks to represent the cause-effect relations inherently present in multimodal decision-making. Note that ambiguity or uncertainty is not always present in multimodal decision-making and therefore Bayesian networks are only applied within MediaHub when they are deemed necessary.

The final layer of decision-making in MediaHub addresses rule-based decisions. This layer involves decisions at various stages of processing, e.g., what to do if the message type is of the form *x.y.z*, which DTD to check an XML segment against, how to proceed if a check against a DTD fails, how to interpret the results from a Bayesian network and how to proceed based on the results read from a Bayesian network.

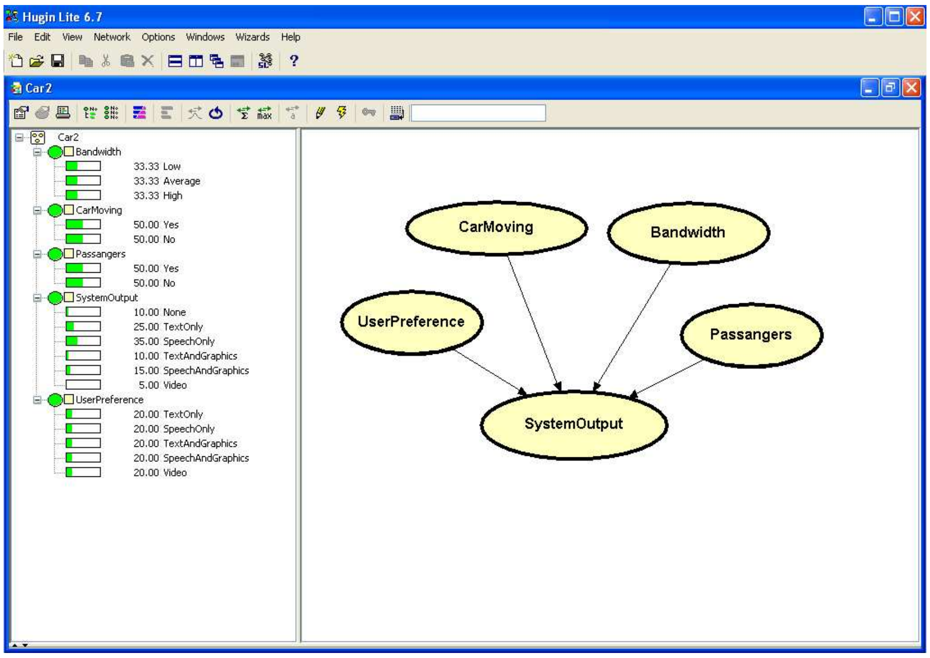
It should be noted that the layers shown in Figure 5.10 are not mutually exclusive but interleave with each other. It is possible that all layers can contribute to a given decision. For example, a message may be automatically delivered to the *Decision-making Module* based on the triggering information in the *psySpec* read by Psyclone (i.e. Layer 1) and, depending on its message type (i.e. Layer 2), the XML contents may be checked against a particular DTD to ensure all the relevant information has been received (i.e. Layer 3). Then, based on the message type and/or semantic content, an appropriate Bayesian network can be invoked for a particular application domain (i.e. Layer 4). Finally, the result of running the Bayesian network can be interpreted with rule-based decision-making (i.e. Layer 5). An example rule is: if the confidence of the state of a certain node is greater than 60%, then perform a certain action and, if it is less than 60%, ask for clarification from the user.

5.6. *Bayesian decision-making using Hugin*

MediaHub deploys Hugin to perform Bayesian decision-making over multimodal data. The Hugin software tools, consisting of the Hugin Graphical User Interface (GUI) and the Hugin Decision Engine or inference engine, enable the implementation of Bayesian networks as Causal Probabilistic Networks (CPNs). The functionality of the Hugin Decision Engine, or inference engine, is accessed through a set of Application Programming Interfaces (APIs) or through the Hugin GUI. MediaHub uses Hugin's Java API to access Bayesian networks for different application domains constructed using the Hugin GUI.

5.6.1. Hugin GUI

Bayesian networks for MediaHub are developed with the Hugin GUI and the API for Java opens, supplies evidence to, and runs the networks at run-time. The Hugin GUI, in run mode, is shown in Figure 5.11. The right pane of Figure 5.11 shows the Bayesian network, while the left pane shows the probabilities for all states of each of the nodes. It is possible to add evidence to the network by clicking on the input nodes in the left pane. The Hugin Decision Engine then automatically recalculates the new conditional probabilities for each of the nodes in the Bayesian network. The Hugin GUI is thus an invaluable tool for constructing and tuning a Bayesian network. When the iterative design process of a network has been completed, Java code is then written to open and run the network through the Hugin API.



```
Car2
  Bandwidth
    33.33 Low
    33.33 Average
    33.33 High
  CarMoving
    50.00 Yes
    50.00 No
  Passangers
    50.00 Yes
    50.00 No
  SystemOutput
    10.00 None
    25.00 TextOnly
    35.00 SpeechOnly
    10.00 TextAndGraphics
    15.00 SpeechAndGraphics
     5.00 Video
  UserPreference
    20.00 TextOnly
    20.00 SpeechOnly
    20.00 TextAndGraphics
    20.00 SpeechAndGraphics
    20.00 Video
```

Figure 5.11: Hugin Graphical User Interface (GUI)

5.6.2. Documentation of Bayesian networks

The Hugin GUI offers a useful feature that enables automatic generation of HTML documentation for Bayesian networks developed with the GUI. Provided a description has been given for each of the states and nodes of a Bayesian network, the generated document provides a concise summary of all nodes in the Bayesian network and the relationships between them. The generation of documentation can be accessed by selecting *Network* | *Generate Documentation* in the Hugin GUI. Doing so allows the developer to save a HTML file containing documentation of the current Bayesian network. Appendix C shows an example of

HTML documentation automatically generated for a Bayesian network implemented in MediaHub.

5.6.3. Use of Hugin API (Java)

This section discusses the use of the Hugin API (Java) to access Bayesian networks developed with the Hugin GUI.

Accessing Bayesian networks

The following code creates a new domain and opens an existing Bayesian network with the Hugin API for Java:

```
Domain domain = new Domain ("[Filename].net", new
DefaultClassParseListener());
domain.openLogFile("[Filename]." + ".log");
```

For example, the following code opens a Bayesian network called *CinemaTicketReservation* in the cinema ticket reservation domain:

```
Domain domain = new Domain ("CinemaTicketReservation.net", new
DefaultClassParseListener());
domain.openLogFile("CinemaTicketReservation" + ".log");
```

Since Bayesian networks are developed with the Hugin GUI, names of nodes in these networks are already defined and the following syntax accesses a node by its name:

```
LabelledDCNode [Node Name] = (LabelledDCNode) domain.getNodeByName
                              ("[Node Name]");
```

The following line of code accesses the *EyeGaze* node of the *CinemaTicketReservation* Bayesian network:

```
LabelledDCNode EyeGaze = (LabelledDCNode) domain.getNodeByName ("EyeGaze");
```

Supplying evidence

There are two steps involved in supplying evidence to a Bayesian network with the Hugin API: (1) retrieve the conditional probability table (CPT) for the node and (2) enter evidence to the states of the node. The following segment of code retrieves the CPT of a node in a Bayesian network:

```
[Table Name] = [NodeName].getTable();
```

Hence, the following line of code gets the CPT of the *EyeGaze* node of a Bayesian network:

```
table = EyeGaze.getTable();
```

The following syntax enters evidence on the node of a Bayesian network:

```
[Table Name]. enterFinding([State Number], [Probability Value]);
```

Note that states are numbered from 0, so the first state of a node has an index of 0, the second has an index of 1 and so on. The *EyeGaze* node in the *CinemaTicketReservation* Bayesian network has four states: *First*, *Second*, *Third* and *Fourth* that indicate which film on the list an eye-gaze tracking module believes the user is looking at. The following segment of code enters 65%, 25%, 5% and 5% on the *First*, *Second*, *Third* and *Fourth* states of the *EyeGaze* node:

```
EyeGaze.enterFinding(0, 0.65);
EyeGaze.enterFinding(1, 0.25);
EyeGaze.enterFinding(2, 0.05);
EyeGaze.enterFinding(3, 0.05);
```

The following code propagates the evidence and calculates updated beliefs for all the nodes in the Bayesian network:

```
domain.triangulate(Domain.H_TM_FILL_IN_WEIGHT);

domain.compile();

domain.propagate(Domain.H_EQUILIBRIUM_SUM, Domain.H_EVIDENCE_MODE_NORMAL);
```

Note that triangulation is the process of converting a graph of a domain into a triangulated graph. The triangulated graph forms the basis for the construction of the *JunctionTree* of the Domain. In this example, *H_TM_FILL_IN_WEIGHT* is the specified triangulation method. The propagate method propagates the evidence through the domain and has the equilibrium and evidence mode parameters, *Domain.H_EQUILIBRIUM_SUM*, representing the sum equilibrium state, and *Domain.H_EVIDENCE_MODE_NORMAL*, representing the normal mode for propagating evidence in Hugin.

Reading updated beliefs

On propagation of evidence in a Bayesian network the following syntax reads the updated beliefs of the states of a node:

```
[Variable Name]  = [Node Name].getBelief([State Number]);
```

The following segment of code reads the four states of the *ChosenMovie* node of the *CinemaTicketReservation* Bayesian network:

```
first = ChosenMovie.getBelief(0);
second = ChosenMovie.getBelief(1);
third = ChosenMovie.getBelief(2);
fourth = ChosenMovie.getBelief(3);
```

Saving a Bayesian network

The code below saves a network:

```
domain.saveAsNet("[File name]");
```

The following line of code saves the *CinemaTicketReservation* Bayesian network in the cinema ticket reservation domain:

```
domain.saveAsNet("CinemaTicketReservation.net");
```

5.7. *Example decision-making scenarios in MediaHub*

In order to demonstrate MediaHub's approach to decision-making, a number of scenarios are considered that address the key problems of anaphora resolution, domain knowledge awareness, multimodal presentation, turn-taking, dialogue act recognition and parametric learning with application domains of building data, cinema ticket reservation, in-car safety, intelligent agents and emotional state recognition. The nature of the decision-making in each of these problems demonstrates various capabilities of MediaHub.

5.7.1. Anaphora resolution

This example focuses on anaphora resolution in MediaHub using dialogue history. Consider the following sequence of turns taken from a dialogue between a user and an intelligent 'building data' system:

1 U: Whose office is this [→]?
2 S: That is Paul's office.
3 U: Ok. Whose office is that [→]?
4 S: That's Sheila's office.
5 U: Show me the route from her office to this [→] office.

Note that, in turns 1, 3 and 5, it is necessary to use domain-specific information to determine which offices the user is referring to. Additionally, in turn 5, it is necessary to use dialogue history to determine who '*her*' refers to. An extract from the *Domain Model* is shown in Figure 5.12. Note that each office or room has a set of *X-Y* coordinates that define its boundary on a 2D building plan. To illustrate MediaHub's approach to resolving this ambiguity, we can look closer at turns 1, 3 and 5 of the example dialogue and describe the corresponding actions taken within MediaHub. Semantic representations relating to turn 1 are packaged in two XML segments: (1) an XML segment containing the semantics of the speech input and (2) an XML segment containing the semantics of the deictic gesture, i.e., the coordinates of the pointing gesture. The semantics of the speech input is shown in Figure 5.13.

```
<?xml version="1.0"?>
<!DOCTYPE Offices SYSTEM
"C:\Psyclone2\DomainModel\BuildingInformation.dtd"
>
<Offices>
      <Office>
            <ID>MG221</ID>
      <Person>
             <FirstName>Paul</FirstName>
             <Surname>McKevitt</Surname>
              <Gender>Male</Gender>
      </Person>
   <Coordinates>
         <From>
                  <X>1100</X>
                  <Y>2150</Y>
         </From>
            <To>
                  <X>1311</X>
                  <Y>2323</Y>
            </To>
      </Coordinates>
      </Office>
      <Office>
            <ID>MG203</ID>
            <Person>
                  <FirstName>Sheila</FirstName>
                  <Surname>McCarthy</Surname>
                  <Gender>Female</Gender>
      </Person>
        <Coordinates>
            <From>
                  <X>1400</X>
                  <Y>5300</Y>
            </From>
            <To>
                  <X>1525</X>
                  <Y>5500</Y>
            </To>
      </Coordinates>
</Office>
</Offices>
```

Figure 5.12: *Domain Model* XML file for ‘anaphora resolution’

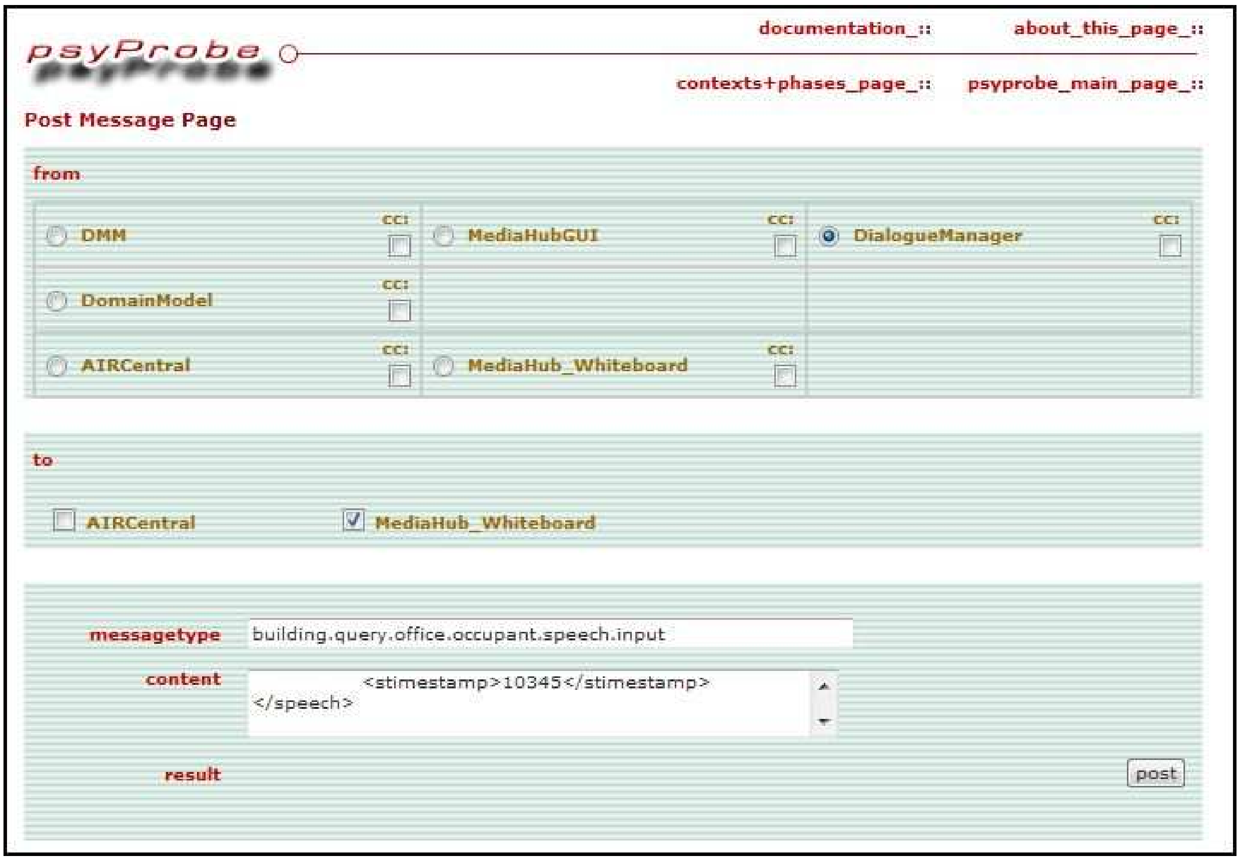
```
<speech>
      <stype>query-partial</stype>
      <category>Who</category>
      <subject>office</subject>
      <stimestamp>10345</stimestamp>
</speech>
```

Figure 5.13: Semantics of speech input for ‘anaphora resolution’

The semantics of the speech input is posted from the *Dialogue Manager* to *MediaHub Whiteboard* in a message of the following type:

building.query.office.occupant.speech.input

The posting of the speech input using Psyclone *psyProbe* is shown in Figure 5.14.



Post Message Page

from

( ) DMM cc: ☐
( ) MediaHubGUI cc: ☐
(●) DialogueManager cc: ☐
( ) DomainModel cc: ☐
( ) AIRCentral cc: ☐
( ) MediaHub_Whiteboard cc: ☐

to

☐ AIRCentral
☑ MediaHub_Whiteboard

messagetype building.query.office.occupant.speech.input

content
```
        <stimestamp>10345</stimestamp>
</speech>
```

result

Figure 5.14: Use of Psyclone *psyProbe* for 'anaphora resolution'

The corresponding semantics of the deictic gesture, shown in Figure 5.15, is posted from the *Dialogue Manager* to *MediaHub Whiteboard* in a message of type:

building.query.office.occupant.gesture.deictic.input

```
<gesture>
      <gtype>pointing</gtype>
      <coordinates>
            <x>1155</x>
            <y>2234</y>
      </coordinates>
      <gtimestamp>10312</gtimestamp>
</gesture>
```

Figure 5.15: Semantics of deictic gesture for 'anaphora resolution'

MediaHub Whiteboard is configured to automatically route messages of type *building.query**[5] to the *Decision-making Module*. This configuration is implemented in the *PsySpec.XML* file which configures Psyclone, and hence *MediaHub Whiteboard*, at initialisation. The segment of XML enabling this is shown in Figure 5.16.

[5] The asterisk here acts as a wildcard.

```
…
<module name="DMM">
…
<spec>
…
   <triggers from="any" allowselftriggering="no">
      <trigger type="building.query*"/>
      <trigger type="MediaHub.shutdown"/>
   </triggers>
 <posts>
   <post to="MediaHub_Whiteboard" type="dmm.register"/>
 </posts>
</spec>
…
```

Figure 5.16: Segment of *PsySpec.XML* configuring *MediaHub Whiteboard*

When the speech segment is received in the *Decision-making Module*, an if-else statement identifies the purpose of the message. The XML content is then retrieved, as shown in Figure 5.17.

```
 else if (strMsgType.equals("building.query.office.occupant.speech.input"))
{
    strSpeechPart = retrievedMsg.content; //retrieve speech semantics
```

Figure 5.17: Segment of if-else statement in *Decision-Making Module*

If the corresponding gesture input has been received, this is appended to the speech segment to form an integration document (*IntDoc*) that is stored in a string variable called *strIntDoc* (see Figure 5.18). This new XML segment is then checked against a Document Type Definition (DTD), *SpeechGesture.DTD*, which checks if all the required information is present in the XML segment. The DTD for this example is shown in Figure 5.19. If the check against *SpeechGesture.DTD* returns no errors, the integration document is packaged in a message and posted to *MediaHub Whiteboard* as follows:

```
boolean posted = plugDMM.postMessage("MediaHub_Whiteboard","building.query
                 .office.occupant.intdoc", strIntDoc, "English", "");
```

Domain Model

The *Domain Model* is configured to receive all messages of type *building.query.office.occupant.intdoc*. Again, this triggering is configured in the *PsySpec.XML* file. Note that, if the corresponding deictic gesture has not been received, the check against *SpeechGesture.DTD* fails and the *Decision-Making Module* continues to wait for the gesture input. Again, an if-else statement in the *Domain Model* applies appropriate processing to the XML integration document. The *X* and *Y* coordinates of the deictic gesture are then extracted from the XML, as shown in Figure 5.20.

```
strSpeechGestureDTD = "\u003C!DOCTYPE multimodal SYSTEM
\"C:/Psyclone2/DomainModel/SpeechGesture.dtd\"\u003E";

  if(strGesturePart != null){
  strIntDoc = strSpeechGestureDTD + strSpeechPart + strGesturePart;
  }
 else
      strIntDoc = strSpeechPart;
      System.out.println(strIntDoc);

      SAXBuilder builder = new SAXBuilder(true);
      Document doc;
      try {
    //convert the string to an xml document
    doc = builder.build(new InputSource(new
StringReader(strIntDoc)));
```

Figure 5.18: Checking XML segment against a Document Type Definition

```
<!-- speech and gesture can be in any order-->
<!ELEMENT multimodal ((speech, gesture)| (gesture, speech))>
<!ELEMENT speech (stype, category, subject, stimestamp)>
<!ELEMENT stype (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT stimestamp (#PCDATA)>
<!ELEMENT gesture (gtype, coordinates, gtimestamp)>
<!ELEMENT gtype (#PCDATA)>
<!ELEMENT coordinates (x, y)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT gtimestamp (#PCDATA)>
```

Figure 5.19: *SpeechGesture.DTD* for ‘anaphora resolution’

```
//convert the string to an xml document
doc = builder.build(new InputSource(new StringReader(strIntDoc)));

List allChildren = rootElement.getChildren();
 //Get the x coordinates of pointing gesture
 String strX = ((Element)allChildren.get(1)).getChild("coordinates")
                .getChild("x").getText();
 int intX = Integer.parseInt(strX);

//Get the x coordinates of pointing gesture
String strY = ((Element)allChildren.get(1)).getChild("coordinates")
               .getChild("y").getText();
int intY = Integer.parseInt(strY);
```

Figure 5.20: Extracting coordinates from XML Integration Document

A similar approach opens and parses the *BuildingData.XML* file, before checking which two offices the coordinates relate to. The X and Y coordinates of each office in the building are selected with the code given in Figure 5.21.

```
for(x1 = 0; x1 < intElementCount; x1++ )
           {
int xFrom =
Integer.parseInt(((Element)offices.get(x1)).getChild("Coordinates")
.getChild("From").getChild("X").getText());

int xTo =
Integer.parseInt(((Element)offices.get(x1)).getChild("Coordinates")
.getChild("To").getChild("X").getText());

int yFrom =
Integer.parseInt(((Element)offices.get(x1)).getChild("Coordinates")
.getChild("From").getChild("Y").getText());

int yTo =
Integer.parseInt(((Element)offices.get(x1)).getChild("Coordinates")
.getChild("To").getChild("Y").getText());
```

Figure 5.21: Extraction of coordinates for each office

Then, each set of coordinates is compared against the coordinates contained in the semantics. When a match has been found, the office ID and the name and gender of its occupant are extracted as shown in Figure 5.22. A replenished document (*RepDoc*) is then created containing this information and is forwarded to the *Dialogue Manager* via *MediaHub Whiteboard*. The *RepDoc* is now replenished with the data necessary for turn 2 of the dialogue. A segment of the *RepDoc*, containing the new data, is shown in Figure 5.23.

```
if(intX >= xFrom && intX <= xTo && intY >= yFrom && intY <= yTo){

// Get the office ID
String strOfficeNo = ((Element)offices.get(x1)).getChild("ID").getText();

// Get the name of occupant
String strOccupantName =
((Element)offices.get(x1)).getChild("Person").getChild("FirstName").getText();

// Get the gender of occupant
String strOccupantGender =
((Element)offices.get(x1)).getChild("Person").getChild("Gender").getText();
```

Figure 5.22: Parsing *Domain Model* for ‘anaphora resolution’

```
...
<gender>Male</gender>
<occupant>Paul</occupant>
<tts>That's Paul's office.</tts>
...
```

Figure 5.23: Segment of Replenished Document (*RepDoc*)

The *RepDoc* is posted to *MediaHub Whiteboard* with the following message type:

building.query.office.occupant.repdoc

All messages of type **repdoc* are automatically routed to the *Dialogue Manager*. Turn 3 of the example ‘building data’ dialogue is dealt with in exactly the same manner as turn 1.

Dialogue History

In order to respond to turn 5 (“Show me the route from her office to this [→] office.”) MediaHub must access dialogue history on *MediaHub Whiteboard* to determine who the user is referring to by uttering the word ‘her’. Here the gender of the occupant is relevant and, before the speech semantics can be combined with the semantics of the corresponding deictic gesture, the speech segment (see Figure 5.24) is checked against a different DTD, namely *SpeechGender.DTD*.

```
<!DOCTYPE speech SYSTEM "C:/Psyclone2/DomainModel/SpeechGender.dtd">
<speech>
      <type>request-partial</type>
      <category>show-route</category>
      <subcategory>from-to</subcategory>
      <subject>office</subject>
      <gender>female</gender>
      <stimestamp>10345</stimestamp>
</speech>
```

Figure 5.24: Speech segment for turn 5 of ‘anaphora resolution’

The request for dialogue history is packaged in a new type of MediaHub XML document called History Document (*HisDoc*) and this XML document is stored in a string variable called *strHisDoc*. As with all XML segments passed within MediaHub, the *HisDoc* is converted back into an XML document for parsing. In the *Decision-Making Module*, the History class is called with two parameters: (1) *QueryType* contains either *Building.Occupant.Male* or *Building.Occupant.Female* depending on gender and (2) *strSpeechFrom* which contains the relevant XML speech segment. The code which invokes the *History* class is shown in Figure 5.25.

```
else if (QueryType == "Building.Occupant.Male"){

        strXML = "<retrieve from=\"MediaHub_Whiteboard\"
type=\"building.occupant.hisdoc\"> <latest>3</latest> </retrieve>";
        coll = plugHistory.retrieveMessages(strXML);
```

Figure 5.25: Retrieval of dialogue history from *MediaHub Whiteboard*

Checking MediaHub Whiteboard in the History class

In the *History* class the last three messages of type *building.occupant.hisdoc* are retrieved from *MediaHub Whiteboard* as shown in Figure 5.26. Next, the contents of each message is converted to an XML document and parsed for information, e.g., occupant name, office ID, gender and timestamp. The timestamp then facilitates finding the last male referred to by the user, as shown in Figure 5.27.

```
if(strGender.equals("male")){
        QueryType = "Building.Occupant.Male";

        strSpeechFrom = retrievedMsg.content;

        new History(QueryType, strSpeechFrom);
        }

else if(strGender.equals("female")){
          QueryType = "Building.Occupant.Female";

        strSpeechFrom = retrievedMsg.content;

        new History(QueryType, strSpeechFrom);
        }
```

Figure 5.26: Calling *History* class from *Decision-Making Module*

```
if(strGender.equals("Male")){
   if(timestamp > latest){
       latest = timestamp;
       strLatestTimestamp = strTimestamp;
       lastMale = strName;
       lastOfficeNo = strOfficeNo;
        }
```

Figure 5.27: Finding the last male referred to in a dialogue

When the last male referred to in the dialogue has been identified, this information is repackaged in an XML speech segment as shown in Figure 5.28.

```
strHeader = "<multimodal><speech><stype>query-routepartial</stype>
<category>from-to</category><subject>from-office</subject>";
     strNameTag = "<from-occupant>" + lastMale + "</from-occupant>";
     strOfficeNoTag = "<no>" + lastOfficeNo + "</no>";
     strGenderTag = "<gender>male</gender>";
     strTimestampTag = "<stimestamp>" + strLatestTimestamp +
     "</stimestamp>"; strFooter = "</speech>";

strSpeechPart = strHeader + strNameTag + strOfficeNoTag + strGenderTag +
strTimestampTag + strFooter;
```

Figure 5.28: Repackaging speech segment in the *History* class

The speech segment in Figure 5.24 is then posted to *MediaHub Whiteboard* with the single line of code shown in Figure 5.29.

```
boolean posted = plugHistory.postMessage("MediaHub_Whiteboard","building
     .request.route.speech.from.office", strSpeechPart, "English", "");
```

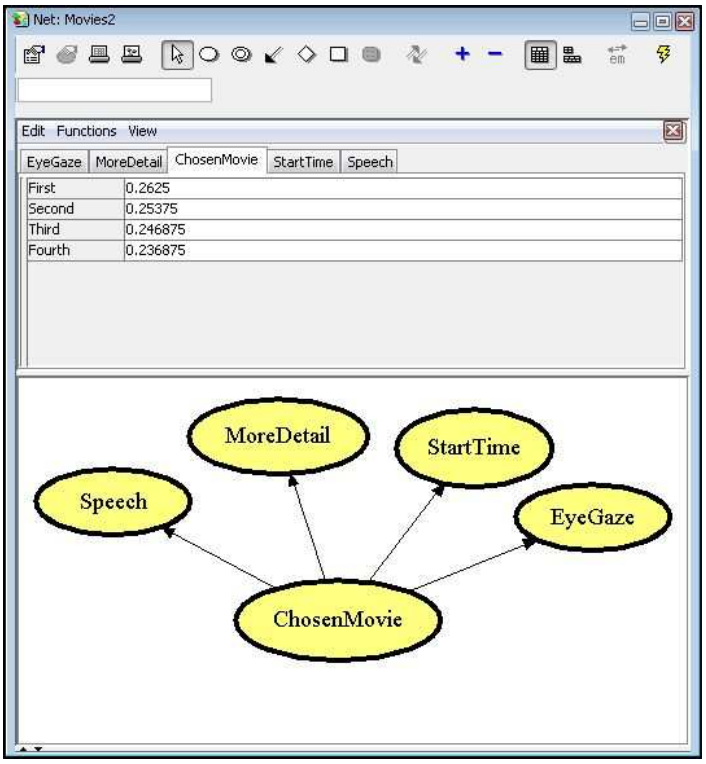
Figure 5.29: Posting speech segment from the History class to *MediaHub Whiteboard*

This message is automatically routed, based on its message type, to the *Decision-Making Module* which then waits for the corresponding deictic gesture. This is performed in exactly the same manner as before at turn 1 by checking the *IntDoc* against a Document Type Definition – a check that will only succeed when all the required information, i.e., semantics of speech and deictic gesture, is present.

This anaphora resolution example has focused on distributed processing and the resolution of internal ambiguity using dialogue history on *MediaHub Whiteboard*. In this example it was not necessary to utilise Bayesian decision-making. The next example focuses on the use of the *Domain Model* to support multimodal decision-making in MediaHub.

5.7.2. Domain knowledge awareness

Consider domain knowledge awareness in a multimodal system for booking cinema tickets. The system presents a list of four films to the user who selects the desired film with speech and eye-gaze input. Figure 5.30 shows the Bayesian network, called *CinemaTicketReservation,* for this example.



First	0.2625
Second	0.25375
Third	0.246875
Fourth	0.236875

Figure 5.30: Bayesian network for ‘domain knowledge awareness’

The Bayesian network in Figure 5.30 has one node, *ChosenMovie*, which has influence over four other nodes, namely *Watch*, *MoreDetail*, *StartTime* and *EyeGaze*. All nodes have the states *first*, *second*, *third* and *fourth*, which relate to a list of four movies that are currently showing at the cinema complex. The *Watch* node represents the belief, based on speech input, that the user wants to reserve tickets for a movie on the list. The *MoreDetail* node represents the fact that the user has previously asked for more information on the first, second, third and fourth movies on the list. The *MoreDetail* node accepts hard, or instantiation, evidence, i.e., either the user has asked for more details (indicated by *1*) or not (indicated by *0*). Unity on one or more states of the *MoreDetail* node indicates that the user has previously acquired more information about the movie.

MediaHub checks the dialogue history on *MediaHub Whiteboard* to determine if the user has previously requested more information about a particular movie. The *StartTime* node, again contains hard evidence, and represents the certain belief that the user has, or has not, enquired about the start time of a movie. The *EyeGaze* node represents the belief that that user is looking at each movie on the list based on input from a gaze-tracking module. The *EyeGaze* and *Watch* nodes primarily have soft, or likelihood, evidence applied (e.g., *0.1, 0.45, 0.67*).

Document Type Definition (DTD)

Two of the input nodes, *MoreDetail* and *StartTime*, in the Bayesian network shown in Figure 5.29 are populated following a query of *MediaHub Whiteboard* to establish whether or not the user had inquired about the start time or asked for more information about any of the movies. Note that information on all nodes is mandatory and this is reflected in the Document Type Definition (DTD) shown in Figure 5.31. As the speech, eye-gaze and dialogue history segments arrive in the *Decision-Making Module*, they are appended to an integration document (*IntDoc*). The *IntDoc* is checked against the *CinemaTicketReservatio*n DTD after each relevant input is received in the *Decision-Making Module*. As discussed in Section 5.7.1, the check against the DTD will not succeed until all the required information is present. When all the required information has been received, the *IntDoc* contains all the parameters to be supplied to the Bayesian network as shown in Figure 5.32. The *IntDoc* is then posted to *MediaHub Whiteboard*, which automatically delivers it to the *Domain Model* to be replenished with domain-specific information, i.e., the titles of the first, second, third and fourth movies in the list and the movie that is believed to be the focus of the user's eye-gaze. In this example, the *Domain Model* accesses domain-specific information from the *MoviesCurrentlyShowing.XML* file. This file contains a list of movies currently being shown, including their title, the start time, a link to a *.wav* file containing more information, a value indicating their position in the list and their coordinates on the display. A segment of this XML file is shown in Figure 5.33 and its corresponding DTD file is shown in Figure 5.34.

```
<!ELEMENT cinemaTicketReservation (speech, eyeGaze, moreDetail, startTime)>
<!ELEMENT speech (sFirst, sSecond, sThird, sFourth, sTimestamp)>
<!ELEMENT sFirst (#PCDATA)>
<!ELEMENT sSecond (#PCDATA)>
<!ELEMENT sThird (#PCDATA)>
<!ELEMENT sFourth (#PCDATA)>
<!ELEMENT sTimestamp (#PCDATA)>
<!ELEMENT eyeGaze (coordinates, belief, eTimestamp)>
<!ELEMENT coordinates (x, y)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT belief (#PCDATA)>
<!ELEMENT eTimestamp (#PCDATA)>
<!ELEMENT moreDetail (mFirst, mSecond, mThird, mFourth, mTimestamp)>
<!ELEMENT mFirst (#PCDATA)>
<!ELEMENT mSecond (#PCDATA)>
<!ELEMENT mThird (#PCDATA)>
<!ELEMENT mFourth (#PCDATA)>
<!ELEMENT mTimestamp (#PCDATA)>
<!ELEMENT startTime (stFirst, stSecond, stThird, stFourth, stTimestamp)>
<!ELEMENT stFirst (#PCDATA)>
<!ELEMENT stSecond (#PCDATA)>
<!ELEMENT stThird (#PCDATA)>
<!ELEMENT stFourth (#PCDATA)>
<!ELEMENT stTimestamp (#PCDATA)><!ELEMENT gTimestamp (#PCDATA)>
```

Figure 5.31: DTD for ‘domain knowledge awareness’

```
<!DOCTYPE multimodal SYSTEM
"C:/Psyclone2/DomainModel/CinemaTicketReservation.dtd">
<multimodal>
<Speech>
      <sType>request</sType>
      <category>reservation</category>
      <subject>movie</subject>
      <sTimestamp>10354</sTimestamp>
</Speech>
<MoreDetail>
      <mdFirst>1.0</mdFirst>
      <mdSecond>0.0</mdSecond>
      <mdThird>0.0</mdThird>
      <mdFourth>0.0</mdFourth>
</MoreDetail>
<StartTime>
      <stFirst>1.0</stFirst>
      <stSecond>0.0</stSecond>
      <stThird>0.0</stThird>
      <stFourth>0.0</stFourth>
</StartTime>
<EyeGaze>
      <coordinates>
            <x>1155</x>
            <y>2234</y>
      </coordinates>
      <gTimestamp>10312</gTimestamp>
</EyeGaze>
</multimodal>
```

Figure 5.32: Complete *IntDoc* for ‘domain knowledge awareness’

```
<?xml version="1.0"?>
<!DOCTYPE movies SYSTEM "C:\Psyclone2\DomainModel\MoviesCurrentlyShowing.dtd">
<movies>
  <movie>
      <title>The Whole Nine Yards</title>
      <starttime>2015</starttime>
      <moredetails>"C:\Psyclone2\DomainModel\TheWholeNineYardsSummary.wav"
      </moredetails>
      <no>1</no>
      <coordinates>
          <x>
            <from>900</from>
            <to>1200</to>
          </x>
          <y>
            <from>1800</from>
            <to>1900</to>
          </y>
            </coordinates>
   </movie>
   <movie>
      <title>The Green Mile</title>
      <starttime>2115</starttime>
      <moredetails>"C:\Psyclone2\DomainModel\TheGreenMileSummary.wav"
      </moredetails>
      <no>2</no>
      <coordinates>
          <x>
            <from>900</from>
            <to>1200</to>
          </x>
          <y>
            <from>1600</from>
            <to>1700</to>
          </y>
      </coordinates>
    </movie>
```

Figure 5.33: Domain-specific information for ‘domain knowledge awareness’

```
<!ELEMENT movies (movie+)>
<!ELEMENT movie (title, starttime, moredetails, no, coordinates)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT starttime (#PCDATA)>
<!ELEMENT moredetails (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT coordinates (x,y)>
<!ELEMENT x (from, to)>
<!ELEMENT y (from, to)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
```

Figure 5.34: DTD for ‘domain knowledge awareness’

In the *Domain Model*, the *IntDoc* is first parsed for the coordinates of the user’s eye-gaze. These are then checked against the position coordinates of each of the movies in the *MoviesCurrentlyShowing.XML* file using the code in Figure 5.35. When a match has been found the contents of the following XML tags are read:

- <*title*> which contain the name of the movie.
- <*starttime*> containing the start time in twenty-four hour format.
- <*moredetails*> which contains a URL to a *.wav* file.
- <*no*> which holds a number indicating the movie's position in the list presented to the user.

This information is then repackaged into two XML documents that are posted to *MediaHub Whiteboard*: (1) *RepDoc*, i.e., replenished document, which is the *IntDoc* instantiated with additional domain-specific data, i.e., the movie the user is believed to be looking at based on eye-gaze input, and (2) *HisDoc*, i.e. history document, which is a more concise document stored on *MediaHub Whiteboard* for the purpose of dialogue history retrieval (see Figure 5.36).

```
// intX and intY contain the eye-gaxe coordinates from the IntDoc
// xFrom, xTo, yFrom and yTo contain the coordinate vaules in the Domain Model

if(intX >= xFrom && intX <= xTo && intY >= yFrom && intY <= yTo){

String strMovieTitle =
(Element)movies.get(x1)).getChild("title").getText();

String strStartTime =
((Element)movies.get(x1)).getChild("starttime").getText();

String strMoreDetails =
((Element)movies.get(x1)).getChild("moredetails").getText();

String strNumber = ((Element)movies.get(x1)).getChild("no").getText();
}
```

Figure 5.35: Matching coordinates of eye-gaze in the *Domain Model*

```
// Send RepDoc to MediaHub Whiteboard

boolean posted = plugDomainModel.postMessage("MediaHub_Whiteboard",
"building.request.route.repdoc", strRepDoc, "English", "");

//Send HisDoc with movie title and position in list to Whiteboard (for
dialogue history)

boolean posted = plugDomainModel.postMessage("MediaHub_Whiteboard",
"building.request.route.hisdoc", strHistory, "English", "");
```

Figure 5.36: Code which posts *RepDoc* and *HisDoc* to *MediaHub Whiteboard*

To conclude this example the remaining key interactions in MediaHub are as follows:

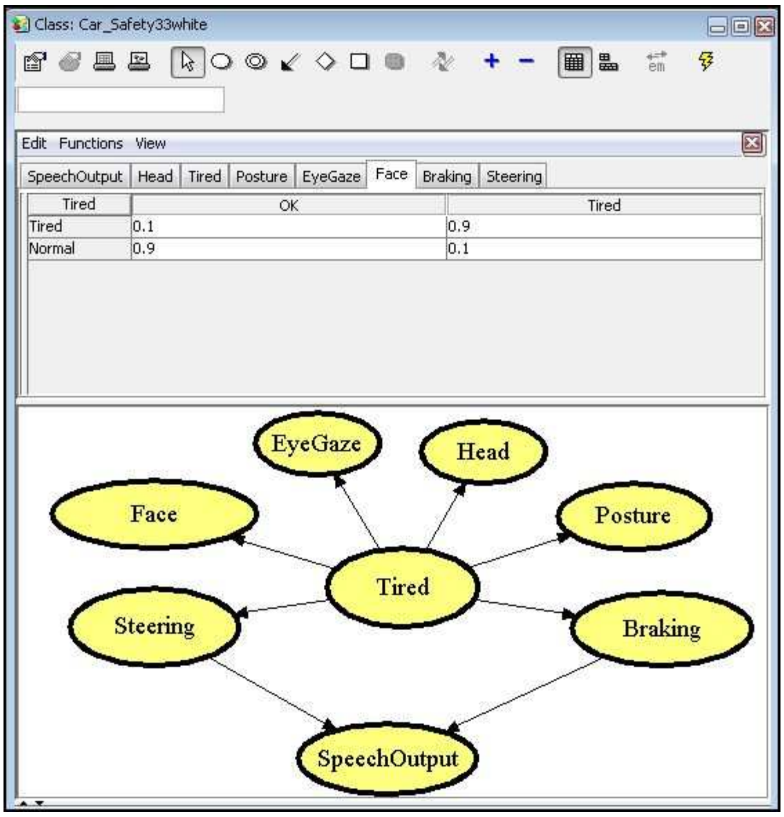
- When the *IntDoc* is received in the *Decision-Making Module*, it is checked against another DTD before the domain-specific data (movie title, position in list, name of the movie the user is looking at) is extracted.

- The values of each state of the *Speech*, *MoreDetails*, *StartTime* and *EyeGaze* nodes are read into variables.
- The *CinemaTicketReservation* Bayesian network is accessed via the Hugin API. Evidence, contained in the variables discussed in the previous step, is applied to the Bayesian network.
- The Bayesian network is run and the resulting values of the *First*, *Second*, *Third* and *Fourth* nodes are captured. These are posted to MediaHub Whiteboard and are then automatically routed to the *Decision-Making Module*.
- The *Decision-Making Module* decides whether a conclusion can be reached or not, i.e., is there sufficient confidence attached to the winning hypothesis? A decision is taken subsequently to either confirm the booking of the identified movie or ask the user for clarification.
- An XML-based representation of the required action is posted to MediaHub Whiteboard, where it is automatically delivered to the *Dialogue Manager*.

This ‘domain knowledge awareness’ example has focused on the role of the *Domain Model* in supporting multimodal decision-making in MediaHub. This has included detail on how Document Type Definitions (DTDs) facilitate checking the validity of XML semantic representations and ensure that all the required data relating to different modalities has been received. A Bayesian network represents the semantics of speech and eye-gaze input in the Speech and *EyeGaze* nodes. Dialogue history determines whether the user had previously asked for more information about a movie or had inquired about its start time. The semantics of this dialogue history information is captured in the *MoreDetail* and *StartTime* nodes of the Bayesian network. The actual opening, editing and running of the *CinemaTicketReservation* Bayesian network has not been explicitly discussed in this section. In the remaining examples, the focus is placed entirely on the implementation of Bayesian networks for decision-making in MediaHub.

5.7.3. Multimodal presentation

Consider the problem of multimodal presentation in an in-car safety system which monitors the driver’s steering, braking, facial expression, gaze, head movement and posture and gives a warning if it believes the driver is tired. The Bayesian network for this decision-making scenario is shown in Figure 5.37. As shown in Figure 5.37, there are four nodes that represent the belief that the driver is tired based on facial expression (*Face*), eye-gaze (*EyeGaze*), head movement (*Head*) and posture (*Posture*). Each of these multimodal nodes has the states *Tired* and *Normal* which represent the belief that the driver looks tired, or not, based on the modality, or evidence, observed. Two nodes, *Steering* and *Braking*, monitor the driver's behaviour. Both these nodes have two states: (1) *Normal* – representing the belief that the driver's steering or braking is normal and (2) *Abrupt* – expressing the belief that the driver's steering or braking is abrupt or harsh.



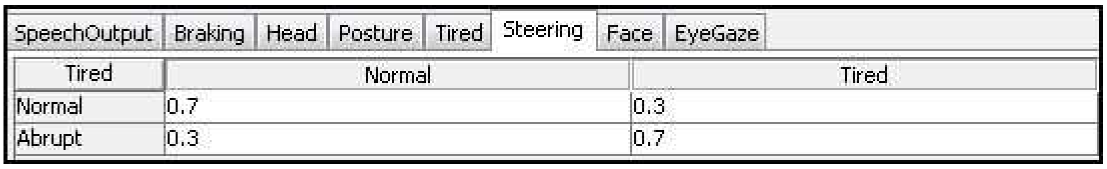
Tired	OK	Tired
Tired	0.1	0.9
Normal	0.9	0.1

Figure 5.37: Bayesian network for 'multimodal presentation'

The *Tired* node in the Bayesian network has the states *Tired* and *Normal*. The *SpeechOutput* node has three states: (1) *None* – representing the belief that no action on the part of the system is necessary, (2) *FancyBreak?* – which represents the belief that the system should suggest that the driver takes a break and (3) *Warning* – representing the belief, based on the evidence observed, that the driver is too tired and a warning should be issued through speech output.

The Bayesian network shown in Figure 5.37 captures a number of cause-effect relations in the in-car safety application domain. As shown by the directed edges in the Bayesian network, the *Tired* node has influence over the *Steering*, *Braking*, *Face*, *EyeGaze*, *Head* and *Posture* nodes, i.e., the fact that the driver is tired will affect steering, braking, and the signs of tiredness evident in the facial expression, eye-gaze, head movement and posture of the driver. Also note that the *Steering* and *Braking* nodes have direct influence over the *SystemOutput* node, whilst the *Face*, *EyeGaze*, *Head* and *Posture* nodes have indirect influence over the *SystemOutput* node through the *Tired* node in the Bayesian network. The causal relations present in the 'in-car safety' application domain are encoded in the Conditional Probability

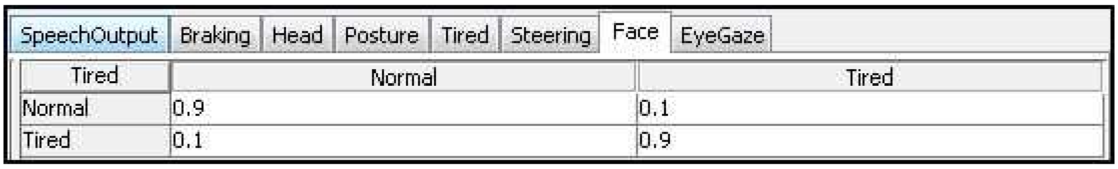
Tables (CPTs) of the nodes in the Bayesian network. The CPTs of the 'multimodal presentation' Bayesian network are shown in Figures 5.38 – 5.45.



SpeechOutput | Braking | Head | Posture | Tired | Steering | Face | EyeGaze

Tired	Normal	Tired
Normal	0.7	0.3
Abrupt	0.3	0.7

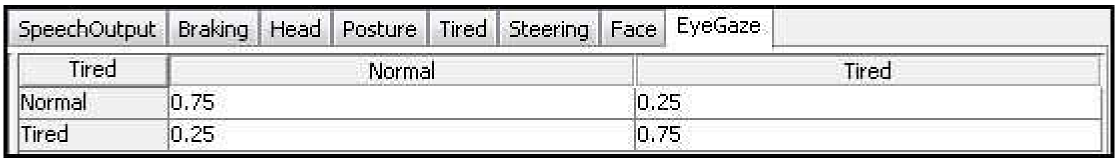
Figure 5.38: CPT of *Steering* node



SpeechOutput | Braking | Head | Posture | Tired | Steering | Face | EyeGaze

Tired	Normal	Tired
Normal	0.9	0.1
Tired	0.1	0.9

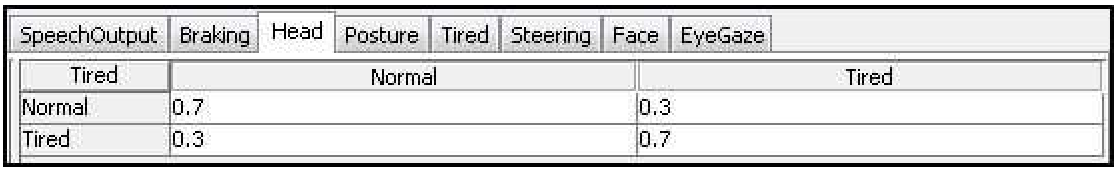
Figure 5.39: CPT of *Face* node



SpeechOutput | Braking | Head | Posture | Tired | Steering | Face | EyeGaze

Tired	Normal	Tired
Normal	0.75	0.25
Tired	0.25	0.75

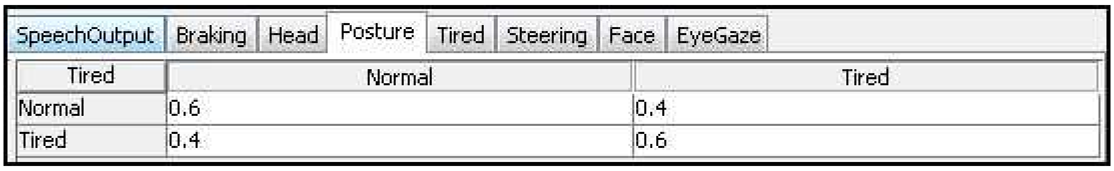
Figure 5.40: CPT of *EyeGaze* node



SpeechOutput | Braking | Head | Posture | Tired | Steering | Face | EyeGaze

Tired	Normal	Tired
Normal	0.7	0.3
Tired	0.3	0.7

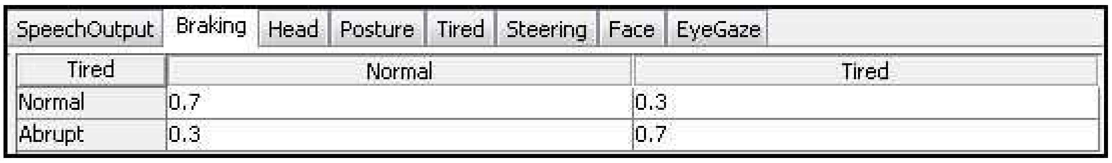
Figure 5.41: CPT of *Head* node



SpeechOutput | Braking | Head | Posture | Tired | Steering | Face | EyeGaze

Tired	Normal	Tired
Normal	0.6	0.4
Tired	0.4	0.6

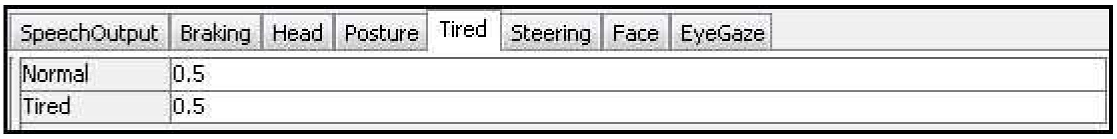
Figure 5.42: CPT of *Posture* node



SpeechOutput | Braking | Head | Posture | Tired | Steering | Face | EyeGaze

Tired	Normal	Tired
Normal	0.7	0.3
Abrupt	0.3	0.7

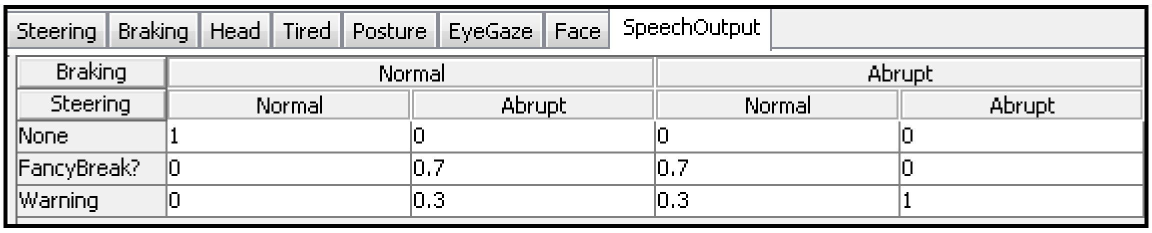
Figure 5.43: CPT of *Braking* node



SpeechOutput | Braking | Head | Posture | Tired | Steering | Face | EyeGaze

Normal	0.5
Tired	0.5

Figure 5.44: CPT of *Tired* node



Steering | Braking | Head | Tired | Posture | EyeGaze | Face | SpeechOutput

Braking	Normal		Abrupt	
Steering	Normal	Abrupt	Normal	Abrupt
None	1	0	0	0
FancyBreak?	0	0.7	0.7	0
Warning	0	0.3	0.3	1

Figure 5.45: CPT of *SpeechOutput* node

Due to the ability of Bayesian networks to perform abductive reasoning, i.e., from effect to cause, evidence that a driver is braking abruptly will increase the belief that the person is tired. Similarly, if the belief that the driver is tired based on his/her facial expression is increased, then the value of the *Tired* state of the *Tired* node in the Bayesian network in Figure 5.37 will also increase.

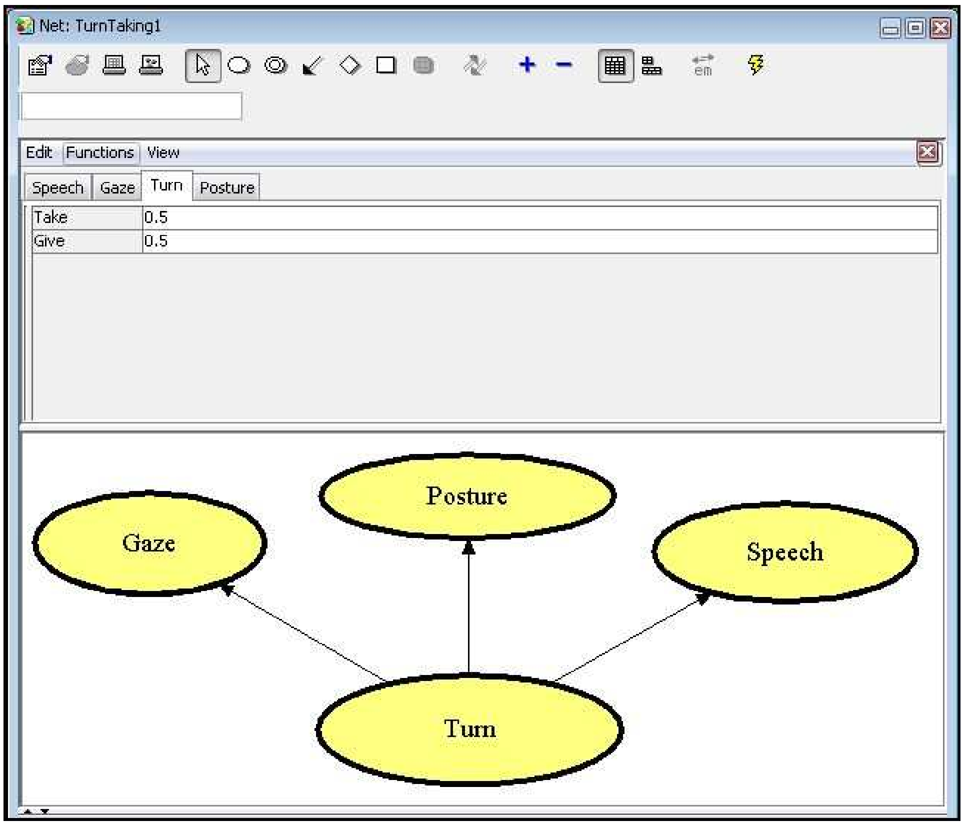
When accessed through the Hugin API, the Bayesian network in Figure 5.37 can recommend system output depending upon its beliefs about the tiredness of the driver. For example, if the driver is deemed tired, i.e., the *FancyBreak?* state of the *SystemOutput* node has a value greater than that of the *None* and *Warning* states, the system can issue the prompt, "Would you like a break? You look a little tired.". If the driver is believed to be very tired, i.e., the *Warning* state of the *SystemOutput* node has a value greater than that of the *None* and *FancyBreak?* states, then the system could issue the prompt, "Please pull over for a short break, as you appear too tired to drive!". Of course, other information could be used to influence the decision on the likelihood of the driver being tired. For example, the length of time since the journey commenced or the time since the last break could be incorporated into the set of rules applied in interpreting the resulting values of the states in the *SystemOutput* node. The key interactions in MediaHub for this example are summarised as follows:

- XML semantics of the driver's facial expression, eye-gaze, head movement and posture and an XML file relating to the steering and braking behaviour of the driver are received in the *Dialogue Manager*.
- The *Dialogue Manager* identifies the application domain and purpose of both messages using their message types.
- A DTD confirms the accuracy and completeness of the XML semantics.
- In the *Decision-Making Module*, the XML *IntDoc* is checked against a DTD before the input values of the states in the 'multimodal presentation' Bayesian network are extracted.
- The Bayesian network is opened with the Hugin API. Available evidence is supplied to the *Steering*, *Braking*, *Face*, *EyeGaze*, *Head* and *Posture* nodes.

- The supplied evidence is propagated through the Bayesian network.
- The resulting values of the states in the *SystemOutput* node are read and interpreted in the *Decision-Making Module* with *if-else* rules.
- The XML semantics of the recommended system output is sent to *MediaHub Whiteboard* for the attention of a speech synthesis module that could interpret the semantics and produce appropriate speech output.

5.7.4. Turn-taking

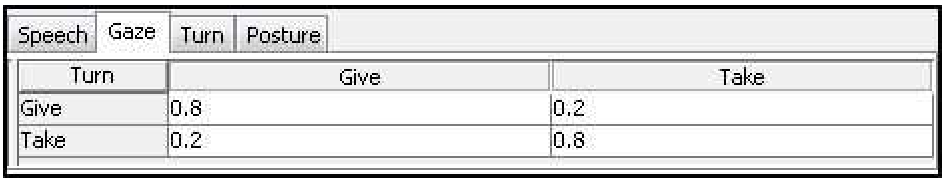
In this example we consider the problem of turn-taking strategy for an intelligent agent. The Bayesian network in Figure 5.46 can support decision-making in respect of turn-taking in an intelligent agent. The Bayesian network has three nodes that receive input information from gaze-tracking (*Gaze*), posture recognition (*Posture*) and speech recognition (*Speech*) modules. These nodes all have the same two states, *Give* and *Take*, that represent the belief that the user wants to give or take a turn. The *Turn* node relates to the decision of the intelligent agent to give or take a turn and also has the states *Give* and *Take*. The CPTs of each node in the Bayesian network are given in Figures 5.47 - 5.50.



Take	0.5
Give	0.5

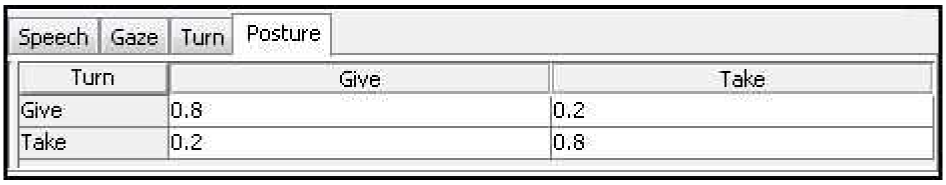
Figure 5.46: Bayesian network for 'turn-taking'

Turn-taking in intelligent agents is a complex task and the Bayesian network in Figure 5.46 is not intended to comprehensively model turn-taking. Rather, the Bayesian network is intended to be used in conjunction with other modules to enable natural turn-taking in an intelligent agent. In this example, the key decisions are those made by the recognition modules that provide the input information to the *Give* and *Take* states of the *Gaze*, *Posture* and *Speech* nodes. Such modules are not implemented in MediaHub, although possible outputs from these modules are assumed and presented in XML format to the *Dialogue Manager*. The Bayesian network in Figure 5.46 augments the individual beliefs of the gaze-tracking, posture recognition and speech recognition modules and decides whether or not it is appropriate for the intelligent agent to take a turn at a particular stage in a multimodal dialogue.



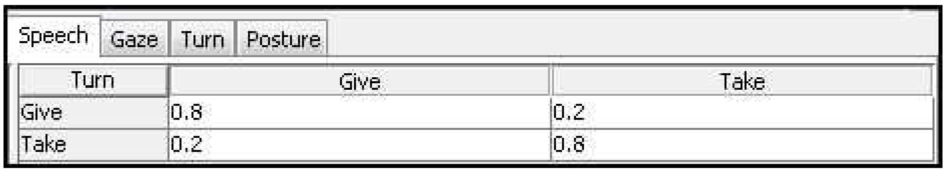
Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

Figure 5.47: CPT of *Gaze* node



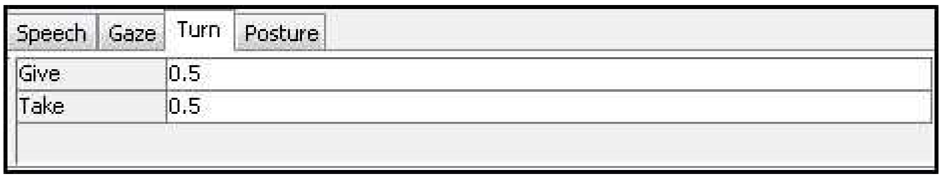
Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

Figure 5.48: CPT of *Posture* node



Turn	Give	Take
Give	0.8	0.2
Take	0.2	0.8

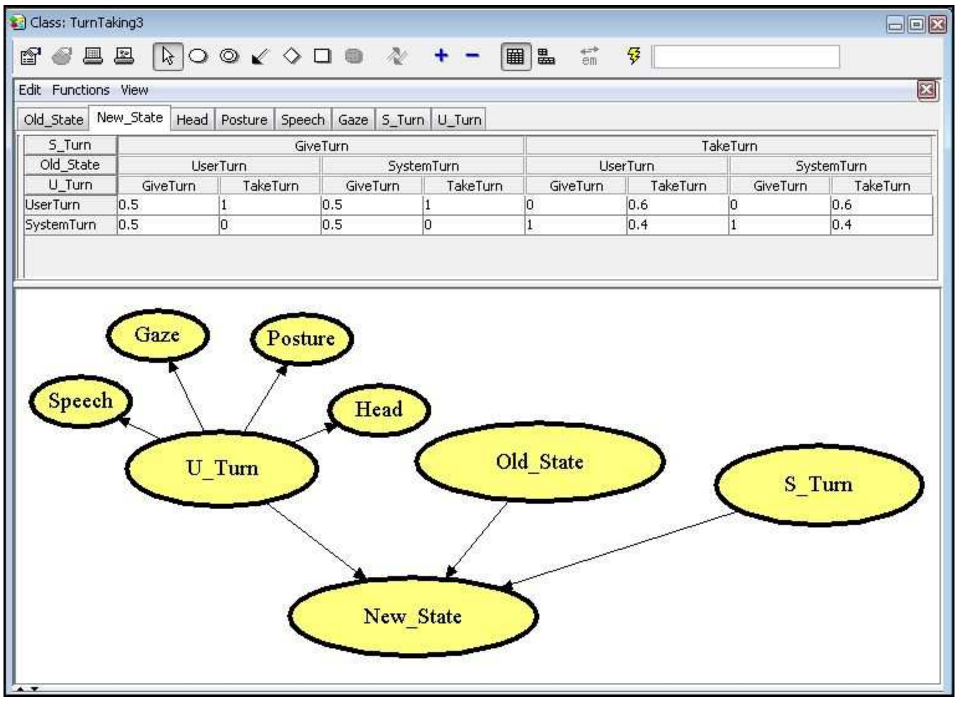
Figure 5.49: CPT of *Speech* node



Give	0.5
Take	0.5

Figure 5.50: CPT of *Turn* node

Whilst the Bayesian network in Figure 5.46 is simplified and is only intended to complement the decision-making of other modules in an intelligent agent system, an alternative more powerful Bayesian network for the ‘turn-taking’ example is shown in Figure 5.51. As shown in Figure 5.51, *Speech*, *Gaze*, *Posture* and *Head* nodes represent beliefs that the user wishes to take or give a turn. Each of these nodes has the states *Give* and *Take*. Note that such nodes are not necessary for the system, or intelligent agent, since the agent will already know when it needs to take a turn. The Bayesian network in Figure 5.51 contains nodes that represent the turn-taking intentions of both the system (*S_Turn*) and the user (*U_Turn*). Both the *U_Turn* and *S_Turn* nodes have two states: (1) *GiveTurn* – representing the belief that the user/system wishes to give the turn to the system/user and (2) *TakeTurn* – representing the belief that the user/system wishes to take the turn from the system/user. Old and new turn-taking states are represented by the *Old_State* and *New_State* nodes.



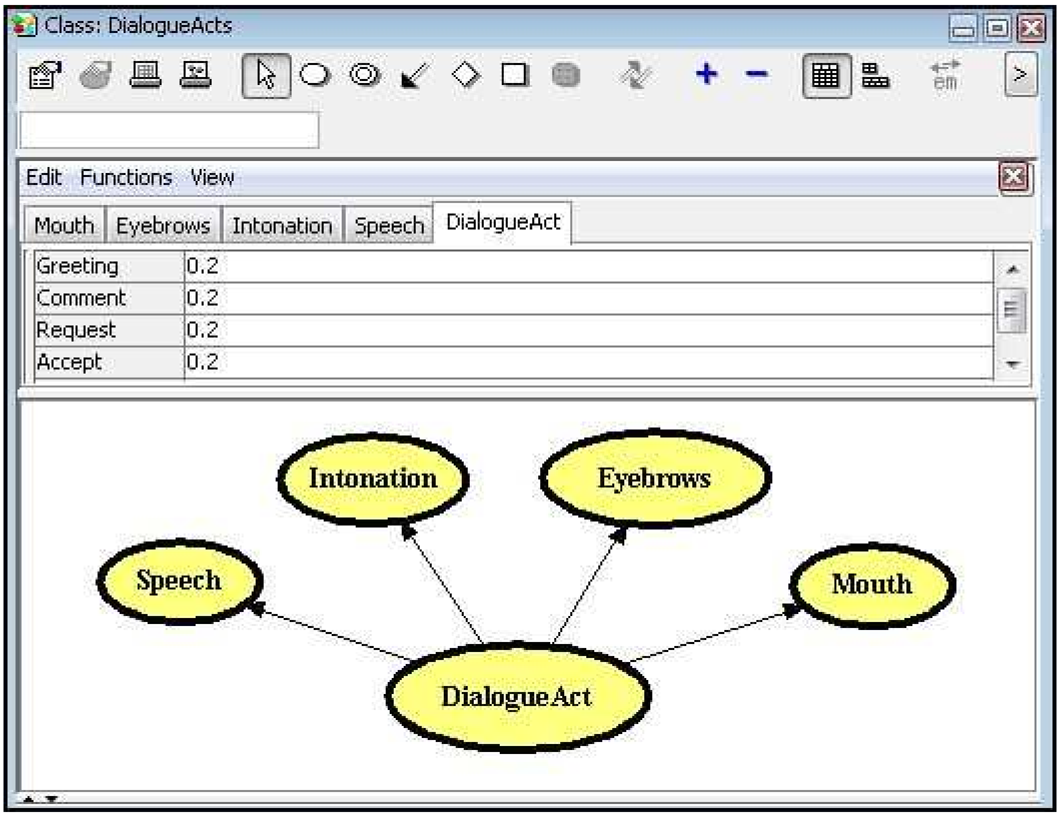
S_Turn	GiveTurn				TakeTurn			
Old_State	UserTurn		SystemTurn		UserTurn		SystemTurn	
U_Turn	GiveTurn	TakeTurn	GiveTurn	TakeTurn	GiveTurn	TakeTurn	GiveTurn	TakeTurn
UserTurn	0.5	1	0.5	1	0	0.6	0	0.6
SystemTurn	0.5	0	0.5	0	1	0.4	1	0.4

Figure 5.51: Alternative Bayesian network for ‘turn-taking’

Both these nodes contain the states *UserTurn* and *SystemTurn*, and relate to the dialogue participant who currently holds the turn (*Old_State*) and the participant that will take the next turn (*New_State*). Note that many other possibilities exist for the design of a Bayesian network to support turn-taking in an intelligent agent. It is likely that several different Bayesian networks will be needed in this, and other, key problem areas. When the required Bayesian networks have been implemented, MediaHub can use a combination of message types, DTDs and basic rules to decide which Bayesian network to invoke for a particular situation.

5.7.5. Dialogue act recognition

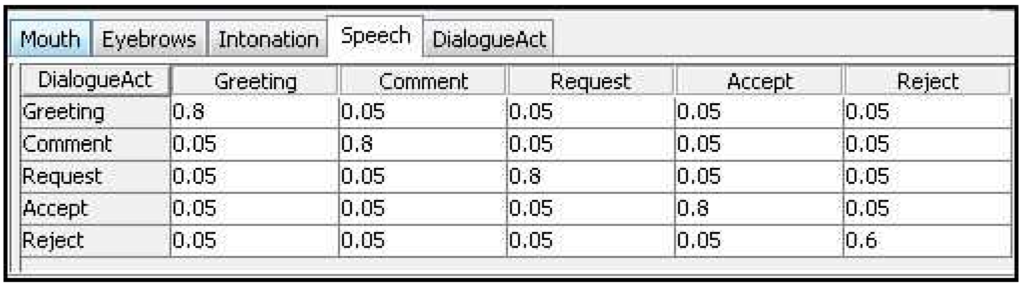
Consider the problem of dialogue act recognition in an ‘intelligent travel agent’ that engages in multimodal communication with users wishing to book a holiday. The understanding of speech signals and recognition of facial expressions (eyes and mouth) facilitates ambiguity resolution relating to user dialogue acts. The system’s Bayesian network combines beliefs associated with multimodal input to make decisions about the intentions of the user. The Bayesian network for this example is shown in Figure 5.52.



Greeting	0.2
Comment	0.2
Request	0.2
Accept	0.2

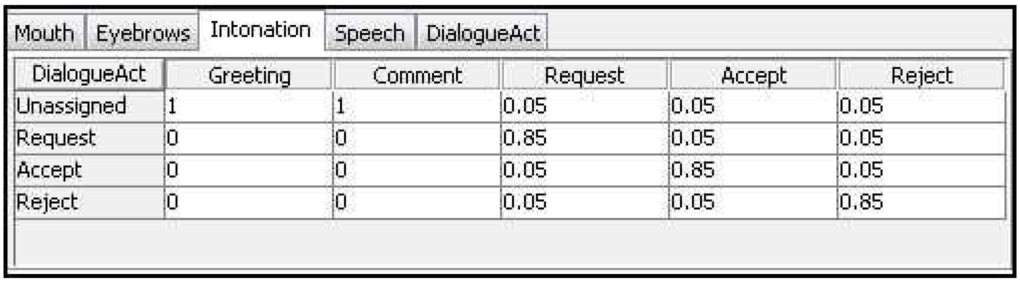
Figure 5.52: Bayesian network for ‘dialogue act recognition’

As shown in Figure 5.52 there are four input nodes in the Bayesian network, *Speech*, *Intonation*, *Eyebrows* and *Mouth* and one output node, *DialogueAct*. Note that the *Eyebrow* node of Figure 5.52 is not concerned with the focus of user’s gaze, rather it pertains to the recognition of muscle movement around the eye and, in particular, the eyebrows. Likewise, the *Mouth* node is not related to the recognition of lip movement but is populated following the interpretation of the shape and movement of the mouth, e.g., smile or frown. The *Speech* node represents the recognition of utterances from the user, whilst the *Intonation* node relates to voice intonation. The CPTs for each of the nodes depicted in Figure 5.52 are shown in Figures 5.53 - 5.57.


Mouth
Eyebrows
Intonation
Speech
DialogueAct

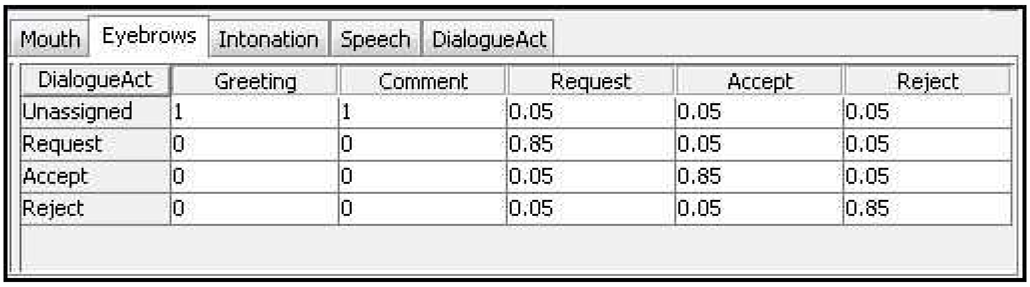
DialogueAct	Greeting	Comment	Request	Accept	Reject
Greeting	0.8	0.05	0.05	0.05	0.05
Comment	0.05	0.8	0.05	0.05	0.05
Request	0.05	0.05	0.8	0.05	0.05
Accept	0.05	0.05	0.05	0.8	0.05
Reject	0.05	0.05	0.05	0.05	0.6

Figure 5.53: CPT of *Speech* node



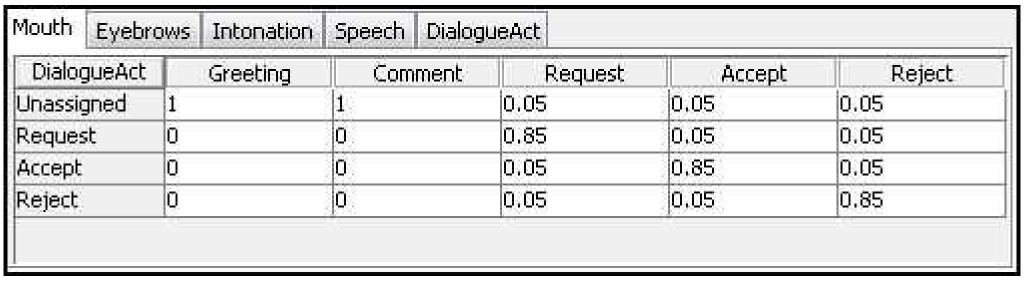
DialogueAct	Greeting	Comment	Request	Accept	Reject
Unassigned	1	1	0.05	0.05	0.05
Request	0	0	0.85	0.05	0.05
Accept	0	0	0.05	0.85	0.05
Reject	0	0	0.05	0.05	0.85

Figure 5.54: CPT of *Intonation* node



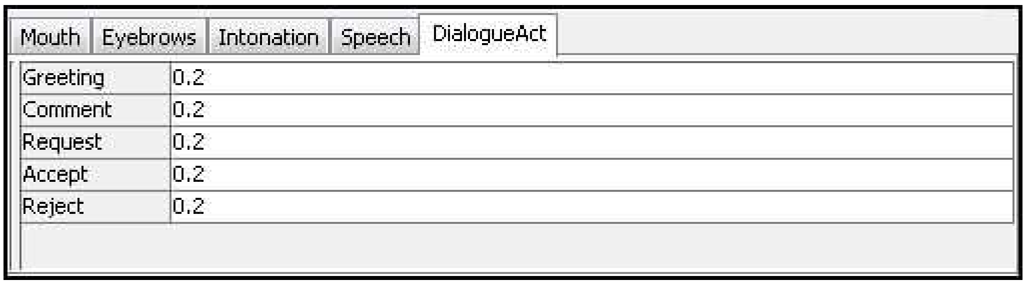
DialogueAct	Greeting	Comment	Request	Accept	Reject
Unassigned	1	1	0.05	0.05	0.05
Request	0	0	0.85	0.05	0.05
Accept	0	0	0.05	0.85	0.05
Reject	0	0	0.05	0.05	0.85

Figure 5.55: CPT of *Eyebrows* node



DialogueAct	Greeting	Comment	Request	Accept	Reject
Unassigned	1	1	0.05	0.05	0.05
Request	0	0	0.85	0.05	0.05
Accept	0	0	0.05	0.85	0.05
Reject	0	0	0.05	0.05	0.85

Figure 5.56: CPT of *Mouth* node



Greeting	0.2
Comment	0.2
Request	0.2
Accept	0.2
Reject	0.2

Figure 5.57: CPT of *DialogueAct* node

As shown in Figures 5.53 and 5.57, the *Speech* and *DialogueAct* nodes in the Bayesian network in Figure 5.52 have five states: (1) *Greeting*, (2) *Comment*, (3) *Request*, (4) *Accept* and (5) *Reject*. Figures 5.54 – 5.56 show that the remaining nodes in the Bayesian network have four states: (1) *Unassigned*, (2) *Request*, (3) *Accept* and (4) *Reject*. In order to simplify the Bayesian network, the Request state represents both requests and questions, the latter being a request for more information. The Bayesian network can resolve ambiguity that occurs in the speech input by considering the beliefs associated with voice intonation and facial expressions of the user.

An example of ambiguity that can occur in the ‘intelligent travel agent’ application domain is where the user says “OK” in response to the system utterance, “A seven night stay in Venice would be great this time of year”. Here the utterance “OK” has three possible interpretations: (1) the user wants to go to Venice, i.e., the dialogue act is *Accept*, (2) the user wants more details on the trip to Venice, i.e., the utterance “OK” constitutes a *Request* dialogue act, or (3) the user is just considering the agent’s suggestion, i.e., the dialogue act is *Comment*. Another example is where the user says ‘right’ in response to a suggestion made by the agent. Again, this could be either an acceptance of a proposition, a request for further information or a comment. In both these situations, recognition of the speech input alone is not sufficient for resolving the ambiguity. In these cases the voice intonation of the user and, to a lesser degree, the image processing of facial gestures facilitate resolution of ambiguity.

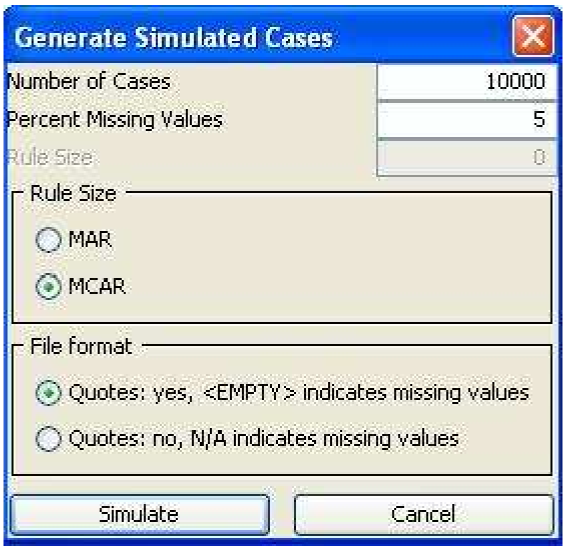
5.7.6. Parametric learning

Suppose an ‘intelligent interviewer’ multimodal system is being trained to recognise the emotional state (e.g., happy, nervous, confused, defensive) of a person during an interview based on voice intonation, facial expression, posture and body language. Assume that, initially, a team of experts were consulted by decision engineers during the design of the ‘intelligent interviewer’ and that a Bayesian network has been created that models relationships between the voice intonation (I), facial expression (FE), posture (P), body language (BL) and emotional state (ES) of the interviewee. In order to refine the decision-making accuracy of the ‘intelligent interviewer’, a Wizard-of-Oz experiment is undertaken in the form of 100 live interviews. The same team of experts who assisted the decision engineers in designing the Bayesian network now monitor live video of the interviews and are asked to make judgements on the emotional states of the interviewees at various stages in the interview. As a result of this process a number of large data files are created containing each expert’s interpretation of the person’s voice intonation, facial expression, posture and body language at various stages throughout the interview. For each such set of interpretations, the experts also make a judgement on the emotional state of the interviewee at that exact time, based on their multimodal interpretations. A subset of an expert’s data file is shown in Figure 5.58. Finally, all the individual data files from the experts are combined into one complete data set.

Parametric learning, i.e., Estimation-Maximum (EM), is now performed to learn the parameters, or the CPTs, of the Bayesian network. Adaptive and EM learning in Hugin were discussed in greater detail in Chapter 3, Section 3.11.5. The CPTs of the Bayesian network are now updated to more accurately model the decision-making of the team of experts. In order to confirm the correctness of the new Bayesian network it is possible to generate a case set of data in the Hugin GUI. This is done by selecting *File | Simulate Cases* which opens the Generate Simulated Cases window, as shown in Figure 5.59.

```
I, FE, P, BL, ES
unassigned, confused, defensive, neutral, confused
relaxed, happy, relaxed, relaxed, relaxed
confused, confused, defensive, neutral, confused
happy, neutral, happy, relaxed, happy
unassigned, neutral, neutral, open, neutral
neutral, neutral, neutral, closed, neutral
```

Figure 5.58: Section of data file for ‘parametric learning’


Generate Simulated Cases

Number of Cases	10000
Percent Missing Values	5
Rule Size	0

Rule Size: MAR [✓ MCAR]

File format: [✓ Quotes: yes, <EMPTY> indicates missing values] Quotes: no, N/A indicates missing values

Figure 5.59: ‘Generate Simulated Cases’ window

Selecting *Simulate* produces a random set of evidence data that is propagated through the Bayesian network. The resulting generated data file will be of a similar format to that produced by the team of experts when watching the live video of the interviews. The experts can now check this data file to ensure that they agree with the conclusions being reached by the Bayesian network. A better method of evaluating the Bayesian network would be to conduct another Wizard-of-Oz experiment, this time enabling the ‘intelligent interviewer’ to make judgements on the emotional state of the interviewee, and have the team of experts monitor these decisions to ensure their correctness.

5.8. *Summary*

This chapter discussed the implementation of a multimodal distributed platform hub, called MediaHub, which performs Bayesian decision-making over multimodal input/output data. Initially, MediaHub’s architecture and key modules were described. Next, each of MediaHub’s modules including the *Dialogue Manager*, *MediaHub Whiteboard*, *Domain Model* and

Decision-Making Module were discussed in detail. Semantic representation and storage with *MediaHub Whiteboard* was then considered, before the role of Psyclone (Thórisson et al. 2005) in enabling distributed processing within MediaHub was described. Five decision-making layers were outlined, before Hugin (Jensen 1996), which implements Bayesian decision-making in MediaHub, was detailed. MediaHub's approach to multimodal decision-making was demonstrated for six key problems (anaphora resolution, domain knowledge awareness, multimodal presentation, turn-taking, dialogue act recognition and parametric learning) across five application domains (building data, cinema ticket reservation, in-car safety, intelligent agents and emotional state recognition). The next chapter discusses testing and evaluation of MediaHub.

Chapter 6 Evaluation of MediaHub

This chapter details the evaluation of MediaHub. First, the test environment in terms of the hardware and software specification of the machines on which MediaHub is tested are outlined. Next, the preliminary testing of MediaHub, with NetBeans IDE, Hugin GUI and Psyclone's *psyProbe* is outlined. Then, the results of testing MediaHub on six key problems in multimodal decision-making are discussed: (1) anaphora resolution, (2) domain knowledge awareness, (3) multimodal presentation, (4) turn-taking, (5) dialogue act recognition and (6) parametric learning across five application domains: (1) building data, (2) cinema ticket reservation, (3) in-car safety, (4) intelligent agents and (5) emotional state recognition. Next, the performance and potential scalability of MediaHub is considered. The chapter concludes with a discussion on how MediaHub meets the essential and desirable criteria required for a multimodal distributed platform hub outlined in Chapter 4, Section 4.9.

6.1. *Test environment systems specifications*

MediaHub has been tested on two versions of the Windows Operating System (XP and Vista) and one Linux distribution (Kubuntu). The hardware and software specifications of each test machine are given in Table 6.1.

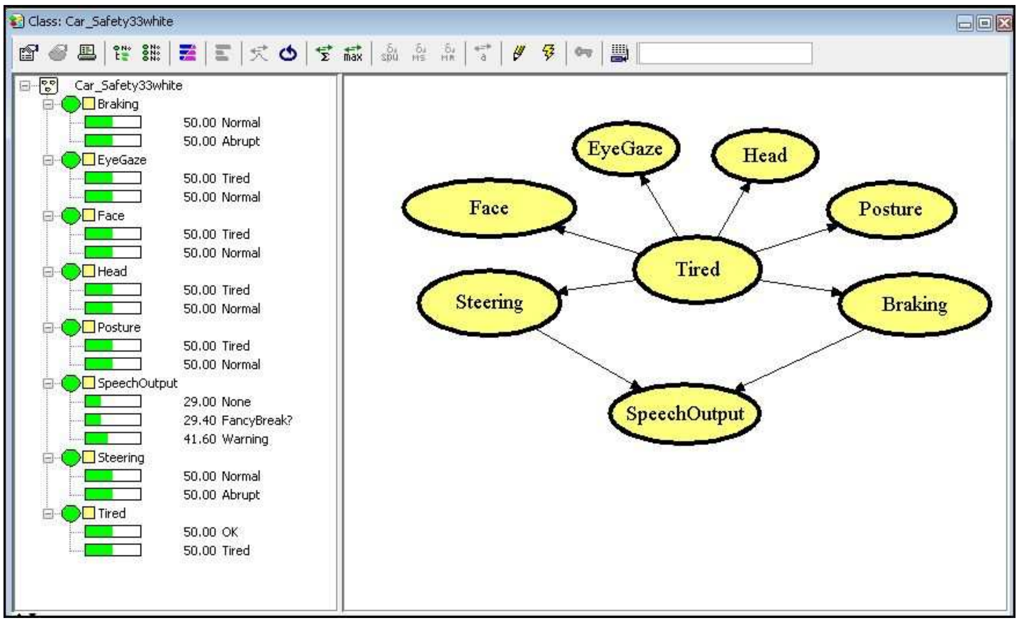
Operating System	*Windows XP*	*Windows Vista*	*Linux (Kubuntu)*
Edition	Professional Version 2002 Service Pack 2	Home Premium Service Pack 1	7.10 (Gutsy Gibbon)
RAM	1024 MB	2046 MB	512 MB
Processor	2.33 GHz	3.20 GHz	2.8 GHz
Version of NetBeans IDE	5.0	5.5.1	5.5.1
Version of Psyclone	1.0.6	1.0.6	1.0.6
Version of Hugin	7.0	7.0	7.0

Table 6.1: Test environment system specifications

6.2. *Initial testing*

As discussed in Chapter 5, Section 5.6, the implementation and testing of MediaHub Bayesian networks is completed in two stages: (1) the qualitative part, i.e., the structure, and quantitative part, i.e., the parameters of the Conditional Probability Tables (CPTs), are defined with the

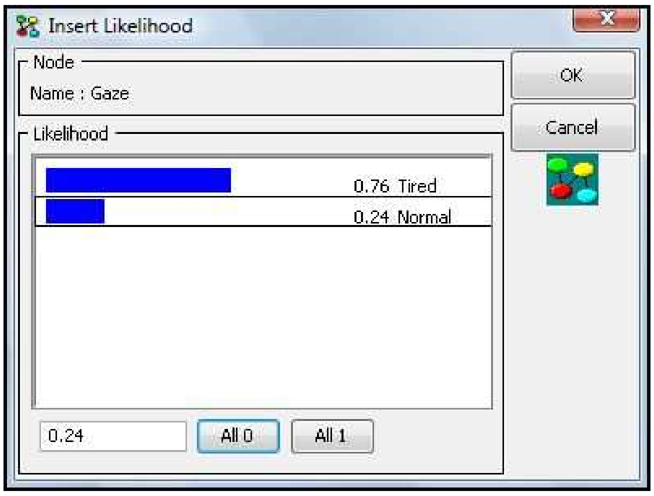
Hugin GUI and (2) the network is accessed and run through the Hugin API. The Hugin GUI runs the Bayesian network and enables viewing of resulting values of each state of every node in the network as shown in Figure 6.1.



```
Car_Safety33white
  Braking
      50.00 Normal
      50.00 Abrupt
  EyeGaze
      50.00 Tired
      50.00 Normal
  Face
      50.00 Tired
      50.00 Normal
  Head
      50.00 Tired
      50.00 Normal
  Posture
      50.00 Tired
      50.00 Normal
  SpeechOutput
      29.00 None
      29.40 FancyBreak?
      41.60 Warning
  Steering
      50.00 Normal
      50.00 Abrupt
  Tired
      50.00 OK
      50.00 Tired
```

Figure 6.1: Hugin GUI deploying Bayesian network

In initial testing of MediaHub a set of test values for each of the states in the Bayesian network is drafted and these are applied to the network by right-clicking on the node in the left pane of the GUI depicted in Figure 6.1. This invokes the *Insert Likelihood* window, as shown in Figure 6.2.



Name : Gaze

0.76 Tired
0.24 Normal

0.24

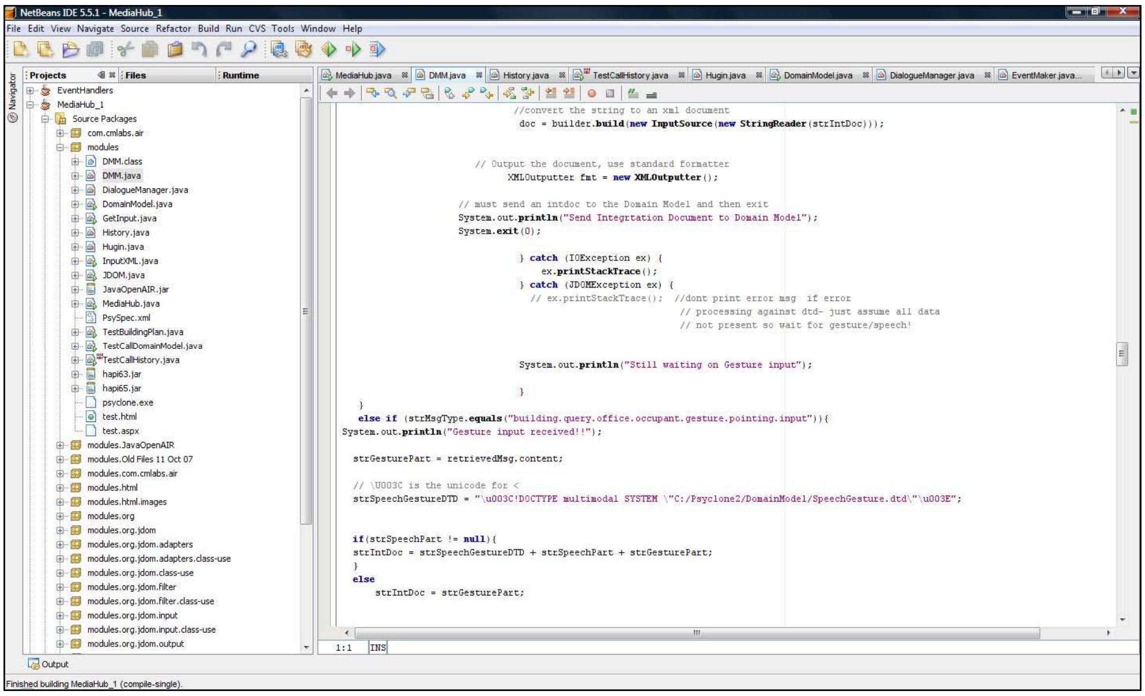
Figure 6.2: Entering evidence on a node through the Hugin GUI

When the evidence is entered, the beliefs on all nodes of the Bayesian network are automatically updated and the resulting values are observed. For each set of test values, the resulting values of the output node are manually recorded in a table, as shown in Table 6.2.

Input Nodes						Output Node			
Node 1		Node 2		Node 3		Hugin GUI		Hugin API	
State 1	State 2	State 1	State 2	State 1	State 2	State 1	State 2	State 1	State 2

Table 6.2: Generic structure of initial testing results table

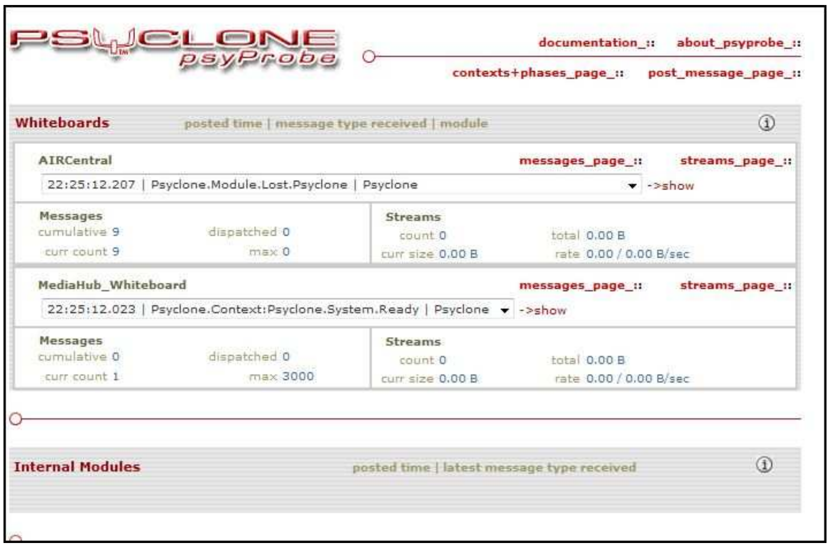
The next stage of initial testing is to open and run the Bayesian network via the Hugin API. This step is performed with NetBeans IDE, as shown in Figure 6.3. The resulting values of the nodes in the Bayesian network are then recorded, e.g., in a results table as illustrated in Table 6.2, and compared to the values obtained by the network running through the Hugin GUI. Hence, the initial testing of MediaHub ensures that identical results are achieved when the Bayesian network is accessed with the Hugin API and when it is run in the Hugin GUI.



```
                    //convert the string to an xml document
                     doc = builder.build(new InputSource(new StringReader(strIntDoc)));


                //  Output the document, use standard formatter
                   XMLOutputter fmt = new XMLOutputter();

              // must send an intdoc to the Domain Model and then exit
              System.out.println("Send Integrtation Document to Domain Model");
              System.exit(0);

                     } catch (IOException ex) {
                        ex.printStackTrace();
                     } catch (JDOMException ex) {
                       // ex.printStackTrace();  //dont print error msg  if error
                                                  // processing against dtd- just assume all data
                                                  // not present so wait for gesture/speech!



                     System.out.println("Still waiting on Gesture input");

                     }
    }
    else if (strMsgType.equals("building.query.office.occupant.gesture.pointing.input")){
System.out.println("Gesture input received!!");

  strGesturePart = retrievedMsg.content;

  // \U003C is the unicode for <
  strSpeechGestureDTD = "\u003C!DOCTYPE multimodal SYSTEM \"C:/Psyclone2/DomainModel/SpeechGesture.dtd\"\u003E";


  if(strSpeechPart != null){
  strIntDoc = strSpeechGestureDTD + strSpeechPart + strGesturePart;
  }
  else
      strIntDoc = strGesturePart;
```

Figure 6.3: NetBeans IDE

Psyclone’s *psyProbe*, shown in Figure 6.4, also facilitates initial testing of MediaHub. The *Post Message* page of *psyProbe* facilitates checking that messages of a certain type are being correctly routed to MediaHub modules subscribed to that message type. The *Post Messages* page of the *psyProbe* is shown in Figure 6.5. The *Post Message* page enables the developer to define the sender, e.g., *Dialogue Manager*, receiver, e.g., *MediaHub Whiteboard*, message type and XML content of the message. The *psyProbe Whiteboard Messages* page can then confirm that these messages have been posted to *MediaHub Whiteboard*, as shown in Figure 6.6. Additional information on each message can be viewed by selecting the message in the *Messages* page. For example, Figure 6.7 shows more information on a message of type *building.query.office.occupant.gesture.deictic.input*. The *psyProbe* thus proves very useful in the testing of MediaHub, particularly since it enables specific parts of MediaHub's processing to be quickly tested, i.e., functionality that has been previously tested, such as the integration of speech and gesture input into a single *IntDoc*, can be bypassed by posting the complete *IntDoc* from the *Decision-Making Module* to *MediaHub Whiteboard* using the *Post Message* page of Psyclone *psyProbe*.



documentation_:: about_psyprobe_::
contexts+phases_page_:: post_message_page_::

Whiteboards — posted time | message type received | module

AIRCentral — messages_page_:: streams_page_::
22:25:12.207 | Psyclone.Module.Lost.Psyclone | Psyclone ->show

Messages
cumulative 9 dispatched 0
curr count 9 max 0

Streams
count 0 total 0.00 B
curr size 0.00 B rate 0.00 / 0.00 B/sec

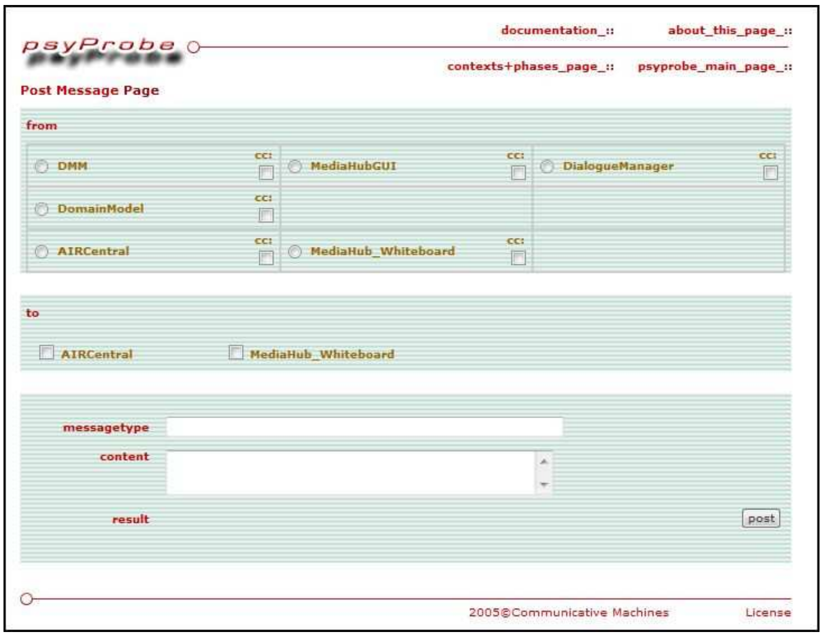
MediaHub_Whiteboard — messages_page_:: streams_page_::
22:25:12.023 | Psyclone.Context:Psyclone.System.Ready | Psyclone ->show

Messages
cumulative 0 dispatched 0
curr count 1 max 3000

Streams
count 0 total 0.00 B
curr size 0.00 B rate 0.00 / 0.00 B/sec

Internal Modules — posted time | latest message type received

Figure 6.4: Psyclone's *psyProbe* for testing MediaHub



documentation_:: about_this_page_::
contexts+phases_page_:: psyprobe_main_page_::

Post Message Page

from
DMM cc: | MediaHubGUI cc: | DialogueManager cc:
DomainModel cc:
AIRCentral cc: | MediaHub_Whiteboard cc:

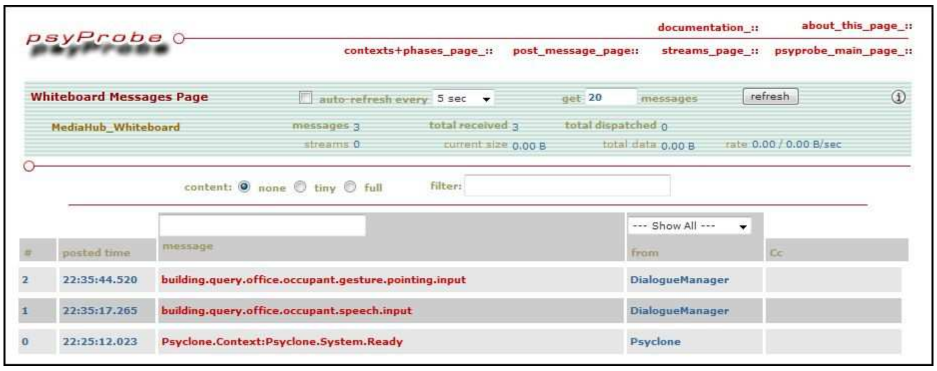
to
AIRCentral | MediaHub_Whiteboard

messagetype
content
result

post

2005©Communicative Machines License

Figure 6.5: *psyProbe Post Message* page



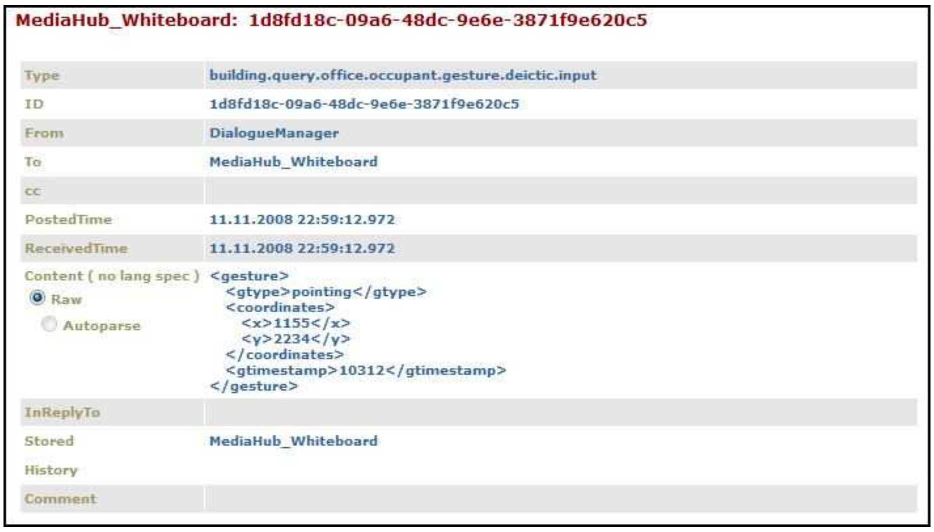
Whiteboard Messages Page auto-refresh every 5 sec get 20 messages

MediaHub_Whiteboard messages 3 total received 3 total dispatched 0
streams 0 current size 0.00 B total data 0.00 B rate 0.00 / 0.00 B/sec

content: [✓ none] tiny full filter:

#	posted time	message	from	Cc
2	22:35:44.520	building.query.office.occupant.gesture.pointing.input	DialogueManager	
1	22:35:17.265	building.query.office.occupant.speech.input	DialogueManager	
0	22:25:12.023	Psyclone.Context:Psyclone.System.Ready	Psyclone	

Figure 6.6: *psyProbe Whiteboard Messages* page


MediaHub_Whiteboard: 1d8fd18c-09a6-48dc-9e6e-3871f9e620c5

Type	building.query.office.occupant.gesture.deictic.input
ID	1d8fd18c-09a6-48dc-9e6e-3871f9e620c5
From	DialogueManager
To	MediaHub_Whiteboard
cc	
PostedTime	11.11.2008 22:59:12.972
ReceivedTime	11.11.2008 22:59:12.972
Content (no lang spec) [✓ Raw] Autoparse	`<gesture> <gtype>pointing</gtype> <coordinates> <x>1155</x> <y>2234</y> </coordinates> <gtimestamp>10312</gtimestamp> </gesture>`
InReplyTo	
Stored	MediaHub_Whiteboard
History	
Comment	

Figure 6.7: Viewing more information on a message with *psyProbe*

6.3. *Evaluation of MediaHub*

The evaluation of MediaHub focuses on decision-making scenarios from the six key problem areas within five application domains, as discussed in Chapter 5, Section 5.7. This section considers evaluation of MediaHub's performance with respect to decision-making in each of the six key problem areas.

6.3.1. Anaphora resolution

Decision-making with regard to anaphora resolution is demonstrated in the 'building data' application domain. The 'anaphora resolution' example, as discussed in Chapter 5, Section 5.7.1, focuses on MediaHub's capability of using dialogue history during the course of a multimodal dialogue. To demonstrate the process of evaluation of MediaHub in this application domain we will consider the sequence of turns below:

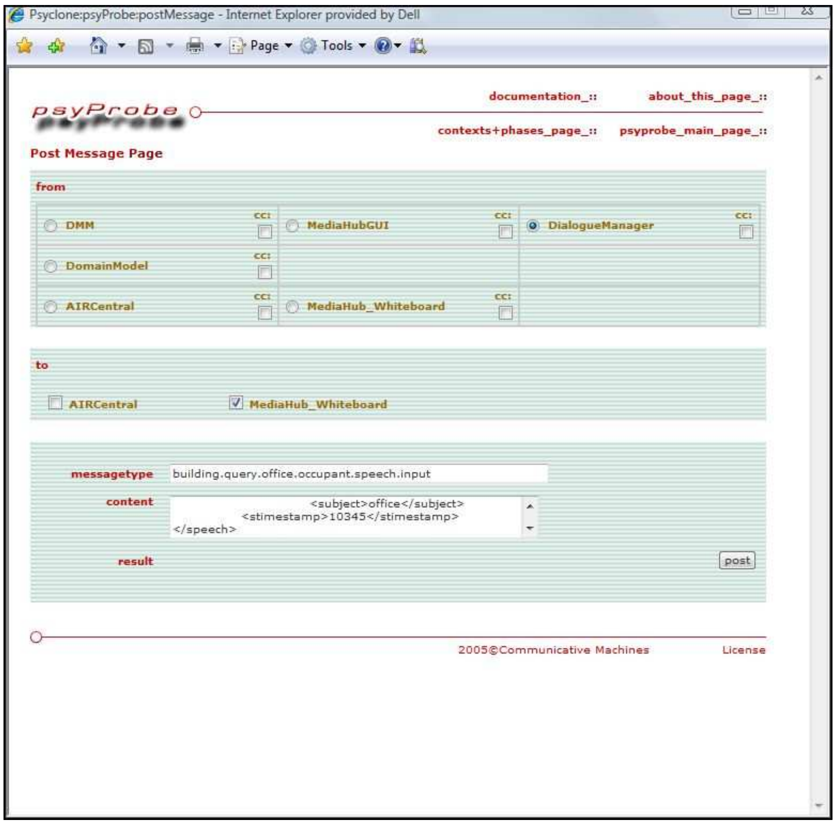
1 U: Whose office is this [→]?
2 S: That's Paul's office.
3 U: Ok. Whose office is that [→]?
4 S: That's Sheila's office.

5 U: Show me the route from her office to this [→] office.

First, the speech semantics of turn 1 is posted from the *Dialogue Manager* to *MediaHub Whiteboard* in a message of type:

building.query.office.occupant.speech.input

This is executed with the *Post Message* page of *psyProbe*, as shown in Figure 6.8.



Psyclone:psyProbe:postMessage - Internet Explorer provided by Dell

psyProbe

documentation_:: about_this_page_::
contexts+phases_page_:: psyprobe_main_page_::

Post Message Page

from

○ DMM cc: ☐	○ MediaHubGUI cc: ☐	● DialogueManager cc: ☐
○ DomainModel cc: ☐		
○ AIRCentral cc: ☐	○ MediaHub_Whiteboard cc: ☐	

to

☐ AIRCentral ☑ MediaHub_Whiteboard

messagetype: building.query.office.occupant.speech.input

content:
```
                        <subject>office</subject>
                <stimestamp>10345</stimestamp>
</speech>
```

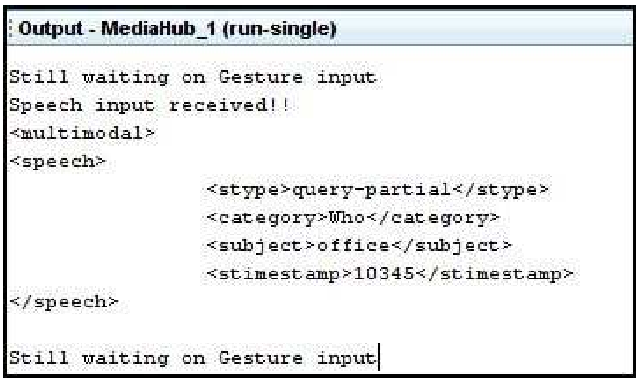
result

post

2005©Communicative Machines License

Figure 6.8: Sending speech segment from *Dialogue Manager* to *MediaHub Whiteboard*

MediaHub Whiteboard is configured in the *psySpec* to automatically deliver messages of type *building.query** to the *Decision-making Module*. NetBeans' output window, as shown in Figure 6.9, can then enable checking if the speech segment has been received by the *Decision-Making Module*.



Output - MediaHub_1 (run-single)

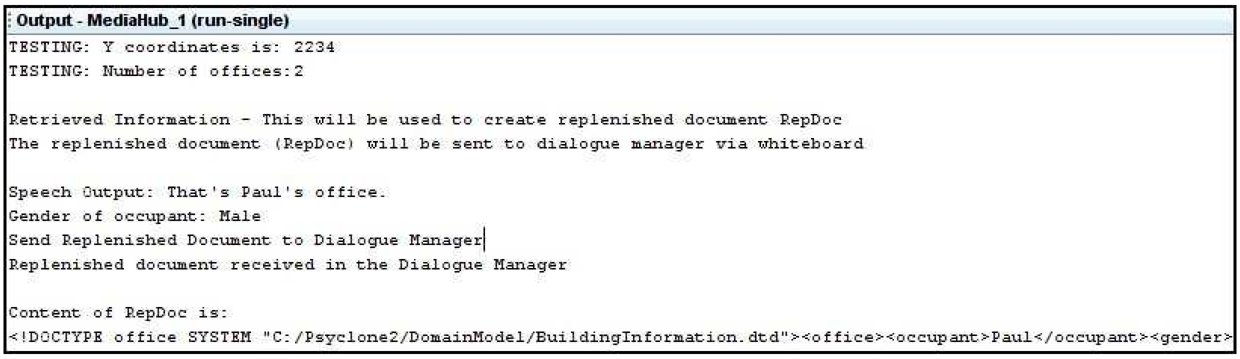
```
Still waiting on Gesture input
Speech input received!!
<multimodal>
<speech>
                <stype>query-partial</stype>
                <category>Who</category>
                <subject>office</subject>
                <stimestamp>10345</stimestamp>
</speech>

Still waiting on Gesture input
```

Figure 6.9: NetBeans' output window confirming speech input received

Next, the corresponding semantics of the deictic gesture in turn 1 is posted from the *Dialogue Manager* to *MediaHub Whiteboard* with a message of type:

building.query.office.occupant.gesture.deictic.input

Again the *Post Message* page in *psyProbe*, as shown in Figure 6.8, enables posting of the message to *MediaHub Whiteboard*. The NetBeans output window confirms that the corresponding deictic gesture semantics has been received in the *Decision-Making Module* and that a replenished document (*RepDoc*) has been created and sent to the *Dialogue Manager*, as shown in Figure 6.10.



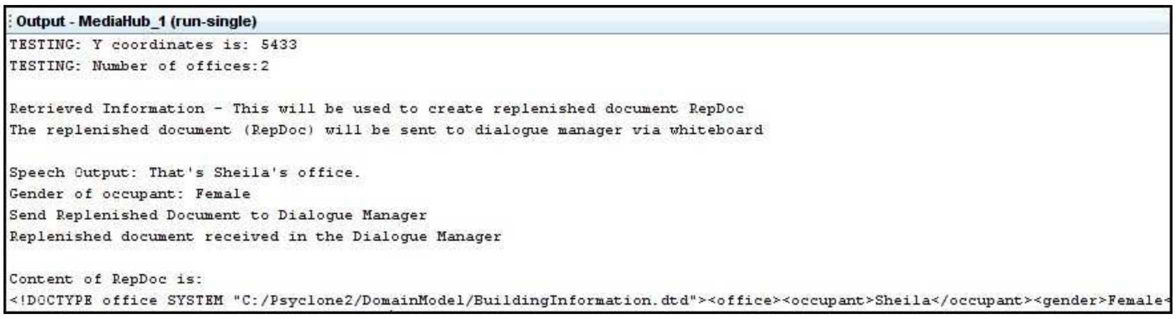
```
Output - MediaHub_1 (run-single)
TESTING: Y coordinates is: 2234
TESTING: Number of offices:2

Retrieved Information - This will be used to create replenished document RepDoc
The replenished document (RepDoc) will be sent to dialogue manager via whiteboard

Speech Output: That's Paul's office.
Gender of occupant: Male
Send Replenished Document to Dialogue Manager
Replenished document received in the Dialogue Manager

Content of RepDoc is:
<!DOCTYPE office SYSTEM "C:/Psyclone2/DomainModel/BuildingInformation.dtd"><office><occupant>Paul</occupant><gender>
```

Figure 6.10: *RepDoc* received in *Dialogue Manager*

The *RepDoc* contains data obtained from the *Domain Model*, e.g., the occupant’s name, gender and office number. This data is processed in the *Dialogue Manager* to enable the system to respond with turn 2 (*That’s Paul’s office.*). Next, the speech and gesture semantics of turn 3 (*Ok. Whose office is that [→]?)* is posted from the *Dialogue Manager* to *MediaHub Whiteboard* with messages of the following respective types:

- *building.query.office.occupant.speech.input*
- *building.query.office.occupant.gesture.deictic.input*

As with turn 1, both messages are posted to *MediaHub Whiteboard* with the *Post Message* page of *psyProbe* shown in Figure 6.8 which results in the two XML segments being combined and the *Domain Model* being queried for office data. The resulting NetBeans output window trace is shown in Figure 6.11.



```
Output - MediaHub_1 (run-single)
TESTING: Y coordinates is: 5433
TESTING: Number of offices:2

Retrieved Information - This will be used to create replenished document RepDoc
The replenished document (RepDoc) will be sent to dialogue manager via whiteboard

Speech Output: That's Sheila's office.
Gender of occupant: Female
Send Replenished Document to Dialogue Manager
Replenished document received in the Dialogue Manager

Content of RepDoc is:
<!DOCTYPE office SYSTEM "C:/Psyclone2/DomainModel/BuildingInformation.dtd"><office><occupant>Sheila</occupant><gender>Female<
```

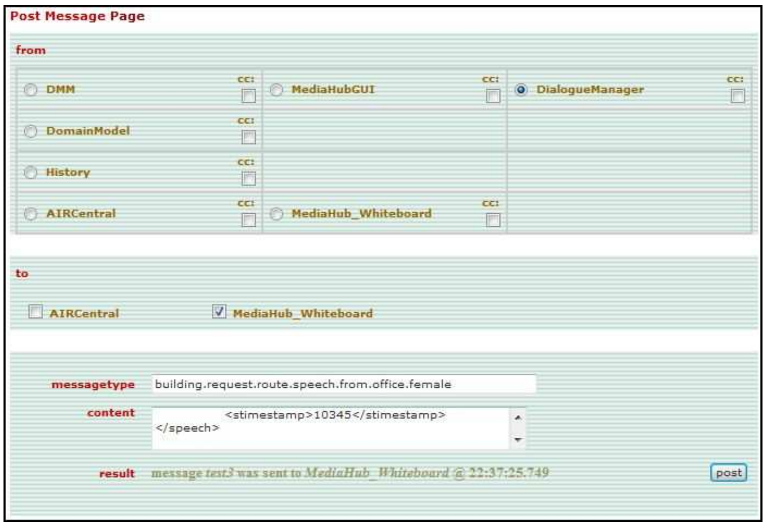
Figure 6.11: Output trace for turn 3 of ‘anaphora resolution’

Again, the *RepDoc* sent from the *Domain Model* to the *Dialogue Manager* processes turn 4 (*That's Sheila's office*.). The speech segment for turn 5 is shown in Figure 6.12.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE speech SYSTEM "C:/Psyclone2/DomainModel/SpeechGender.dtd">
<speech>
<type>request-partial</type>
<category>show-route</category>
<subcategory>from-to</subcategory>
<subject>office</subject>
<gender>female</gender>
<stimestamp>10345</stimestamp>
</speech>
```

Figure 6.12: Speech segment for turn 5

This XML segment in Figure 6.12 is posted from *Dialogue Manager* to *MediaHub Whiteboard* through the *Post Message* page of *psyProbe* as shown in Figure 6.13. The speech segment is packaged in a message of type *building.request.route.speech.from.office.female* and all messages of this type posted on *MediaHub Whiteboard* are automatically delivered to the *Decision-Making Module*.


Post Message Page

from

○ DMM cc: ☐	○ MediaHubGUI cc: ☐	● DialogueManager cc: ☐
○ DomainModel cc: ☐		
○ History cc: ☐		
○ AIRCentral cc: ☐	○ MediaHub_Whiteboard cc: ☐	

to

☐ AIRCentral ☑ MediaHub_Whiteboard

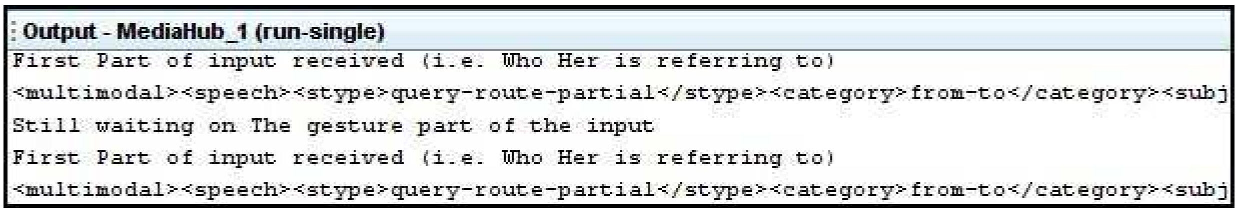
messagetype building.request.route.speech.from.office.female

content <stimestamp>10345</stimestamp>
</speech>

result message test3 was sent to MediaHub_Whiteboard @ 22:37:25.749

Figure 6.13: Posting the speech segment of turn 5 to *MediaHub Whiteboard*

As shown in Figure 6.14, the NetBeans output window confirms that the first component of the input has been received in the *Decision-Making Module* and that MediaHub is waiting on the corresponding deictic gesture.


Output - MediaHub_1 (run-single)

```
First Part of input received (i.e. Who Her is referring to)
<multimodal><speech><stype>query-route-partial</stype><category>from-to</category><subj
Still waiting on The gesture part of the input
First Part of input received (i.e. Who Her is referring to)
<multimodal><speech><stype>query-route-partial</stype><category>from-to</category><subj
```

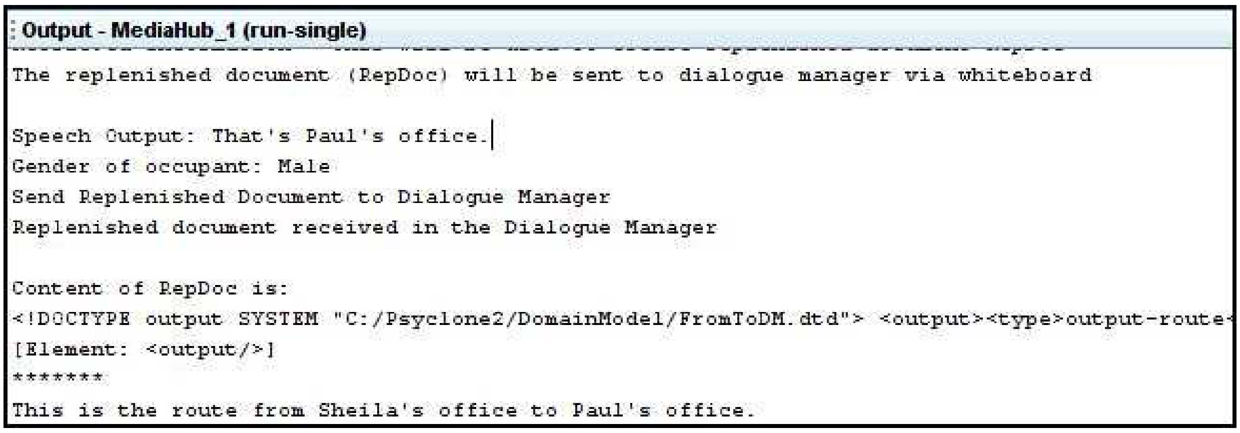
Figure 6.14: First part of turn 5 received in *Decision-Making Module*

Next, the semantics of the corresponding deictic gesture, shown in Figure 6.15, is posted to *MediaHub Whiteboard* in a message of type *building.request.route.gesture.to.office*.

```
<gesture>
<gtype>pointing-to-office</gtype>
<coordinates>
            <x>1155</x>
            <y>2234</y>
</coordinates>
<gtimestamp>12150</gtimestamp>
</gesture>
</multimodal>
```

Figure 6.15: Semantics of deictic gesture for turn 5

The semantics of the deictic gesture is combined with that of the speech segment and the *IntDoc* is sent to the *Domain Model* via *MediaHub Whiteboard*. A *RepDoc* is then created in the *Domain Model* and forwarded to the *Dialogue Manager*. As shown in Figure 6.16, the *RepDoc* contains a text to speech string that is printed to the NetBeans output window to confirm the correct operation of MediaHub.



```
The replenished document (RepDoc) will be sent to dialogue manager via whiteboard

Speech Output: That's Paul's office.
Gender of occupant: Male
Send Replenished Document to Dialogue Manager
Replenished document received in the Dialogue Manager

Content of RepDoc is:
<!DOCTYPE output SYSTEM "C:/Psyclone2/DomainModel/FromToDM.dtd"> <output><type>output-route
[Element: <output/>]
*******
This is the route from Sheila's office to Paul's office.
```

Figure 6.16: Final output trace for ‘anaphora resolution’

Hence, a combination of Psyclone’s *psyProbe* and the NetBeans IDE output window have facilitated testing the various stages of processing in the ‘anaphora resolution’ example and have confirmed MediaHub’s capabilities in this problem area.

6.3.2. Domain knowledge awareness

In the ‘domain knowledge awareness’ example discussed in Chapter 5, Section 5.5.2, MediaHub integrates the semantics of speech and eye-gaze input together with domain-specific information and dialogue history to determine which movie from a list the user wishes to reserve cinema tickets for. The Bayesian network implemented to demonstrate MediaHub’s application in this problem area is given in Figure 6.17.

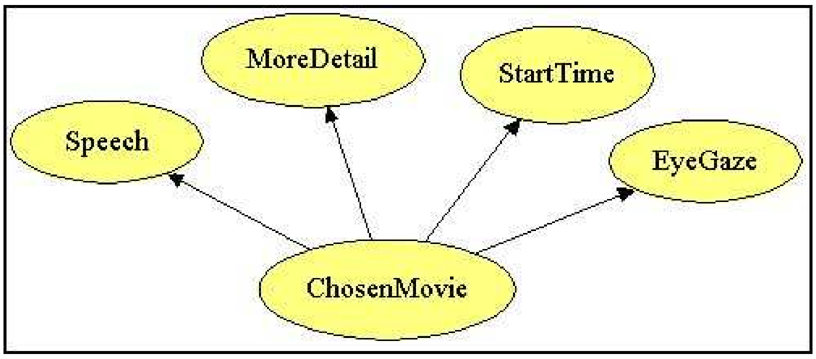
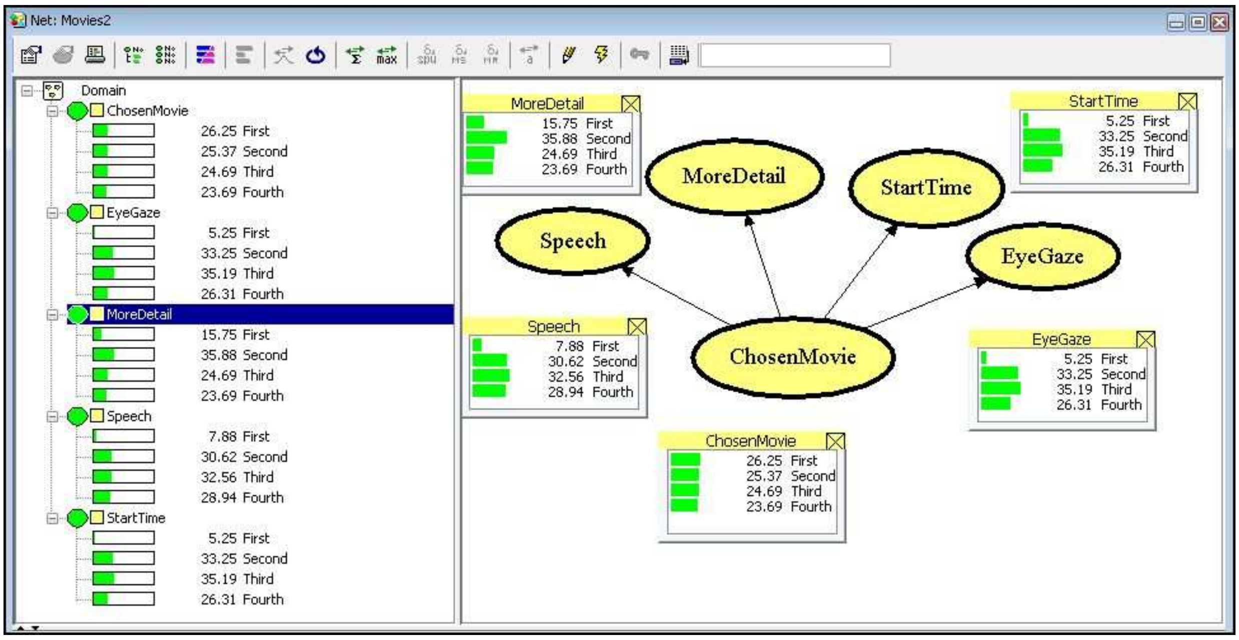


Figure 6.17: Bayesian network for 'domain knowledge awareness'

As shown in Figure 6.17, the Bayesian network for this example has two nodes that receive multimodal input, *Speech* and *EyeGaze*, and two nodes that are populated as a result of accessing dialogue history on *MediaHub Whiteboard*, *MoreDetail* and *StartTime*. The initial stage of testing is completed with the Hugin GUI, as shown in Figure 6.18.

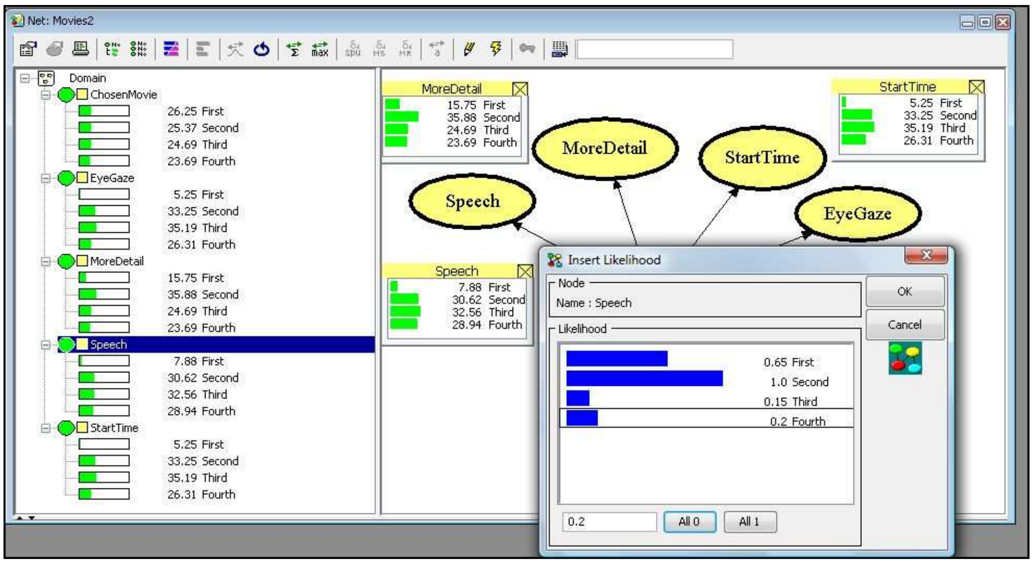


```
Domain
  ChosenMovie
    26.25 First
    25.37 Second
    24.69 Third
    23.69 Fourth
  EyeGaze
     5.25 First
    33.25 Second
    35.19 Third
    26.31 Fourth
  MoreDetail
    15.75 First
    35.88 Second
    24.69 Third
    23.69 Fourth
  Speech
     7.88 First
    30.62 Second
    32.56 Third
    28.94 Fourth
  StartTime
     5.25 First
    33.25 Second
    35.19 Third
    26.31 Fourth
```

Figure 6.18: Testing of 'domain knowledge awareness' Bayesian network in Hugin GUI

Initial testing is part of the iterative process of Bayesian network construction discussed in Chapter 3, Section 3.6 which includes four steps performed repeatedly until the required functionality has been received: (1) design, (2) implementation, (3) testing and (4) analysis. As shown in Figure 3.4 of Chapter 3, the testing phase involves running test scenarios with known outcomes. To facilitate this testing, a table of test cases was drafted to contain different combinations of inputs and their corresponding expected outputs. A subset of this table is given in Table 6.3. The complete test case table is given in Appendix D. Table 6.3 has five columns for each node in the Bayesian network and one column for checking if the expected results have been achieved. Note that the numbers *1-4* refer to the states of each node in the Bayesian network, i.e., first, second, third, fourth which in turn relate to the first, second, third and fourth movie in the list presented to the user. *T* (true) and *F* (false) represent whether the user has, or has not, asked for more details or inquired about the start time of a movie in the list.

The evidence contained in the test case table shown in Table 6.3 is entered into the Bayesian network at run-time, as shown in Figure 6.19.



```
Domain
  ChosenMovie
    26.25 First
    25.37 Second
    24.69 Third
    23.69 Fourth
  EyeGaze
    5.25 First
    33.25 Second
    35.19 Third
    26.31 Fourth
  MoreDetail
    15.75 First
    35.88 Second
    24.69 Third
    23.69 Fourth
  Speech
    7.88 First
    30.62 Second
    32.56 Third
    28.94 Fourth
  StartTime
    5.25 First
    33.25 Second
    35.19 Third
    26.31 Fourth
```

```
Node
Name : Speech
Likelihood
0.65 First
1.0 Second
0.15 Third
0.2 Fourth
0.2
```

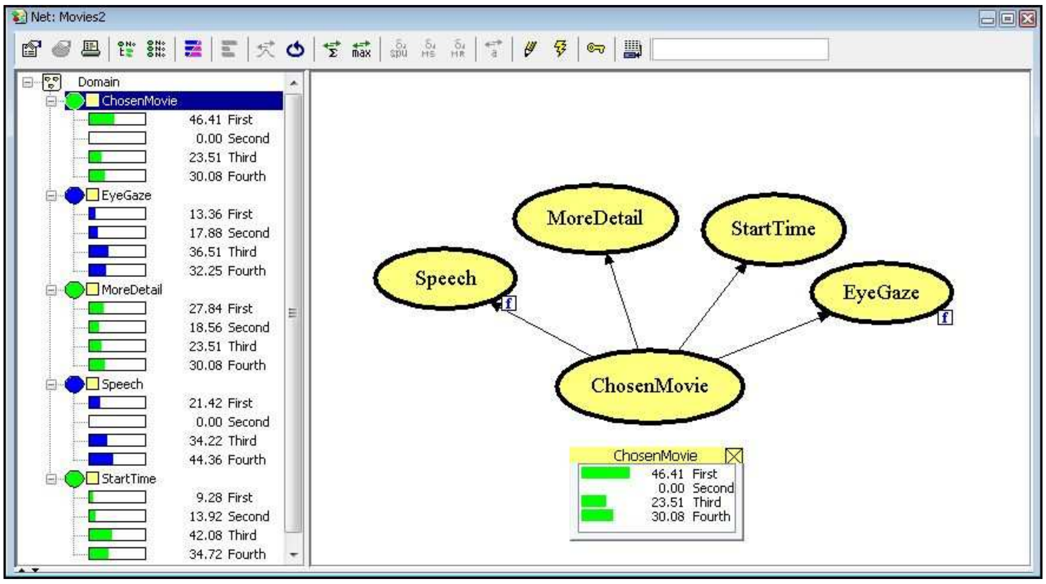
Figure 6.19: Testing of 'domain knowledge awareness' Bayesian network

Speech				MoreDetail				StartTime				EyeGaze				ChosenMovie				OK?
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	
65	0	15	20	T	F	F	F	F	F	F	F	55	45	0	0	95	3	1	1	Yes
66	0	0	34	T	F	F	F	T	F	F	T	80	20	0	0	83	0	0	17	No – % of 1 should be > 83
74	0	26	0	T	F	F	F	T	F	F	F	76	24	0	0	89	11	0	0	No – % of 1 should be > 89

Table 6.3: Subset of test cases for 'domain knowledge awareness' Bayesian network

If the desired results are not observed, the parameters of the CPTs of the nodes in the Bayesian network are adjusted and the test evidence is re-propagated through the network. This iterative process of design, implementation, testing and analysis continues until the Bayesian network correctly models the decision-making in the application domain.

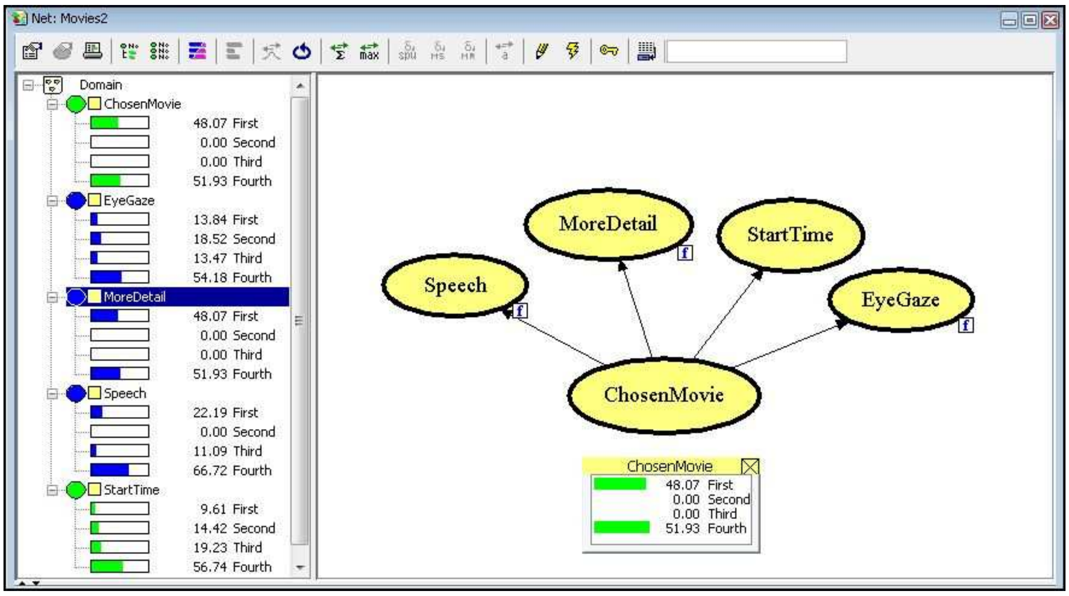
To demonstrate ambiguity resolution in this example, consider the following scenario. Based on the semantics of speech input, MediaHub is 40% certain that the user wants to view the first movie in the list, 20% certain that the user wants to view the second movie in the list and 40% certain that the user wants to view the fourth movie. Based on the semantics produced by a gaze-tracking module, MediaHub is 27%, 33%, 18% and 12% certain that the user is focusing on the first, second, third and fourth movies in the list respectively. Applying this evidence to the Bayesian network produces beliefs of 46.41%, 0%, 23.51% and 30.08% in the *First*, *Second*, *Third* and *Fourth* states of the *ChosenMovie* node respectively, as shown in Figure 6.20.



```
Domain
  ChosenMovie
     46.41 First
      0.00 Second
     23.51 Third
     30.08 Fourth
  EyeGaze
     13.36 First
     17.88 Second
     36.51 Third
     32.25 Fourth
  MoreDetail
     27.84 First
     18.56 Second
     23.51 Third
     30.08 Fourth
  Speech
     21.42 First
      0.00 Second
     34.22 Third
     44.36 Fourth
  StartTime
      9.28 First
     13.92 Second
     42.08 Third
     34.72 Fourth

ChosenMovie
     46.41 First
      0.00 Second
     23.51 Third
     30.08 Fourth
```

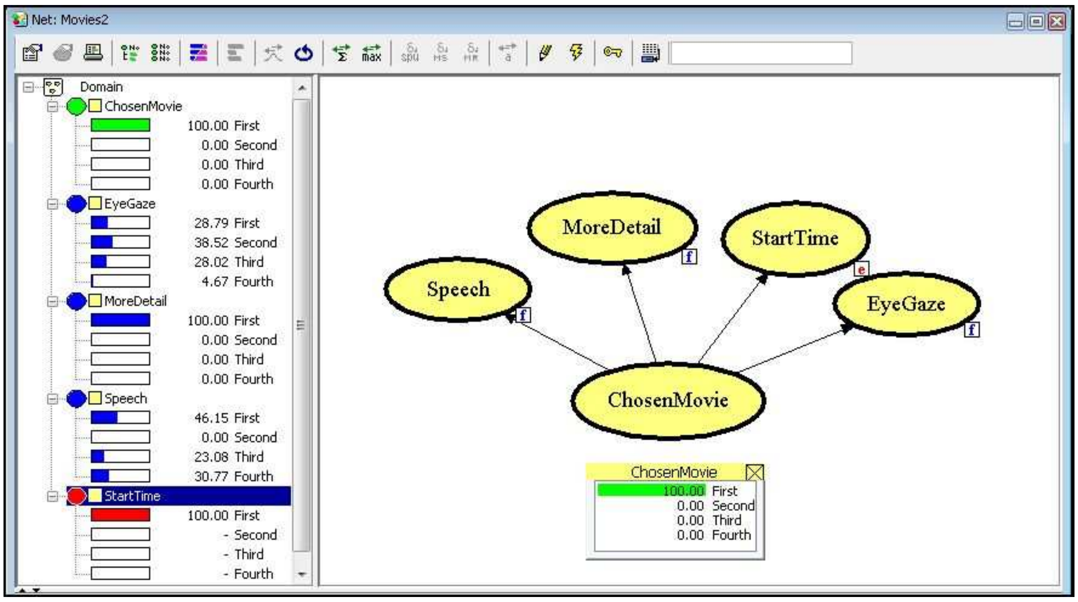
Figure 6.20: Evidence applied to the *Speech* and *EyeGaze* nodes

The *Decision-Making Module* applies rule-based decision-making to decide if the winning hypothesis is greater than 50% and at least 20% more than the closest competing hypothesis. Currently this is not the case and therefore a decision on which movie the user wants to view cannot be made without seeking clarification from the user. Now assume that data from the domain model facilitates updating the *MoreDetails* node, i.e., the user asked for more details on both the first and second movie. This causes a belief of 50% being applied to both the *First* and the *Second* states of the *MoreDetails* node. The *First* and *Fourth* states of the *ChosenMovie* node are updated to 48.07% and 51.93%, as shown in Figure 6.21. Next, information from *MediaHub Whiteboard* that the user previously only asked about the start time of the first movie is applied to the Bayesian network, i.e., the belief in the *First* state of the *StartTime* node is updated to 100%. This causes MediaHub to believe with absolute certainty that the user wishes to reserve tickets to see the first movie, as illustrated in Figure 6.22.



```
Net: Movies2
Domain
  ChosenMovie
     48.07 First
      0.00 Second
      0.00 Third
     51.93 Fourth
  EyeGaze
     13.84 First
     18.52 Second
     13.47 Third
     54.18 Fourth
  MoreDetail
     48.07 First
      0.00 Second
      0.00 Third
     51.93 Fourth
  Speech
     22.19 First
      0.00 Second
     11.09 Third
     66.72 Fourth
  StartTime
      9.61 First
     14.42 Second
     19.23 Third
     56.74 Fourth

ChosenMovie
  48.07 First
   0.00 Second
   0.00 Third
  51.93 Fourth
```

Figure 6.21: Further evidence applied to *Speech* and *EyeGaze* nodes

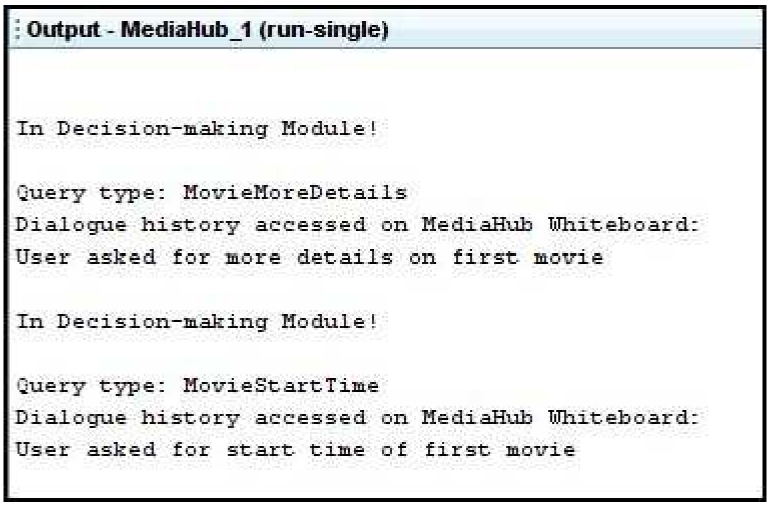


```
Net: Movies2
Domain
  ChosenMovie
    100.00 First
      0.00 Second
      0.00 Third
      0.00 Fourth
  EyeGaze
     28.79 First
     38.52 Second
     28.02 Third
      4.67 Fourth
  MoreDetail
    100.00 First
      0.00 Second
      0.00 Third
      0.00 Fourth
  Speech
     46.15 First
      0.00 Second
     23.08 Third
     30.77 Fourth
  StartTime
    100.00 First
         - Second
         - Third
         - Fourth

ChosenMovie
  100.00 First
    0.00 Second
    0.00 Third
    0.00 Fourth
```

Figure 6.22: Evidence applied on the *Speech* and *EyeGaze* nodes

Hence, a combination of the semantics of multimodal inputs and information from dialogue history facilitates resolving ambiguity on the intentions of the user. As discussed in Chapter 5, Section 5.7.2, domain-specific information is accessed to determine which movies are currently being shown in the cinema and are therefore presented to the user.

For testing access to the Bayesian network in MediaHub via the Hugin API, the semantics of previous queries requesting more details on, and the start times of, certain movies is first posted to *MediaHub Whiteboard* through the *Post Message* page of *psyProbe*. Then, again using *psyProbe*, the marked up XML semantics of the speech and eye-gaze is posted to *MediaHub Whiteboard*. The output window in NetBeans IDE verifies that the data posted to *MediaHub Whiteboard* has been received by the correct MediaHub modules and that the processing within each module is correct. An example output trace from testing in NetBeans IDE is shown in Figure 6.23.



```
Output - MediaHub_1 (run-single)


In Decision-making Module!

Query type: MovieMoreDetails
Dialogue history accessed on MediaHub Whiteboard:
User asked for more details on first movie

In Decision-making Module!

Query type: MovieStartTime
Dialogue history accessed on MediaHub Whiteboard:
User asked for start time of first movie
```

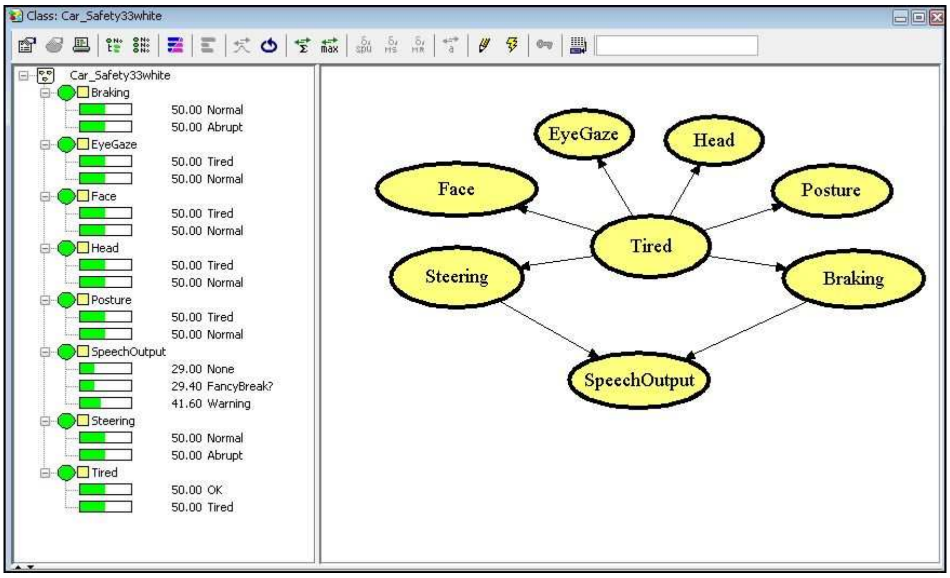
Figure 6.23: NetBeans output trace for 'domain knowledge awareness'

The test scenarios in Table 6.3 facilitate creating the marked-up semantics posted to *MediaHub Whiteboard*. The results of propagating the test evidence held in Table 6.3 is output in NetBeans IDE and checked against the results obtained with the Hugin GUI. To summarise, the key steps in testing the 'domain knowledge awareness' example are:

- Post semantics of dialogue history to *MediaHub Whiteboard* using *psyProbe*.
- Post semantics of speech input to *MediaHub Whiteboard*.
- Post semantics of eye-gaze input to *MediaHub Whiteboard*.
- Confirm that the messages have been successfully delivered to appropriate modules through the output window in NetBeans IDE.
- Output the conclusions reached by the Bayesian network, i.e., values of the states in the *ChosenMovie* node.
- Check that these results are correct by cross referencing against those obtained when running the Bayesian network with the Hugin GUI.

6.3.3. Multimodal presentation

The 'multimodal presentation' example discussed in Chapter 5, Section 5.7.3, demonstrates MediaHub's use in addressing the problem of multimodal presentation in an in-car safety application domain. In this example, the semantics of facial expression, eye-gaze, head and posture input are considered along with the inputs derived from monitoring of the steering and braking behaviour of the driver. The Bayesian network for demonstrating MediaHub's application to this problem area is shown in Figure 6.24.



```
Class: Car_Safety33white

Car_Safety33white
  Braking
    50.00 Normal
    50.00 Abrupt
  EyeGaze
    50.00 Tired
    50.00 Normal
  Face
    50.00 Tired
    50.00 Normal
  Head
    50.00 Tired
    50.00 Normal
  Posture
    50.00 Tired
    50.00 Normal
  SpeechOutput
    29.00 None
    29.40 FancyBreak?
    41.60 Warning
  Steering
    50.00 Normal
    50.00 Abrupt
  Tired
    50.00 OK
    50.00 Tired
```

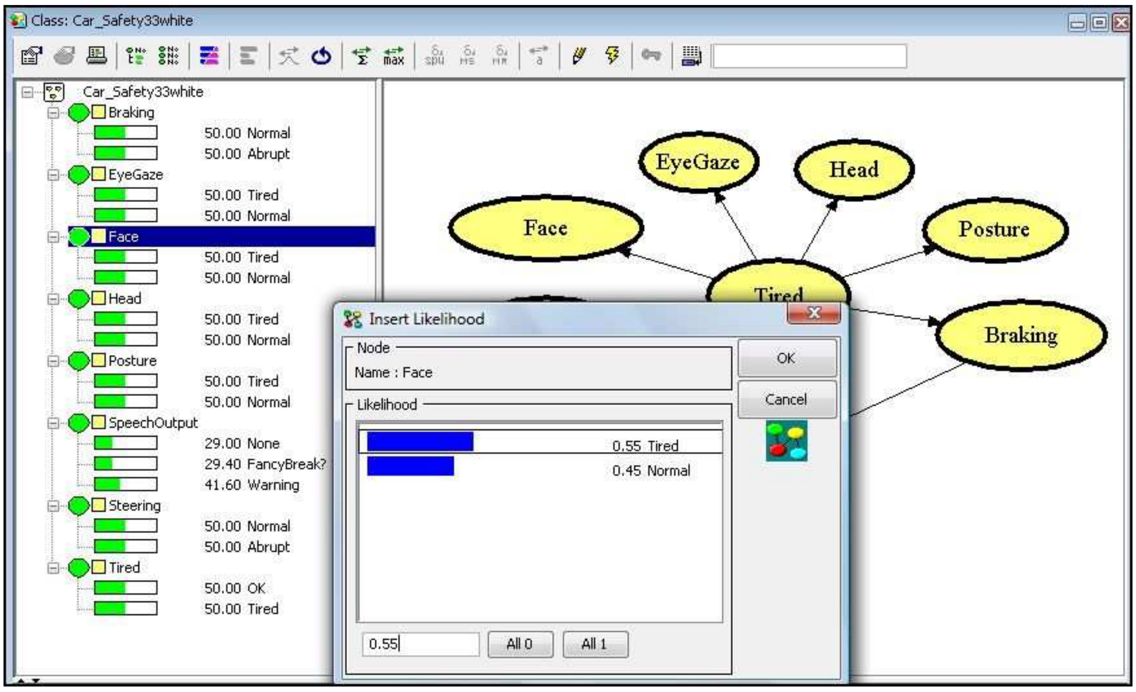
Figure 6.24: Bayesian network for 'multimodal presentation'

As with all Bayesian networks developed in MediaHub, the initial stage of testing is performed with the Hugin GUI and a table of test cases facilitates verification of correct operation of the Bayesian network. A subset of this table is presented in Table 6.4. See Appendix D for the full test case table.

Steering		Braking		Face		EyeGaze		Head		Posture		Tired		SpeechOutput			OK?
N	A	N	A	N	T	N	T	N	T	N	T	N	T	N	F	W	
88	12	85	15	51	49	55	45	19	81	78	22	85	15	90	5	5	No, value of None state of SpeechOutput should be lower
10	90	35	65	50	50	45	55	34	66	55	45	23	77	3	20	77	Yes – but try few more test cases like this!
5	95	30	70	55	45	45	55	40	60	60	40	25	75	2	18	80	Yes

Table 6.4: Subset of test cases for 'multimodal presentation' Bayesian network

Table 6.4 has eight columns, one for each of the nodes in the Bayesian network and one field for recording whether or not correct results have been obtained. Note that the letters *N* and *L* relate to the *Normal* and *Abrupt* states in the *Steering* and *Braking* nodes of the Bayesian network. Letters *N* and *T* in the *Face*, *EyeGaze*, *Head*, *Posture* and *Tired* nodes represent the belief that the driver looks tired or normal. Note that *N* in the *SpeechOutput* node represents that the in-car safety system should not issue any speech output, *F* represents the *FancyBreak?* state, whilst *W* represents the belief that the system should issue a spoken warning to the driver. As with the previous example, the 'multimodal presentation' Bayesian network is tested with test scenarios in the Hugin GUI with known, or expected, outcomes. For example, the process of entering the first row of data in Table 6.4 is shown in Figure 6.25. The left pane in the Hugin GUI window shows the values of states in all nodes of the Bayesian network.



```
Class: Car_Safety33white

Car_Safety33white
  Braking
      50.00 Normal
      50.00 Abrupt
  EyeGaze
      50.00 Tired
      50.00 Normal
  Face
      50.00 Tired
      50.00 Normal
  Head
      50.00 Tired
      50.00 Normal
  Posture
      50.00 Tired
      50.00 Normal
  SpeechOutput
      29.00 None
      29.40 FancyBreak?
      41.60 Warning
  Steering
      50.00 Normal
      50.00 Abrupt
  Tired
      50.00 OK
      50.00 Tired

Insert Likelihood
  Node
    Name : Face
  Likelihood
    0.55 Tired
    0.45 Normal
  0.55   All 0   All 1
  OK   Cancel
```

Figure 6.25: Entering test evidence into 'multimodal presentation' Bayesian network

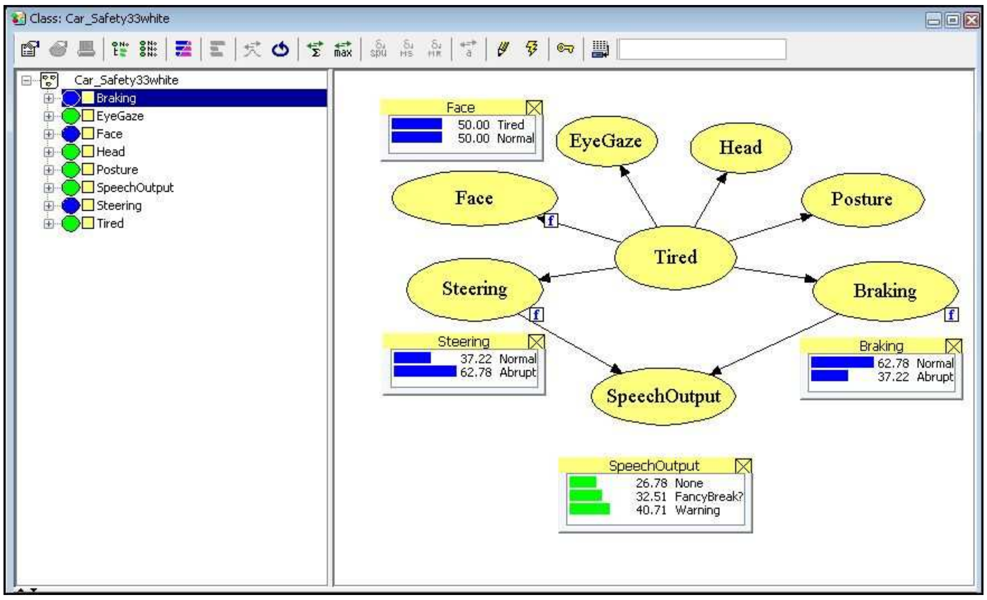
Following each iterative test of the Bayesian network, the *Tired* and *SpeechOutput* fields of Table 6.4 are updated. The results are then analysed and the *OK?* field is completed. Again, the CPTs of the nodes in the Bayesian network are updated based on the analysis of the Bayesian network's performance and the testing is continued until satisfactory performance has been achieved.

An example of ambiguity resolution in this example, is where steering is abrupt (*Normal*: 35, *Abrupt*: 65), braking is normal (*Normal*: 65, *Abrupt*: 35) and the drivers facial expression suggest with a confidence of 50% that the driver is tired. Applying these parameters to the Bayesian network does not reach a conclusion with a sufficient degree of confidence, as shown in Figure 6.26. As shown in the *SpeechOutput* monitor window in Figure 6.26, applying this evidence to the Bayesian network produces the following results in the states of the *SpeechOutput* node:

- *None*: 26.78%
- *FancyBreak?*: 32.51%

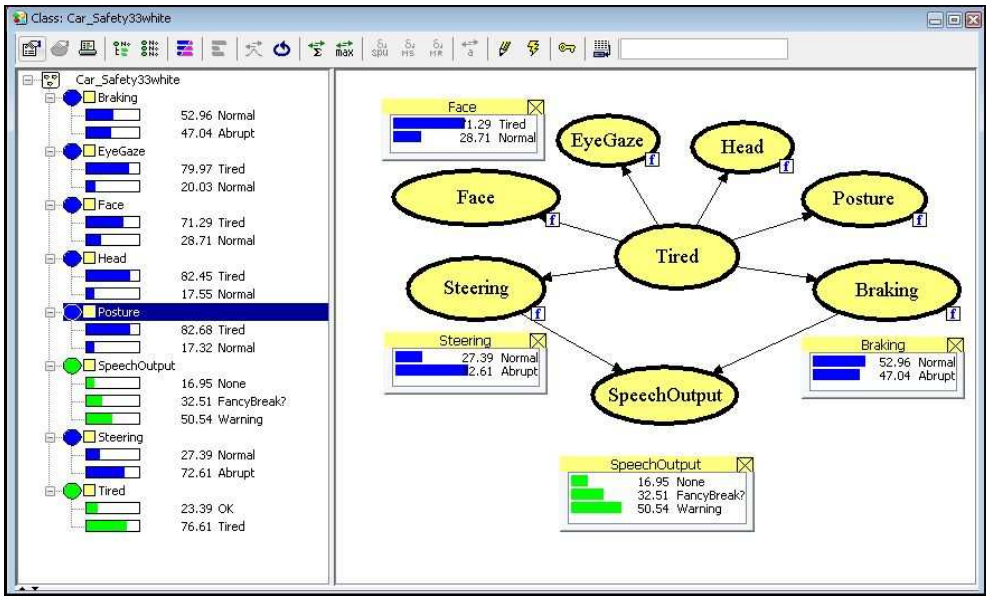
- *Warning*: 40.71%

If we now add evidence to the *EyeGaze* node, i.e., the driver looks tired with a belief of 74%, the *Head* node, i.e., the driver is tired with a belief of 78% and the *Posture* node, i.e., the driver is tired with a belief of 80%, the ambiguity is reduced as illustrated in Figure 6.27.



```
Face
50.00 Tired
50.00 Normal

Steering
37.22 Normal
62.78 Abrupt

Braking
62.78 Normal
37.22 Abrupt

SpeechOutput
26.78 None
32.51 FancyBreak?
40.71 Warning
```

Figure 6.26: Entering test evidence into the ‘multimodal presentation’ Bayesian network

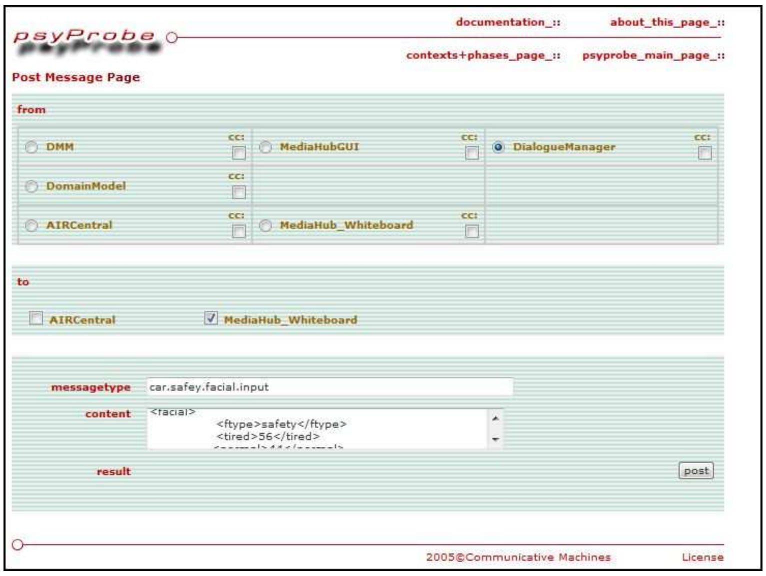


```
Braking
  52.96 Normal
  47.04 Abrupt
EyeGaze
  79.97 Tired
  20.03 Normal
Face
  71.29 Tired
  28.71 Normal
Head
  82.45 Tired
  17.55 Normal
Posture
  82.68 Tired
  17.32 Normal
SpeechOutput
  16.95 None
  32.51 FancyBreak?
  50.54 Warning
Steering
  27.39 Normal
  72.61 Abrupt
Tired
  23.39 OK
  76.61 Tired

Face
1.29 Tired
28.71 Normal

Steering
27.39 Normal
2.61 Abrupt

Braking
52.96 Normal
47.04 Abrupt

SpeechOutput
16.95 None
32.51 FancyBreak?
50.54 Warning
```

Figure 6.27: Entering test evidence into ‘multimodal presentation’ Bayesian network

As shown in Figure 6.27, the belief that a warning has been issued has increased from 40.71% to 50.54%. Also note that as the evidence on each of the nodes is propagated, all nodes are automatically updated to reflect this new evidence and the degree of influence applied from parent nodes. For, example, the *Tired* state of *Head* node was set to 78%, but when the influence of the *Tired* node is taken into account the belief of this state is dynamically updated to 82.45% as shown in Figure 6.27. This dynamic updating of beliefs based on new evidence, and hence the new weighting of influence between the nodes, is true for all nodes in the Bayesian network.

When the Bayesian network is thoroughly tested in the Hugin GUI, the opening, editing and running of the Bayesian network through the Hugin API is then evaluated. Semantics pertaining to the various inputs is posted to *MediaHub Whiteboard* through Psyclone *psyProbe* and the conclusions reached by the Bayesian network are observed in the output window of NetBeans IDE. Figure 6.28 shows the *psyProbe* posting the semantics of the driver's facial expression to from *Dialogue Manager* to *MediaHub Whiteboard*.



Post Message Page

from

○ DMM cc: ☐	○ MediaHubGUI cc: ☐	● DialogueManager cc: ☐
○ DomainModel cc: ☐		
○ AIRCentral cc: ☐	○ MediaHub_Whiteboard cc: ☐	

to

☐ AIRCentral [✓ MediaHub_Whiteboard]

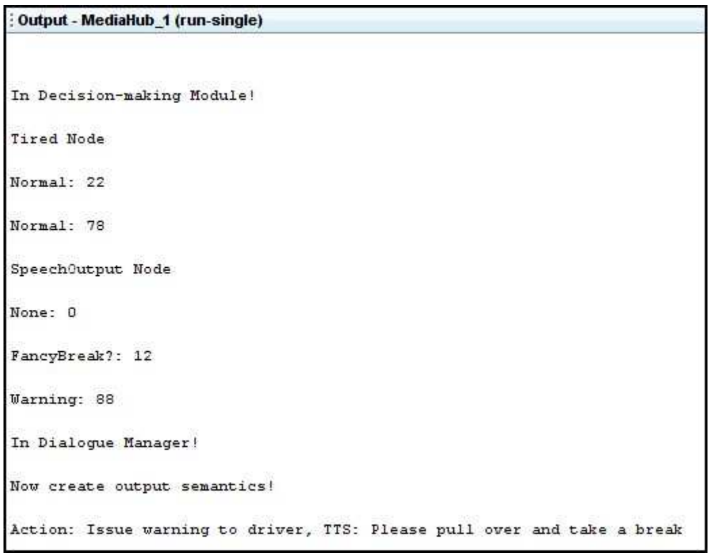
messagetype car.safey.facial.input

content
```
<facial>
            <ftype>safety</ftype>
            <tired>56</tired>
```

result

Figure 6.28: *psyProbe* testing 'multimodal presentation'

Figure 6.29 shows the conclusion reached by the Bayesian network being written to the output window in NetBeans IDE.



```
In Decision-making Module!

Tired Node

Normal: 22

Normal: 78

SpeechOutput Node

None: 0

FancyBreak?: 12

Warning: 88

In Dialogue Manager!

Now create output semantics!

Action: Issue warning to driver, TTS: Please pull over and take a break
```

Figure 6.29: Results of running 'multimodal presentation' Bayesian network

6.3.4. Turn-taking

The Bayesian networks in Figures 6.30 and 6.31 demonstrate MediaHub's application to the problem of turn-taking in intelligent agents. Decision-making within this problem area was discussed in greater detail in Chapter 5, Section 5.7.4.

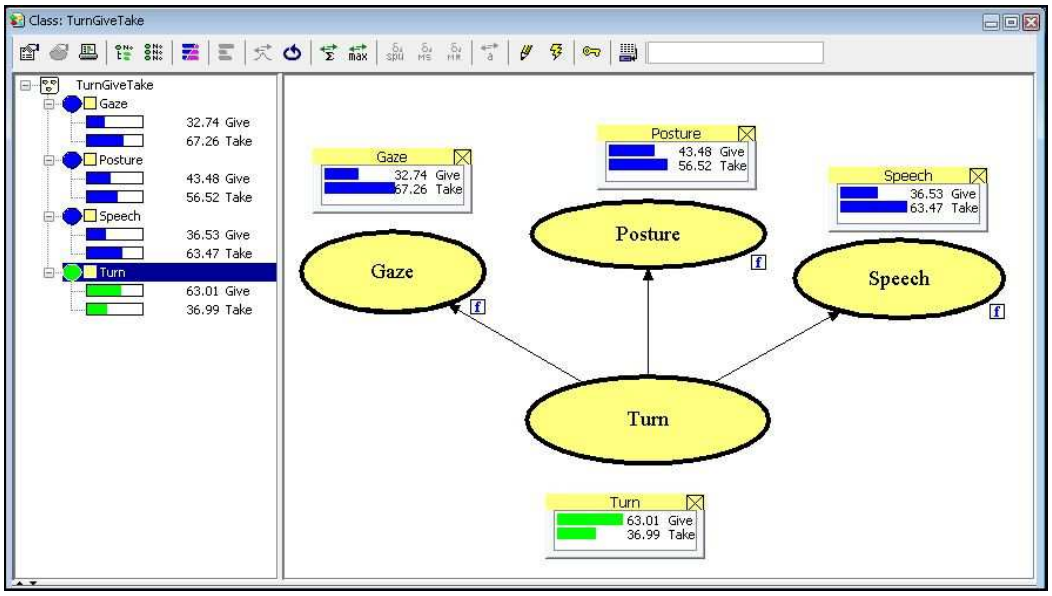


Figure 6.30: 'Turn-taking' Bayesian network in Hugin

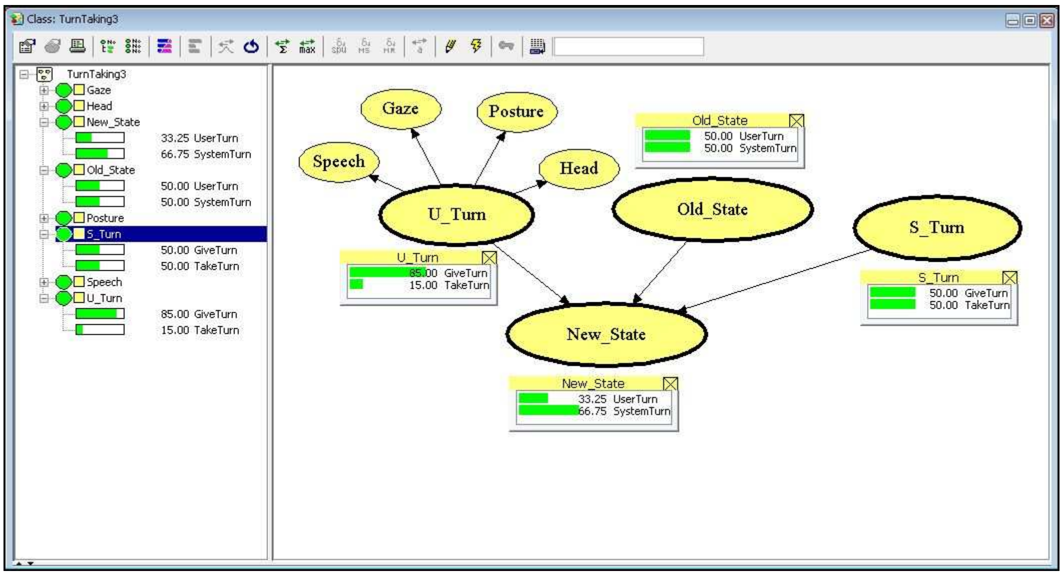


Figure 6.31: Alternative 'turn-taking' Bayesian network

The same approach to testing is applied as in the previous example: (1) the Bayesian networks are thoroughly tested with the Hugin GUI, and (2) access to the Bayesian network in MediaHub via the Hugin GUI is tested with NetBeans IDE and *psyProbe*. Tables 6.5 and 6.6 show subsets of the test case tables for both 'turn-taking' Bayesian networks (See Appendix D for full test case tables).

Gaze		Posture		Speech		Turn		OK?
G	T	G	T	G	T	G	T	
62	38	40	60	60	40	57	43	Yes
55	45	33	67	56	44	46	54	Yes
35	65	52	48	41	59	37	63	Yes

Table 6.5: Subset of test cases for ‘turn-taking’ Bayesian network

Speech		Gaze		Posture		Head		U_Turn		S_Turn		Old_State		New_State		OK?
G	T	G	T	G	T	G	T	GT	TT	GT	TT	U	S	U	S	
45	55	50	50	38	62	30	70	29	71	T	F	F	T	85	15	Yes
74	26	55	45	62	38	49	51	72	28	F	T	T	F	17	83	Yes
91	9	70	30	85	15	70	30	94	6	F	T	T	F	3	97	Yes

Table 6.6: Subset of test cases for alternative ‘turn-taking’ Bayesian network

In Tables 6.5 and 6.6, the letters *G* and *T* represent the states of *Give* and *Take*, *GT* and *TT* are abbreviations of *GiveTurn* and *TakeTurn*, whilst *U* and *S* represent the *UserTurn* and *SystemTurn* of the *New_State* node. Tables 6.5 and 6.6 facilitate testing and refining the decision-making of the Bayesian networks in Figures 6.30 and 6.31 and both networks were considered capable of contributing to the resolution of ambiguity in respect of turn-taking in intelligent agents.

6.3.5. Dialogue act recognition

The problem of dialogue act recognition in the application domain of an ‘intelligent travel agent’ was discussed in Chapter 5, Section 5.7.5. Figure 6.32 illustrates the testing of the Bayesian network implemented for the ‘dialogue act recognition’ example. Table 6.7 presents a subset of test cases used for testing the Bayesian network in Figure 6.32. The complete test case table is given in Appendix D. The testing and evaluation of the Bayesian network in Figure 6.32 confirmed MediaHub’s capability of supporting dialogue act recognition in an intelligent agent.

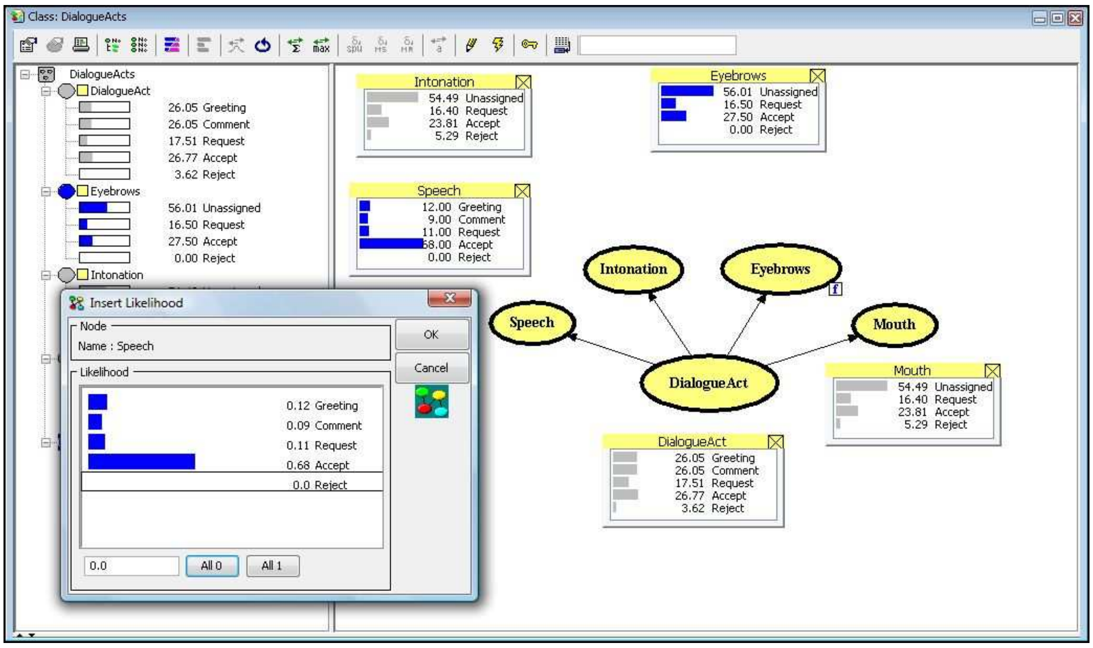


Figure 6.32: Testing of 'dialogue act recognition' Bayesian network

Speech					Intonation				Eyebrows				Mouth				DialogueAct					OK?
G	C	R	A	X	U	R	A	X	U	R	A	X	U	R	A	X	G	C	R	A	X	
95	5	0	0	0	90	0	0	10	90	0	0	10	25	25	25	25	90	10	0	0	0	Yes
0	20	0	80	0	0	0	30	70	0	0	30	70	0	0	50	50	0	0	0	75	25	Yes
0	20	0	80	0	25	25	25	25	0	0	30	70	0	0	50	50	0	0	0	85	15	Yes

Table 6.7: Subset of test cases for 'dialogue act recognition' Bayesian network

6.3.6. Parametric learning

Chapter 5, Section 5.5.6 discussed the example of an 'intelligent interviewer' multimodal system to demonstrate MediaHub's capability of supporting parametric learning with the Hugin GUI. To facilitate the testing of parametric learning the Bayesian network shown in Figure 6.33 was constructed. However, unlike the previous example, the CPTs of the Bayesian network were not altered during its construction. Next, a data file was created containing 100 combinations of inputs to the states of the Bayesian network combined with the corresponding conclusions, i.e., values in the states of the *EmotionalState* node. A section of the data file for learning the parameters of the Bayesian network is shown in Figure 6.34. Note that *I*, *FE*, *P*, *BL* and *ES* in the Bayesian network in Figure 6.34 are abbreviations for intonation, facial expression, posture, body language and emotional state respectively.

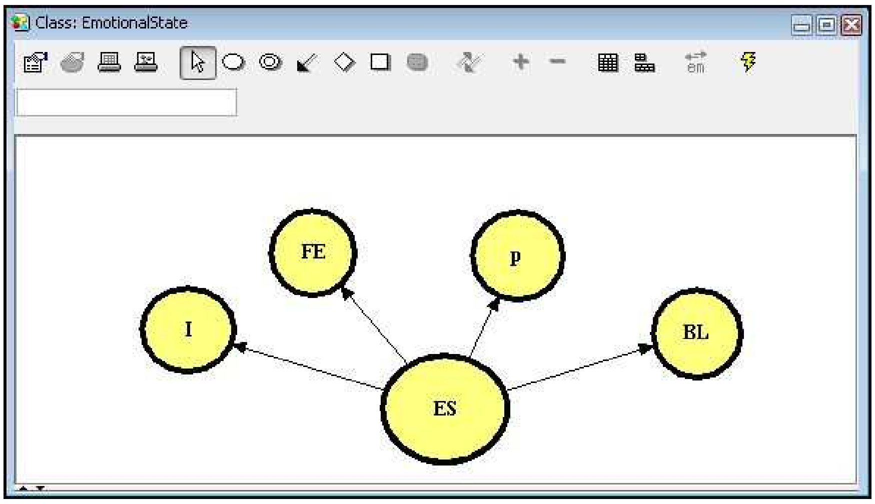
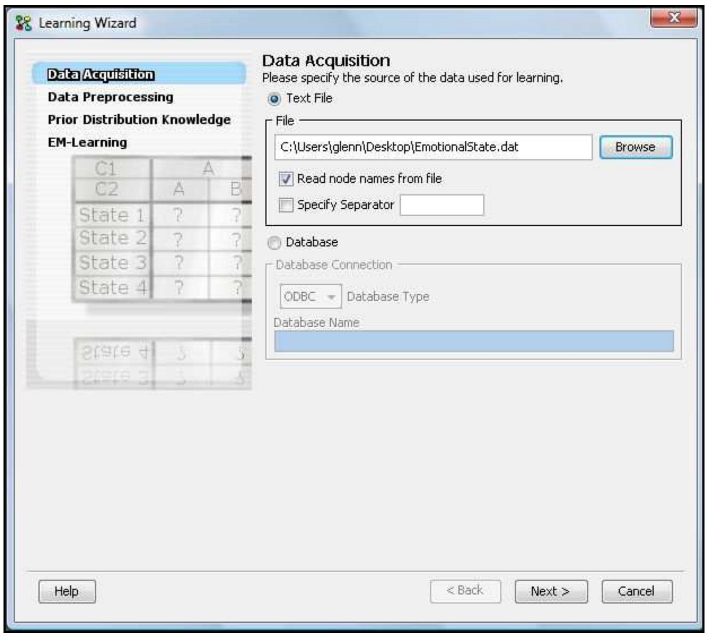


Figure 6.33: Bayesian network for ‘parametric learning’

```
I, FE, P, BL, ES
happy, neutral, happy, relaxed, happy
relaxed, happy, relaxed, relaxed, relaxed
neutral, confused, defensive, neutral, confused
unassigned, neutral, neutral, open, neutral
happy, neutral, neutral, open, neutral
unassigned, confused, defensive, neutral, confused
```

Figure 6.34: Section of data file for ‘parametric learning’

The data file given in Figure 6.34 then facilitates learning the parameters of the Bayesian network and hence modelling the causal relationships between the variables of the data file, as shown in Figure 6.35.



Data Acquisition
Please specify the source of the data used for learning.
(●) Text File
File
C:\Users\glenn\Desktop\EmotionalState.dat
[✓] Read node names from file
[ ] Specify Separator
( ) Database
Database Connection
ODBC Database Type
Database Name

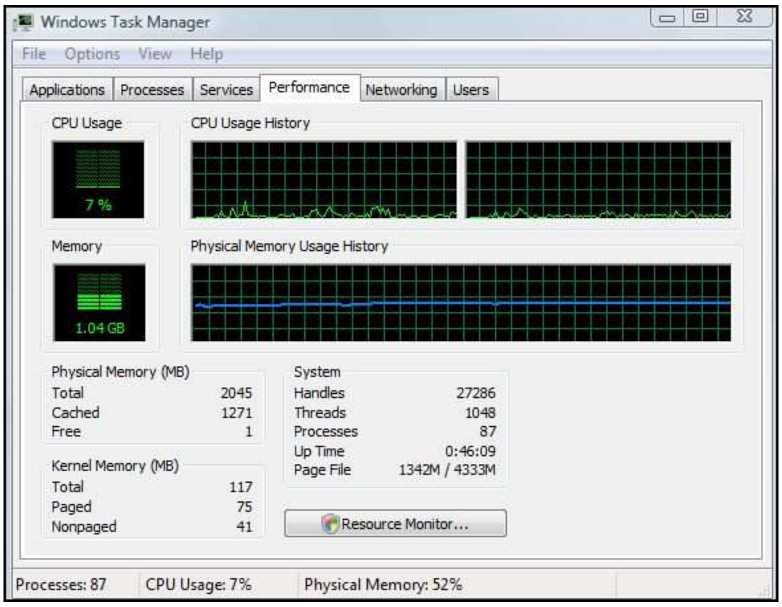
Figure 6.35: Section of data file for ‘parametric learning’

When the parameters of the CPTs in the Bayesian network were learned, the Bayesian network was tested with a test case table as in the previous example. The Bayesian network was then adjusted manually through the Hugin GUI until desired performance was achieved. The Hugin GUI was found to correctly model the relationships between the variables of the data file. However, since the data file was simulated, i.e., not the result of a Wizard-of-Oz experiment as discussed in Section 5.7.6, and it contained just 100 test cases, the resulting Bayesian network required considerable refinement before it was deemed useful for emotional state recognition. This was to be expected, since the data file was created by a non-expert in the field of emotional state recognition. However, the testing did confirm MediaHub's ability to learn the parameters of a Bayesian network from multimodal data.

6.4. *Performance of MediaHub*

The performance of MediaHub obviously has a huge impact on its potential scalability. Since MediaHub constitutes a centralised distributed platform hub, the load on the machine hosting MediaHub will increase proportionally to the number of interacting modules and the frequency of the interactions between modules. The ability of MediaHub to process the semantics of multimodal data in a timely fashion is critical to its applicability within a multimodal system. Temporality, as discussed in Chapter 4, Section 4.4, is hugely significant if intelligent and time critical decisions are to be made during the course of a multimodal dialogue. Psyclone has mechanisms in place that assist temporal management. As observed in Stefánsson et al. (2009, p. 67), "Psyclone does not need to pre-compute the dataflow beforehand but rather manages it dynamically at runtime, optimizing based on priorities of messages and modules". Although MediaHub was tested across six key problem areas and five application domains and could be potentially applied to a number of other problem areas and application domains, it should be noted that MediaHub has yet to be fully tested in a live fully functional multimodal system. MediaHub's performance and scalability will be dependent upon the application domain in which it is deployed. It is therefore difficult to make definitive claims on MediaHub's expected performance in a fully implemented multimodal system. Throughout testing, however, MediaHub's impact on system resources was monitored with Task Manager in the Windows Operating System (see Figure 6.36) and KDE System Guard (KSysGuard) Performance Monitor in Linux (Kubuntu), as shown in Figure 6.37.

MediaHub was found to achieve acceptable levels of performance on all three test environment operating systems, i.e., Windows XP, Windows Vista and Linux (Kubuntu). There was no noticeable difference in performance between the Windows XP and Vista PCs. Nor was MediaHub found to run noticeably faster or more efficiently on the Linux machine. Both speed and impact on system resources was comparable across all three test environment operating systems. However, it is again worth noting that, MediaHub is a testbed distributed platform hub that has yet to be fully implemented within a live multimodal system. It is therefore not possible to draw complete conclusions on its performance and scalability. However, initial testing across six key problem areas and five application domains has produced satisfactory performance results.



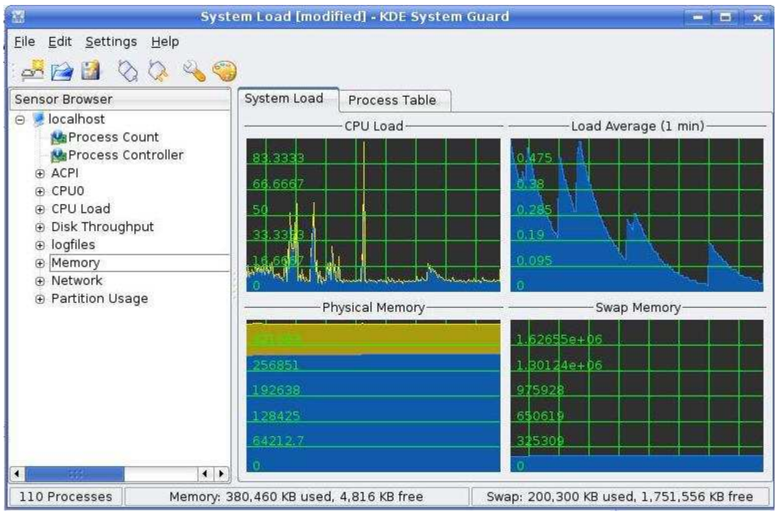
Physical Memory (MB)	
Total	2045
Cached	1271
Free	1

Kernel Memory (MB)	
Total	117
Paged	75
Nonpaged	41

System	
Handles	27286
Threads	1048
Processes	87
Up Time	0:46:09
Page File	1342M / 4333M

Processes: 87 CPU Usage: 7% Physical Memory: 52%

Figure 6.36: Task Manager in Windows Vista



110 Processes Memory: 380,460 KB used, 4,816 KB free Swap: 200,300 KB used, 1,751,556 KB free

Figure 6.37: KSysGuard Performance Monitor in Linux (Kubuntu)

6.5. *Requirements criteria check*

Table 6.8 summarises a check against MediaHub's capabilities against each of the requirements criteria for a multimodal distributed platform hub listed in Chapter 4, Section 4.9. A ✔ symbol indicates full capability and a ✘ symbol denotes partial capability.

Capability	**MediaHub**
E1. Ability to process both multimodal input and output.	✓
E2. Fusion of both input and output semantics.	✓
E3. Representation of semantics on both input and output.	✓
E4. Dynamic updating of belief associated with multimodal input and output.	✓
E5. Distributed processing.	✓
E6. Maintenance of dialogue history.	✓
E7. Current context consideration.	✓
E8. Ambiguity resolution.	✓
E9. Storage of domain-specific information.	✓
E10. Ability to deal with missing data.	✓
E11. Decisions on best combination of output.	✓
E12. Ability to learn from sample data.	✓
D1. Multi-platform.	✗
D2. Ability to learn from experience.	✗
D3. Ability to learn from real data.	✗

Table 6.8: Check on multimodal hub requirements criteria

As shown in Table 6.8, MediaHub offers full capability for each of the essential criteria listed in Chapter 4, Section 4.9. MediaHub is concerned with the processing of multimodal input/output data and with the fusion and storage of input/output semantics. Bayesian networks dynamically update the states of all nodes as new evidence is applied. Psyclone enables distributed processing in MediaHub and the maintenance of dialogue history on the *MediaHub Whiteboard*. The current context is encoded in Bayesian networks applicable to each problem domain. Also, Psyclone offers its own context mechanism for enabling different module behaviour that is context dependant. Ambiguity resolution with different modalities is a key task for MediaHub's decision-making mechanism. The *Domain Model* stores domain-specific information in XML format. As previously mentioned, Bayesian networks are capable of reaching conclusions when some of the relevant inputs to the problem domain are absent. Therefore MediaHub has the capability of dealing with missing data. MediaHub can also make decisions on the optimal combinations for multimodal output.

6.6. *Summary*

This chapter discussed the objective evaluation of MediaHub. The evaluation focused on MediaHub's performance in six key problem areas: (1) anaphora resolution, (2) domain knowledge awareness, (3) multimodal presentation, (4) turn-taking, (5) dialogue act recognition and (6) parametric learning, across five application domains: (1) building data, (2) cinema ticket reservation, (3) in-car safety, (4) intelligent agents and (5) emotional state recognition. The iterative process of design, implementation, testing and analysis of the implemented Bayesian networks was described and the utilisation of Psyclone psyProbe and Hugin GUI for testing Bayesian networks was detailed. The use of NetBeans IDE for verifying access to Bayesian networks via the Hugin GUI was also described. MediaHub's performance and potential scalability was then discussed. Finally, MediaHub was checked against the necessary and sufficient requirements criteria for a distributed multimodal platform hub. MediaHub was found to offer full capability for all essential criteria and partial capability for the remaining three desirable criteria. In summary, based on the evaluation discussed here, MediaHub is considered capable of performing effective Bayesian decision-making in a multimodal distributed platform hub.

Chapter 7 Conclusion and Future Work

Decision-making in multimodal systems includes semantic representation, communication and AI reasoning techniques. This chapter concludes the thesis by first providing a summary of the research completed here. Next, the results are compared to other related work. Finally, there is a discussion on future work and applications of this work.

7.1. *Summary*

In this thesis we have discussed the problems and solutions for decision-making in a multimodal distributed platform hub. These are broadly categorised into three areas: (1) semantic representation and storage, (2) communication and (3) decision-making. The three key objectives of this thesis are: (1) interpretation and generation of multimodal semantic representations, (2) coordination of communication between the modules of a multimodal distributed platform hub and with external modules and (3) performing decision-making with Bayesian networks.

Previous work in the areas of multimodal data fusion and synchronisation, semantic representation and storage, communication, decision-making, distributed processing, multimodal platforms and systems, intelligent multimedia agents, turn-taking in intelligent agents, multimodal corpora and annotation tools, dialogue act recognition and reference resolution was reviewed. A detailed analysis of Bayesian networks was provided, including a discussion of the definition, history and structure of Bayesian networks, intercausal inference, influence diagrams, challenges in Bayesian network construction, Conditional Probability Tables (CPTs), limitations, advantages and applications of Bayesian decision-making, the utilisation of Bayesian networks in multimodal systems and software tools for their implementation.

Having examined the problems and solutions pertinent to multimodal decision-making a Bayesian approach to decision-making in a multimodal distributed platform hub was proposed. This included a discussion on the key problems and the nature of decision-making within multimodal systems, with decisions categorised into two areas relating to: (1) synchronisation of multimodal data and (2) multimodal data fusion. The rationale for MediaHub was discussed by explaining the key advantages of Bayesian networks, i.e., their ability to perform intercausal reasoning, representation of casual dependencies between variables of a problem domain, their compact graphical structure, their ability to represent uncertainty and ambiguity, their tolerance of missing data and their ability to learn. Necessary and sufficient requirements criteria for an intelligent multimodal distributed platform hub and the benefits derived from the application of Bayesian networks in multimodal decision-making were also discussed.

Next, the implementation of a multimodal distributed platform hub, called MediaHub, was discussed. The following four key modules of MediaHub were detailed: (1) *Dialogue Manager*, (2) *MediaHub Whiteboard*, (3) *Decision-Making Module* and (4) *Domain Model*. MediaHub constitutes a publish-subscribe architecture with a central whiteboard for semantic representation. Communication within MediaHub is based on the OpenAIR specification (Mindmakers 2009; Thórisson et al. 2005) and is achieved by exchanging semantic representations between modules via *MediaHub Whiteboard*. The role of Psyclone (Thórisson et al. 2005) which facilitates distributed processing in MediaHub was then described in detail and the Hugin tools (Jensen 1996) for Bayesian decision-making were explained. The role of MediaHub *psySpec* in defining the configuration of MediaHub's modules on invocation was also described.

Bayesian networks were developed through an iterative process of design, implementation, testing and analysis. The four key stages in implementing Bayesian network are: (1) defining the variables of the application domain, (2) understanding the causal relationships between the variables of the domain, (3) determining the structure of a Bayesian network, i.e., the qualitative component, to model the causal relations and (4) eliciting the parameter values of the Bayesian network in the Conditional Probability Tables (CPTs), i.e., the quantitative component. When these four stages are resolved, the actual construction and testing of the Bayesian network in the Hugin GUI is a relatively straightforward task. Five decision-making making layers were outlined: (1) *psySpec* and Contexts, (2) Message Types, (3) Document Type Definitions (DTDs), (4) Bayesian networks and (5) Rule-based.

Multimodal decision-making in MediaHub was demonstrated through worked examples that provide solutions to six key problems in five application domains. The evaluation of MediaHub focused on its performance in six key problem areas for multimodal decision-making: (1) anaphora resolution, (2) domain knowledge awareness, (3) multimodal presentation, (4) turn-taking, (5) dialogue act recognition and (6) parametric learning, across five application domains: (1) building data, (2) cinema ticket reservation, (3) in-car safety, (4) intelligent agents and (5) emotional state recognition. Finally, MediaHub's capabilities were checked against the requirements criteria for a multimodal distributed platform hub.

To sum up, this thesis provides:

(1) A generic approach to Bayesian-based decision-making within a multimodal distributed platform hub.

(2) Applications of Bayesian networks to decision-making for six key problems in multimodal systems: anaphora resolution, domain knowledge awareness, multimodal presentation, turn-taking, dialogue act recognition and learning.

(3) Implementation and evaluation of (1) within MediaHub.

7.2. *Relation to other work*

MediaHub relates to other research within a similar theoretical and practical context including multimodal platforms, multimodal presentation systems, intelligent agents and other multimodal systems as discussed in Chapter 2, Sections 2.8 and 2.9. This section discusses MediaHub in relation to other research.

DARBS (Distributed Algorithmic and Rule-Based System) (Choy et al. 2004a, 2004b; Nolle et al. 2001) proposes the use of rule-based, neural network and genetic algorithm knowledge sources working in parallel around a central blackboard. However, DARBS does not implement or advocate the use of Bayesian networks. Similarities exist between MediaHub and Chameleon (Brøndsted et al. 1998, 2001), discussed in Chapter 2, Section 2.8.1. For example, Chameleon's dialogue manager and Blackboard operate in a similar fashion to MediaHub's *Dialogue Manager* and *MediaHub Whiteboard*. Both implement a domain model and in both communication is achieved by exchanging semantic representation between modules via a semantic storage module. Both Chameleon and MediaHub address the problem of anaphora resolution in a 'building data' application domain. However, Chameleon uses a frame-based method for semantic representation, whilst MediaHub uses XML. Also, domain-specific information in Chameleon is stored in text files linked together in hierarchical linked-list structures with a series of search functions, whilst all domain information in MediaHub is stored in XML format. MediaHub's use of Psyclone for distributed processing compares favourably with DACS (Fink et al. 1996) used for communication in Chameleon. Chameleon performs rule-based decision-making, whilst MediaHub implements Bayesian networks. Additionally, MediaHub implements a Whiteboard using Psyclone which is an extension of the blackboard-based model of semantic storage implemented in Chameleon.

XWand (Wilson & Shafer 2003; Wilson & Pham 2003) is a wireless sensor package enabling natural interaction within intelligent spaces. XWand has a dynamic Bayesian network for action selection within an intelligent space focussing on the home environment. XWand could potentially be applied to select movies from a list on a computer screen as discussed in the 'domain knowledge awareness' example in Chapter 5, Section 5.7.2. However, the hand-held wand would clearly not be suitable for use by the driver in the car environment as considered in the 'multimodal presentation' example in Chapter 5, Section 5.7.3. SmartKom offers a wide range of capabilities in a host of areas important to multimodal systems. However, it does not specifically explore the application of a generic Bayesian approach to decision-making within the hub of a distributed platform. Moreover, the focus of MediaHub is the development of a multimodal distributed platform hub that can be utilised within other multimodal systems. Much multimodal research is concerned with improving the quality of the time a driver spends in a car. Multimodality in the car environment has been considered at length in SmartKom (Wahlster 2006). In Berton et al. (2006) driver interaction with mobile services in the car is investigated. However, SmartKom is not applied to in-car safety as described in Section 5.5.3, Chapter 5. SmartKom deploys rule-based processing and a stochastic model for decision-making. Driver interaction with both online and offline entertainment and information services is considered in Rist (2001) where monitoring of the status of the driving situation, i.e., visibility, distance from another vehicle, road condition and the status of the driver, i.e., steering, pressure on the steering wheel, eye-gaze and heartbeat, is addressed.

7.3. *Future work*

In this section other problem areas and functionality that will be addressed by MediaHub in the future are discussed. The potential deployment of MediaHub within other application domains is also considered.

7.3.1. MediaHub increased functionality

Future work includes the integration of MediaHub with existing multimodal systems, such as TeleTuras (Solon et al. 2007) and CONFUCIUS (Ma 2006), that require complex decision-making and distributed communication. This integration will address the problem of synchronisation, which was not fully addressed in this thesis. It is also possible that structural learning could facilitate generation of entirely new Bayesian networks that model the causal dependencies that exist between variables in a given data set. The data could be derived from existing multimodal corpora, e.g., AMI (Carletta et al. 2006; Petukhova 2005), or it could be created with a Wizard-of-Oz experiment for the application domain. Structural learning, as discussed in Chapter 3, Section 3.11.5, is a feature offered by the Hugin software tool and will be investigated further in the future development of MediaHub. Currently, all semantics in

MediaHub is represented in XML format and manually created for the purpose of demonstration and evaluation. The potential for applying the EMMA (2009) semantic representation formalism is a future consideration, as is the automatic learning of Bayesian networks from corpora of existing data, e.g., AMI (Carletta et al. 2006). Future work will also aim to meet the requirements criteria discussed in Chapter 6, Section 6.5, which are presently only partially met, including the ability to operate across multiple platforms and the ability to learn from both experience and real data. Also planned for future work is a more detailed analysis of MediaHub's performance and scalability.

7.3.2. MediaHub application domains

The use of Bayesian networks in MediaHub across various application domains was discussed in Section 5.7, Chapter 5. A number of other potential application domains have been considered including determining the emotional and intentional state of a user during a web-browsing session, strategy adaptation for an intelligent sales agent and structural learning of a new Bayesian network from a data set. Similar to the 'intelligent interviewer' example discussed in Section 5.5.6, provided there are recognition modules available for speech, facial expression and eye-gaze, it is feasible that a Bayesian network can be applied to determining the emotional and intentional state (e.g., happy, confused, frustrated, angry) of the user whilst browsing the Web. An 'intelligent Web browser' multimodal system could monitor a user's speech, facial expression and eye-gaze to determine the user's emotional state at various stages in a Web browsing session. The relevance of web page content, the accuracy of a search strategy and the understanding of the user's intentions could then be improved based on the beliefs about the user's emotional state. Whilst decision engineers and experts may have varying views on the causal relations in this, and indeed any other, application domain, Bayesian networks would certainly be capable of representing these relations. MediaHub, in conjunction with the Hugin API and Psyclone, has both the framework and functionality necessary to implement Bayesian decision-making for user emotional state recognition.

Another possible application considered is related to the strategy adaptation for an 'intelligent sales agent'. The system could operate in a number of contexts derived through discussions with sales and marketing experts (e.g., *Introduction*, *ExplainProduct*, *Listen*, *NegotiateOnPrice*, *ArrangeAnotherMeeting* and *CloseDeal*). The input nodes of the 'intelligent sales agent' Bayesian network would relate to the gesture, posture, facial expression and body language of the potential buyer. Context and dialogue history would influence the decision-making process. Outputs of the Bayesian networks would be decisions on strategy, e.g. 'attempt to close the sale', 'change package offering', 'drop the price', 'arrange another meeting', and ‘give up’. Another Bayesian network could recommend non-verbal cues and body language categories (e.g., neutral, open, relaxed, confident) and speech output of the ‘intelligent sales agent’. Again, the real challenge is not in the actual representation of the multimodal data or the construction of the Bayesian networks, but in understanding the causal relations present between the relevant variables in the application domain. When this knowledge is elicited, e.g., through discussions with sales and marketing and body language experts, the construction of the Bayesian networks is relatively straightforward.

7.4. *Conclusion*

The aim of this thesis is to develop a Bayesian approach to decision-making within a multimodal distributed platform hub. In order to demonstrate this approach, MediaHub, a test-bed multimodal distributed platform hub, was implemented. MediaHub constitutes a publish-subscribe architecture that uses existing software tools, Psyclone and Hugin, to enable Bayesian decision-making over multimodal input/output data. Evaluation results demonstrate how MediaHub has met the objectives of this research and a set of requirements criteria defined for a multimodal distributed platform hub. This evaluation focused on six key problem areas across five application domains. The evaluation gives positive results that highlight MediaHub’s capabilities for decision-making and shows MediaHub to compare favourably with existing approaches.

Suggestions for future work include increased functionality of MediaHub such as the automatic learning of Bayesian networks from multimodal corpora and the utilisation of EMMA for MediaHub’s semantic representation, as well as the development of a more formalised API or user interface to facilitate integration with existing multimodal systems. In addition, there are opportunities to demonstrate the potential of MediaHub to new application domains. MediaHub is domain independent and could be potentially deployed in a range of multimodal application areas that require distributed processing and intelligent multimodal decision-making and this merits further consideration. The Bayesian approach employed in MediaHub has demonstrated a degree of universality, regarding decision making over multimodal data, which has enhanced its applicability in the domain of multimodal decision-making.

Appendices

Appendix A: MediaHub's Document Type Definitions (DTDs)

Example Document Type Definitions (DTDs) which check the validity of XML semantic segments in MediaHub.

```
<!ELEMENT Offices (Office+)>
<!ELEMENT Office (ID, Person, Coordinates)>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT Person (FirstName, Surname, Gender)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT Surname (#PCDATA)>
<!ELEMENT Gender (#PCDATA)>
<!ELEMENT Coordinates (From, To)>
<!ELEMENT From (X,Y)>
<!ELEMENT To (X,Y)>
<!ELEMENT X (#PCDATA)>
<!ELEMENT Y (#PCDATA)>
```

Figure A.1: DTD for 'anaphora resolution'

```
<!ELEMENT movies (movie+)>
<!ELEMENT movie (title, starttime, moredetails, no, coordinates)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT starttime (#PCDATA)>
<!ELEMENT moredetails (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT coordinates (x,y)>
<!ELEMENT x (from, to)>
<!ELEMENT y (from, to)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
```

Figure A.2: DTD for 'domain knowledge awareness'

```
<!-- speech and gesture can be in any order -->
<!ELEMENT carSafety (face, eyeGaze, posture, head, steering, braking)>
<!ELEMENT face (fNormal, fTired, fTimestamp)>
<!ELEMENT fNormal (#PCDATA)>
<!ELEMENT fTired (#PCDATA)>
<!ELEMENT fTimestamp (#PCDATA)>
<!ELEMENT eyeGaze (eNormal, eTired, eTimestamp)>
<!ELEMENT eNormal (#PCDATA)>
<!ELEMENT eTired (#PCDATA)>
<!ELEMENT eTimestamp (#PCDATA)>
<!ELEMENT posture (pNormal, pTired, pTimestamp)>
<!ELEMENT pNormal (#PCDATA)>
<!ELEMENT pTired (#PCDATA)>
<!ELEMENT pTimestamp (#PCDATA)>
<!ELEMENT head (hNormal, hTired, hTimestamp)>
<!ELEMENT hNormal (#PCDATA)>
<!ELEMENT hTired (#PCDATA)>
<!ELEMENT hTimestamp (#PCDATA)>
<!ELEMENT steering (sNormal, sAbrubt, sTimestamp)>
<!ELEMENT sNormal (#PCDATA)>
<!ELEMENT sAbrubt (#PCDATA)>
<!ELEMENT sTimestamp (#PCDATA)>
<!ELEMENT braking (bNormal, bAbrubt, bTimestamp)>
<!ELEMENT bNormal (#PCDATA)>
<!ELEMENT bAbrubt (#PCDATA)>
<!ELEMENT bTimestamp (#PCDATA)>
```

Figure A.3: DTD for 'multimodal presentation'

```
<!ELEMENT turnTaking2 (speech, eyeGaze, posture, head, status)>
<!ELEMENT speech (sGive, sTake, sTimestamp)>
<!ELEMENT sGive (#PCDATA)>
<!ELEMENT sTake (#PCDATA)>
<!ELEMENT sTimestamp (#PCDATA)>
<!ELEMENT eyeGaze (eGive, eTake, eTimestamp)>
<!ELEMENT eGive (#PCDATA)>
<!ELEMENT eTake (#PCDATA)>
<!ELEMENT eTimestamp (#PCDATA)>
<!ELEMENT posture (pGive, pTake, pTimestamp)>
<!ELEMENT pGive (#PCDATA)>
<!ELEMENT pTake (#PCDATA)>
<!ELEMENT pTimestamp (#PCDATA)>
<!ELEMENT head (hGive, hTake, hTimestamp)>
<!ELEMENT hGive (#PCDATA)>
<!ELEMENT hTake (#PCDATA)>
<!ELEMENT hTimestamp (#PCDATA)>
<!ELEMENT status (oldState, sysTurn)>
<!ELEMENT oldState (userTurn, systemTurn)>
<!ELEMENT userTurn (#PCDATA)>
<!ELEMENT systemTurn (#PCDATA)>
<!ELEMENT sysTurn (sysGive, sysTake, sysTimestamp)>
<!ELEMENT sysGive (#PCDATA)>
<!ELEMENT sysTake (#PCDATA)>
<!ELEMENT sysTimestamp (#PCDATA)>
```

Figure A.4: DTD for ‘turn-taking’

```
<!ELEMENT dialogueAct (speech, intonation, eyebrows, mouth)>
<!ELEMENT speech (sGreeting, sComment, sRequest, sAccept, sReject, sTimestamp)>
<!ELEMENT sGreeting (#PCDATA)>
<!ELEMENT sComment (#PCDATA)>
<!ELEMENT sRequest (#PCDATA)>
<!ELEMENT sAccept (#PCDATA)>
<!ELEMENT sReject (#PCDATA)>
<!ELEMENT sTimestamp (#PCDATA)>
<!ELEMENT intonation (iUnassigned, iRequest, iAccept, iReject, iTimestamp)>
<!ELEMENT iUnassigned (#PCDATA)>
<!ELEMENT iRequest (#PCDATA)>
<!ELEMENT iAccept (#PCDATA)>
<!ELEMENT iReject (#PCDATA)>
<!ELEMENT iTimestamp (#PCDATA)>
<!ELEMENT eyebrows (eUnassigned, eRequest, eAccept, eReject, eTimestamp)>
<!ELEMENT eUnassigned (#PCDATA)>
<!ELEMENT eRequest (#PCDATA)>
<!ELEMENT eAccept (#PCDATA)>
<!ELEMENT eReject (#PCDATA)>
<!ELEMENT eTimestamp (#PCDATA)>
<!ELEMENT mouth (mUnassigned, mRequest, mAccept, mReject, mTimestamp)>
<!ELEMENT mUnassigned (#PCDATA)>
<!ELEMENT mRequest (#PCDATA)>
<!ELEMENT mAccept (#PCDATA)>
<!ELEMENT mReject (#PCDATA)>
<!ELEMENT mTimestamp (#PCDATA)>
```

Figure A.5: DTD for ‘dialogue act recognition’

Appendix B: MediaHub message types

A collection of message types implemented in MediaHub.

building.query.office.occupant.speech.input

building.query.office.occupant.gesture.pointing.input

building.query.office.occupant.intdoc

building.query.office.occupant.repdoc

building.query.office.occupant.hisdoc

building.request.route.speech.from.office.gender

building.request.route.speech.from.office

building.request.route.intdoc

Figure B.1: ‘Anaphora resolution’ message types

movies.speech.input

movies.gesture.pointing.input

movies.eyegaze.input

movies.posture.input

movies.moredetail.input

movies.starttime.input

movies.multimodal.repdoc

Figure B.2: ‘Domain knowledge awareness’ message types

car.eyegaze.input

car.posture.input

car.face.expression.input

car.head.input

car.steering.input

car.braking.input

Figure B.3: ‘Multimodal presentation’ message types

turntaking.eyegaze.input

turntaking.posture.input

turntaking.speech.input

turntaking2.speech.input

turntaking2.eyegaze.input

turntaking2.posture.input

turntaking2.head.input

turntaking2.status.input

Figure B.4: ‘Turn-taking’ message types

dialogueact.speech.input

dialogueact.intonation.input

dialogueact.eyebrows.input

dialogueact.mouth.input

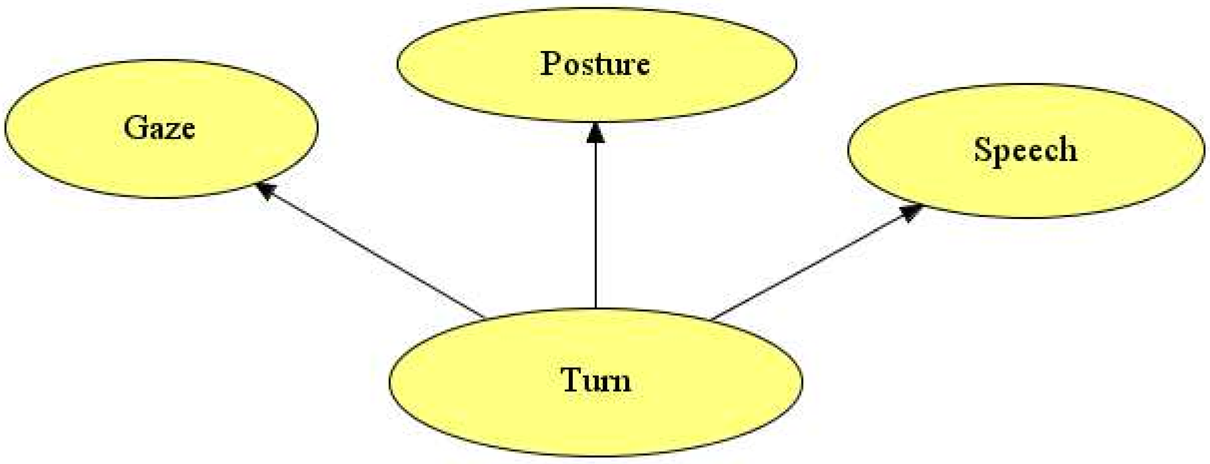
Figure B.5: ‘Dialogue act recognition’ message types

Appendix C: HTML Bayesian network documentation

Example of HTML documentation automatically generated by the Hugin GUI for the 'domain knowledge awareness' Bayesian network.

Domain Knowledge Awareness

The model is depicted below:



Nodes

Gaze

Represent the belief that the user wants to give a turn based on gaze input.

Name = Gaze
Label = Gaze
Type = Discrete Labelled Node

States

Give : Belief, based on gaze input, that the user wishes to give the turn to the agent.
Take : Belief, based on gaze input, that the user wishes to take the turn from the agent.

Parents

- Turn

Posture

Represent the belief that the user wants to give a turn based on posture input.

Name = Posture

Label	= Posture
Type	= Discrete Labelled Node

States

Give	: Belief, based on posture input, that the user wishes to give the turn to the agent.
Take	: Belief, based on posture input, that the user wishes to take the turn from the agent.

Parents

- Turn

Speech

Represent the belief that the user wants to give a turn based on speech input.

Name	= Speech
Label	= Speech
Type	= Discrete Labelled Node

States

Give	: Belief, based on speech input, that the user wishes to give the turn to the agent.
Take	: Belief, based on speech input, that the user wishes to take the turn from the agent.

Parents

- Turn

Turn

The Turn node relates to the turn-taking strategy of the agent.

Name	= Turn
Label	= Turn
Type	= Discrete Labelled Node

States

Give	: Give the turn to the User
Take	: Take the turn from the User

Appendix D: Test case tables

Test case tables used to confirm correctness of Bayesian networks.

Speech				MoreDetail				StartTime				EyeGaze				ChosenMovie				OK?
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	
65	0	15	20	T	F	F	F	F	F	F	F	55	45	0	0	95	3	1	1	Yes
66	0	0	34	T	F	F	F	T	F	F	T	80	20	0	0	83	0	0	17	No – % of 1 should be > 83
74	0	26	0	T	F	F	F	T	F	F	F	76	24	0	0	89	11	0	0	No – % of 1 should be > 89
0	0	35	65	F	T	F	F	F	F	T	T	0	6	40	54	1	7	32	60	Yes
0	0	40	60	F	T	F	F	F	F	T	T	0	0	50	50	1	5	41	53	Yes
0	0	38	62	F	T	F	F	F	F	F	T	0	0	48	52	1	5	8	86	Yes
0	0	67	33	F	T	F	F	F	F	F	T	0	0	46	54	1	6	17	76	Yes
54	0	0	46	T	F	F	F	T	F	F	F	89	11	0	0	100	0	0	0	Yes
59	0	0	41	T	F	F	F	T	F	F	F	50	35	15	0	99	.5	0	.5	Yes
59	0	0	41	T	F	F	F	T	F	F	T	50	35	15	0	97	1	0	2	Yes
0	88	12	0	F	T	F	F	F	T	F	F	0	90	10	0	.5	98	1	.5	Yes
0	88	12	0	F	T	F	F	F	F	T	F	0	90	10	0	.5	93	6	.5	Yes
0	0	75	25	F	F	F	F	F	F	T	F	0	0	80	20	.5	.5	97	2	Yes
0	0	75	25	F	F	T	F	F	F	F	F	0	0	80	20	.5	.5	97	2	Yes
0	0	75	25	F	F	T	F	F	F	T	F	0	0	80	20	0	0	100	0	Yes
80	0	20	0	T	F	T	F	T	F	F	F	55	45	0	0	98	1	1	0	Yes
50	0	50	0	T	T	T	T	T	F	F	F	78	22	0	0	96	2	2	0	Yes
50	0	50	0	T	F	F	F	T	F	F	F	78	22	0	0	99	.5	.5	0	Yes

Table D.1: Test cases for ‘domain knowledge awareness’ Bayesian network

Steering		Braking		Face		EyeGaze		Head		Posture		Tired		SpeechOutput			OK?
N	A	N	A	N	T	N	T	N	T	N	T	N	T	N	F	W	
65	35	45	55	50	50	55	45	60	40	40	60	58	42	36	30	34	Yes
70	30	22	78	70	30	85	15	45	55	50	50	77	23	27	39	34	Yes
30	70	25	75	51	49	90	10	80	20	60	40	67	33	15	29	56	Yes
20	80	20	80	70	30	50	50	65	35	78	22	54	46	8	23	69	Yes
29	71	21	79	95	5	50	50	50	50	50	50	73	27	14	30	56	Yes
55	45	50	50	45	55	43	57	34	66	32	68	35	65	25	30	45	Yes
88	12	85	15	51	49	55	45	19	81	78	22	85	15	90	5	5	No, value of None state of SpeechOutput should be lower
10	90	35	65	50	50	45	55	34	66	55	45	23	77	3	20	77	Yes – but try few more test cases like this!
5	95	30	70	55	45	45	55	40	60	60	40	25	75	2	18	80	Yes
20	80	11	89	60	40	60	40	25	75	65	35	18	82	3	15	82	Yes
14	86	38	62	54	46	49	51	20	80	52	48	24	76	5	22	73	Yes
89	11	94	6	5	95	20	80	50	50	10	90	20	80	72	18	10	Yes
89	11	94	6	50	50	50	50	50	50	50	50	80	20	87	9	4	Yes
50	50	50	50	50	50	50	50	50	50	50	50	50	50	29	29	42	Yes
66	34	49	51	60	40	40	60	55	45	45	55	60	40	40	29	31	Yes
66	34	61	39	60	40	40	60	60	40	45	55	66	34	49	26	25	Yes
72	28	85	15	55	45	45	55	45	55	50	50	71	29	67	20	13	Yes
20	80	15	85	50	50	41	59	38	62	25	75	16	84	2	14	84	Yes
98	2	30	70	52	48	48	52	50	50	40	60	61	39	36	44	20	Yes
30	70	98	2	52	48	48	52	50	50	40	60	61	39	36	44	20	Yes
50	50	50	50	55	45	56	44	40	60	43	57	52	48	30	29	41	Yes

Table D.2: Test cases for ‘multimodal presentation’ Bayesian network

Gaze		Posture		Speech		Turn		OK?
G	T	G	T	G	T	G	T	
65	35	45	55	60	40	67	33	Yes
35	65	30	70	50	50	30	70	Yes
75	25	80	20	11	89	59	41	Yes
80	20	89	11	30	70	78	22	Yes
95	5	90	10	10	90	77	23	Yes
50	50	50	50	50	50	50	50	Yes
50	50	80	20	80	20	82	18	Yes
20	80	80	20	80	20	68	32	Yes
10	90	72	28	52	48	39	61	Yes
25	75	78	22	58	42	57	43	Yes
34	66	48	52	52	48	40	60	Yes
10	90	90	10	50	50	50	50	Yes
30	70	70	30	50	50	50	50	Yes
45	55	55	45	50	50	50	50	Yes
5	95	55	45	50	50	25	75	Yes
12	88	44	56	22	78	14	86	Yes
12	88	44	56	5	95	9	91	Yes
95	5	90	10	92	8	97	3	Yes
40	60	52	48	51	49	46	54	Yes
88	12	50	50	12	88	50	50	Yes
50	50	78	22	18	82	47	53	Yes
50	50	49	51	25	75	34	66	Yes
62	38	40	60	60	40	57	43	Yes
55	45	33	67	56	44	46	54	Yes
35	65	52	48	41	59	37	63	Yes

Table D.3: Test cases for ‘turn-taking’ Bayesian network

Speech		Gaze		Posture		Head		U_Turn		S_Turn		Old_State		New_State		OK?
G	T	G	T	G	T	G	T	GT	TT	GT	TT	U	S	U	S	
74	26	60	40	40	60	40	60	60	40	F	T	T	F	24	76	Yes
84	16	20	80	80	20	21	79	54	46	F	T	T	F	28	72	Yes
84	16	50	50	80	20	21	79	70	30	F	T	T	F	18	82	Yes
60	40	54	46	38	62	50	50	52	48	F	T	T	F	29	71	Yes
45	55	20	80	38	62	30	70	17	83	T	F	F	T	91	9	Yes
55	45	20	80	38	62	30	70	21	79	T	F	F	T	89	11	Yes
50	50	50	50	50	50	50	50	50	50	T	F	F	T	75	25	Yes
45	55	50	50	38	62	30	70	29	71	T	F	F	T	85	15	Yes
74	26	55	45	62	38	49	51	72	28	F	T	T	F	17	83	Yes
91	9	70	30	85	15	70	30	94	6	F	T	T	F	3	97	Yes
95	5	95	5	20	80	20	80	71	29	F	T	T	F	17	83	Yes
50	50	10	90	50	50	50	50	27	73	T	F	F	T	86	14	Yes
50	50	85	15	50	50	50	50	68	32	T	F	F	T	66	34	Yes
50	50	90	10	50	50	90	10	87	13	T	F	T	F	56	44	Yes
35	65	50	50	30	70	20	80	17	83	T	F	F	T	91	9	Yes
35	65	50	50	49	51	49	51	40	60	T	F	F	T	80	20	Yes
49	51	50	50	49	51	49	51	48	52	T	F	F	T	76	24	Yes
85	15	84	16	90	10	96	4	98	2	F	T	T	F	1	99	Yes
10	90	15	85	11	89	30	70	3	97	T	F	F	T	98	2	Yes
10	90	48	52	11	89	30	70	7	93	T	F	F	T	96	4	Yes
45	55	48	52	43	57	50	50	42	58	T	F	F	T	79	21	Yes
55	45	48	52	11	89	50	50	30	70	T	F	F	T	85	15	Yes
90	10	90	10	40	60	40	60	82	18	F	T	T	F	11	89	Yes

Table D.4: Test cases for alternative ‘turn-taking’ Bayesian network

Speech					Intonation				Eyebrows				Mouth				DialogueAct					OK?
G	C	R	A	X	U	R	A	X	U	R	A	X	U	R	A	X	G	C	R	A	X	
12	9	11	68	0	25	25	25	25	36	24	40	0	30	30	35	5	14	12	9	65	0	Yes
34	63	3	0	0	85	0	10	5	50	22	20	8	25	25	25	25	37	63	0	0	0	Yes
66	34	0	0	0	85	0	10	5	50	22	20	8	25	25	25	25	64	36	0	0	0	Yes
85	15	0	0	0	85	0	10	5	50	22	20	8	25	25	25	25	81	19	0	0	0	Yes
95	5	0	0	0	85	0	10	5	50	22	20	8	25	25	25	25	89	11	0	0	0	Yes
95	5	0	0	0	90	0	0	10	90	0	0	10	25	25	25	25	90	10	0	0	0	Yes
0	20	0	80	0	0	0	30	70	0	0	30	70	0	0	50	50	0	0	0	75	25	Yes
0	20	0	80	0	25	25	25	25	0	0	30	70	0	0	50	50	0	0	0	85	15	Yes
0	0	0	80	20	0	0	30	70	25	25	25	25	0	0	50	50	0	0	0	60	40	Yes
0	0	0	80	20	0	0	30	70	25	25	25	25	25	25	25	25	0	0	1	60	39	Yes
0	0	0	80	20	0	0	90	10	25	25	25	25	25	25	25	25	0	0	0	95	5	Yes
0	0	0	90	10	0	0	90	10	25	25	25	25	0	0	90	10	0	0	0	99	1	Yes
0	0	0	90	10	0	0	90	10	25	25	25	25	0	0	10	90	0	0	0	85	15	Yes
0	0	0	90	10	0	0	25	75	25	25	25	25	0	0	10	90	0	0	0	93	7	Yes
0	0	0	55	45	0	0	25	75	25	25	25	25	0	0	10	90	0	0	0	7	93	Yes
20	20	20	20	20	0	0	25	75	25	25	25	25	0	0	10	90	0	0	0	6	94	Yes
20	20	20	20	20	0	0	25	75	25	25	25	25	0	0	75	25	0	0	0	50	50	Yes
20	20	20	20	20	0	0	25	75	0	0	75	25	0	0	75	25	0	0	0	72	28	Yes
0	0	0	60	40	0	0	60	40	0	0	45	55	0	0	46	54	0	0	0	60	40	Yes
0	0	0	60	40	0	0	80	20	0	0	30	70	0	0	46	54	0	0	0	66	34	Yes
0	0	0	60	40	0	0	80	20	0	0	30	70	0	0	30	70	0	0	0	51	49	Yes
0	25	75	0	0	25	75	0	0	0	80	20	0	0	90	10	0	0	0	100	0	0	Yes
0	25	75	0	0	25	75	0	0	0	30	70	0	0	30	70	0	0	0	97	3	0	Yes

Table D.5: Test cases for ‘dialogue act recognition’ Bayesian network

References

Adams, D. (1979) The Hitchhiker's Guide to the Galaxy. London, England: Barker.

Agena (2009)
http://www.agenarisk.com/ Site visited 16/03/09.

Amtrup, J.W. (1995) ICE-INTARC Communication Environment Users Guide and Reference Manual Version 1.4, University of Hamburg, October.

Allwood, J., L. Cerrato, K. Jokinen, C. Navarretta & P. Paggio (2007) The MUMIN coding scheme for the annotation of feedback in multimodal corpora: a prerequisite for behavior simulation. In Language Resources and Evaluation. Special Issue. J.-C. Martin, P. Paggio, P. Kuehnlein, R. Stiefelhagen, F. Pianesi (eds.) Multimodal Corpora for Modeling Human Multimodal Behavior, Vol. 41, No. 3-4, 273-287.

André, E., T. Rist (1994) Referring to world objects with text and pictures. In Proceedings of the 15th International Conference on Computational Linguistics, Kyoto, Japan, 530-534.

André, E., J. Muller & T. Rist (1996) The PPP Persona: A Multipurpose Animated Presentation Agent. In Proceedings of Advanced Visual Interfaces, Gubbio, Italy, 245–247.

Babuska, R. (1993) Fuzzy toolbox for MATLAB. In Proceedings of the 2nd IMACS International Symposium on Mathematical and Intelligent Models in System Simulation, University Libre de Bruxelles, Brussels, Belgium.

Bayer, S., C. Doran & B. George (2001) Dialogue Interaction with the DARPA Communicator Infrastructure: The development of Useful Software. In Proceedings of HLP 2001, First International Conference on Human Language Technology Research, San Diego, CA, USA, 114-116.

Berners-Lee, T., J. Hendler & O. Lassila (2001) The Semantic Web, In Scientific American, May 17, p. 35-43.

Berton, A., D. Buhler, W. Minker (2006) SmartKom - Mobile Car: User Interaction with Mobile Services in a Car Environment. In SmartKom: Foundations of Multimodal Dialogue Systems, W. Wahlster (Ed.), Berlin, Germany: Springer-Verlag, 523-537.

Bolt, R.A. (1980) "Put-that-there" Voice and gesture at the graphics interface. Computer Graphics (SIGGRAPH '80 Proceedings), 14(3), July, 262–270.

Bolt, R.A. (1987) Conversing with Computers. In Readings in Human-Computer Interaction: A Multidisciplinary Approach, R. Baecker & W. Buxton (Eds.), California, U.S.A.: Morgan Kaufmann.

Brock, D.C. (2006) (Ed.) Understanding Moore's Law: Four Decades of Innovation. Philadelphia, USA: Chemical Heritage Press.

Brøndsted, T., P. Dalsgaard, L.B. Larsen, M. Manthey, P. Mc Kevitt, T.B. Moeslund & K.G. Olesen (1998) A platform for developing Intelligent MultiMedia applications. Technical Report

R-98-1004, Center for PersonKommunikation (CPK), Institute for Electronic Systems (IES), Aalborg University, Denmark, May.

Brøndsted, T. (1999) Reference problems in Chameleon, In IDS-99, 133-136.

Brøndsted, T., P. Dalsgaard, L.B. Larsen, M. Manthey, P. Mc Kevitt, T.B. Moeslund & K.G. Olesen (2001) The IntelliMedia WorkBench - An Environment for Building Multimodal Systems. In Advances in Cooperative Multimodal Communication: Second International Conference, CMC'98, Tilburg, The Netherlands, January 1998, Selected Papers, Harry Bunt & Robbert-Jan Beun (Eds.), 217-233. Lecture Notes in Artificial Intelligence (LNAI) series, LNAI 2155, Berlin, Germany: Springer Verlag.

BUGS (2009)
http://www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml Site visited 16/03/09.

Bunt, H.C. & S. Keizer (2006) Multidimensional Dialogue Management. In Proceedings of SIGdial Workshop on Discourse and Dialogue, 37-45.

Bunt, H.C., M. Kipp, M. Maybury & W. Wahlster (2005) Fusion and Coordination for Multimodal Interactive Information Presentation. In Multimodal Intelligent Information Presentation (Text, Speech and Language Technology), O. Stock & M. Zancanaro (Eds.), Vol. 27, Dordrecht, The Netherlands: Springer, 325-340.

Carletta, J. (2006) Announcing the AMI Meeting Corpus. In The ELRA Newsletter 11(1), January-March, 3-5.

Carlson, R. (1996) The Dialog Component in the Waxholm System. In Proceedings of Twente Workshop on Language Technology (TWLT11) Dialogue Management in Natural Language Systems, University of Twente, The Netherlands, 209-218.

Carlson, R. & B. Granström (1996) The Waxholm spoken dialogue system. In Palková Z, (Ed.), 39-52, Phonetica Pragensia IX. Charisteria viro doctissimo Premysl Janota oblata. Acta Universitatis Carolinae Philologica 1.

Carpenter, B. (1992) The Logic of Typed Feature Structures. Cambridge, England: Cambridge University Press.

Cassell, J., J. Sullivan, S. Prevost, & E. Churchill (Eds.) (2000) Embodied Conversational Agents. Cambridge, MA: MIT Press.

Cassell, J., H. Vilhjalmsson and T. Bickmore (2001) BEAT: the Behavior Expression Animation Toolkit, Computer Graphics Annual Conference, SIGGRAPH 2001 Conference Proceedings, Los Angeles, Aug 12-17, 477-486.

Chester, M. (2001) Cross-Platform Integration with XML and SOAP. In IT Pro, September/October, 26-34.

Cheyer, A., L. Julia & J.C. Martin (1998) A Unified Framework for Constructing Multimodal Experiments and Applications, In Proceedings of CMC ’98: Tilburg, The Netherlands, 63-69.

Choy, K.W., A.A. Hopgood, L. Nolle & B.C. O'Neill (2004a) Implementing a blackboard system in a distributed processing network. In Expert Update, Vol. 7, No. 1, Spring, 16-24.

Choy, K.W., A.A. Hopgood, L. Nolle & B.C. O'Neill (2004b) Implementation of a tileworld testbed on a distributed blackboard system. In Proceedings of the 18th European Simulation Multiconference (ESM2004), Magdeburg, Germany, June 2004, Horton, G., (Ed.), 129-135.

CMU (2009)
JavaBayes http://www.cs.cmu.edu/~javabayes/Home/ Site visited 16/03/09.

Cohen-Rose, A.L. & S. B. Christiansen (2002) The Hitchhiker's Guide to the Galaxy. In Language, Vision and Music, Mc Kevitt, Paul, Seán Ó Nualláin and Conn Mulvihill (Eds.), 55-66.

CORBA (2009)
http://java.sun.com/developer/onlineTraining/corba/corba.html Site visited 16/03/09.

DAML (2009)
http://www.daml.org/ Site visited 16/03/09.

DAML-S (2009)
http://www.daml.org/services/owl-s/ Site visited 16/03/09.

DAML+OIL (2009)
http://www.daml.org/2001/03/daml+oil-index Site visited 16/03/09.

Davis, L. (Ed.) (1991) Handbook of Genetic Algorithms. New York, USA: Van Nostrand Reinhold.

de Rosis, F., c, I. Poggi, V. Carofiglio & B. De Carolis (2003) From Greta's mind to her face: modelling the dynamics of affective states in a conversational embodied agent, International Journal of Human-Computer Studies, Vol. 59 No. 1-2, 81-118.

EMBASSI (2009)
http://www.embassi.de/ewas/ewas_frame.html Site visited 16/03/09.

EMMA (2009)
http://www.w3.org/TR/2004/WD-emma-20041214/ Site visited 16/03/09.

Fensel, D., F. van Harmelen, I. Horrocks, D. McGuinness & P. Patel-Schneider (2001) OIL: An Ontology Infrastructure for the Semantic Web. In IEEE Intelligent Systems, 16(2), 38-45.

Finin, T., R. Fritzson, D. McKay & R. McEntire (1994) KQML as an Agent Communication Language. In Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM '94), Gaithersburg, MD, USA, 456-463.

Fink, G.A., N. Jungclaus, F. Kummert, H. Ritter & G. Sagerer (1995) A Communication Framework for Heterogeneous Distributed Pattern Analysis. In International Conference on Algorithms And Architectures for Parallel Processing, Brisbane, Australia, 881-890.

Fink, G.A., N. Jungclaus, F. Kummert, H. Ritter & G. Sagerer (1996) A Distributed System for Integrated Speech and Image Understanding. In International Symposium on Artificial Intelligence, Cancun, Mexico, 117-126.

Foster, M.E. (2004) Corpus-based Planning of Deictic Gestures in COMIC. Student session, Third International Conference on Natural Language Generation (INLG 2004), Brockenhurst, England, July, 198-204.

Freeman (2009)
Make Room For JavaSpaces Part 1
http://www.javaworld.com/javaworld/jw-11-1999/jw-11-jiniology.html Site visited 16/03/09.

Genie (2009)
http://genie.sis.pitt.edu/ Site visited 16/03/09.

Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley.

Gratch, J., N. Wang, J. Gerten, E. Fast & R. (2007) Duffy Creating Rapport with Virtual Agents. In Proceedings of the International Conference on Intelligent Virtual Agents, Paris, France, 125-138.

Grosz, B.J. & C.L. Sidner (1986) Attention, intentions and the structure of discourse. Computational Linguistics, Vol. 12, 175-204.

Grosz, B.J., C.L. Sidner (1990) Plans for discourse. In P.R. Cohen, J.L. Morgan & M.E. Pollack (eds.), 417-444, Intentions and Communication, Cambridge, MA:MIT Press.

Gruber, T.R. (1993) A translation approach to portable ontology specifications. In Knowledge Specification, Vol. 5, 199-220.

Haddawy, P. (1999) Introduction to this Special Issue: An overview of some recent developments in Bayesian problem-solving techniques, AI Magazine, Special Issue on Uncertainty in AI, Vol. 20, No. 2, 11-19.

Hall, P. & P. Mc Kevitt (1995) Integrating vision processing and natural language processing with a clinical application. In Proceedings of the Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, New Zealand, November, 373 – 376.

Haykin, S. (1999) Neural Networks, A Comprehensive Foundation. Prentice Hall, Upper Saddle River, NJ.

Heckerman, D., E. Horvitz, B. Nathwani (1992) Towards normative expert systems: Part I. The Pathfinder project, Methods of Information in Medicine, 31(2), 90-105.

Herzog, G., H. Kirchmann, S. Merten, A. Ndiaye & P. Poller (2003) MULTIPLATFORM Testbed: An Integration Platform for Multimodal Dialog Systems. In H. Cunningham & J. Patrick (Eds.), 75-82, Proceedings of the HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS), Edmonton, Canada.

Holland, J.H. (1992) Genetic Algorithms. Scientific American. Vol. 260, July, 44-51.

Holzapfel, H., C. Fuegen, M. Denecke & A. Waibel (2002) Integrating Emotional Cues into a Framework for Dialogue Management. In Proceedings of the International Conference on Multimodal Interfaces, 141-148.

Hopgood, A.A. (2003) Artificial Intelligence: Hype or Reality? In IEEE Computer Society Press, Vol. 36, No. 5, IEEE Computer Society, May, 24-28.

Horvizt, E. & M. Barry (1995) Display of Information for Time-Critical Decision Making. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, 296-305.

Horvitz, E., J. Breese, D. Heckerman, D. Hovel & K. Rommelse (1998) The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, July, 256-265.

Hugin (2009)
Hugin Expert Developers Site http://developer.hugin.com/ Site visited 16/03/09.

Jensen, F.V., (1996) An introduction to Bayesian networks. London, England: UCL Press.

Jensen, F.V. (2000) Bayesian Graphical Models, Encyclopaedia of Environmetrics, Wiley, Sussex, UK.

Jensen, F.V. & T.D. Nielsen (2007) Bayesian Networks and Decision Graphs, Second Edition, New York, USA: Springer Verlag.

Jeon, H., C. Petrie & M.R. Cutkosky (2000) JATLite: A Java Agent Infrastructure with Message Routing. IEEE Internet Computing Vol. 4, No. 2, Mar/Apr, 87-96.

Johnston, M., P.R. Cohen, D. McGee, S. L. Oviatt, J.A. Pittman & I. Smith (1997) Unification-based multimodal integration. In Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics, Madrid, Spain, 281-288.

Johnston, M. (1998) Unification-based multimodal parsing. In Proceedings of the 36th conference on Association for Computational Linguistics, Montreal, Quebec, Canada, 624-630.

Johnston, M., S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker & P. Maloor (2002) MATCH: An Architecture for Multimodal Dialog Systems. In Proceedings of ACL-02, 376–383.

Jokinen, K., A. Kerminen, M. Kaipainen, T. Jauhiainen, G. Wilcock, M. Turunen, J. Hakulinen, J. Kuusisto & K. Lagus (2002) Adaptive Dialogue Systems – Interactions with Interact. In Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue of ACL-02, Philadelphia, PA, July 11-12, 64-73.

Kadie, C.M., D. Hovel & E. Horvitz (2001) MSBNx: A Component-Centric Toolkit for Modeling and Inference with Bayesian Networks. Microsoft Research Technical Report MSR-TR-2001-67, July 2001.

Kelleher, J., T. Doris, Q. Hussain & S. Ó Nualláin (2000) SONAS: Multimodal, Multi-User Interaction with a Modelled Environment. In S. Ó Nualláin, (Ed.), 171-184, Spatial Cognition. Amsterdam, The Netherlands: John Benjamins Publishing Co.

Kipp, M. (2001) Anvil - a generic annotation tool for multimodal dialogue. In Proceedings of Eurospeech 2001, Aalborg, 1367-1370.

Kipp, M. (2006) Creativity meets Automation: Combining Nonverbal Action Authoring with Rules and Machine Learning. In Proceedings of the 6th International Conference on Intelligent Virtual Agents, 230-242, Springer.

Kirste T., T. Herfet & M. Schnaider (2001) EMBASSI: Multimodal Assistance for Infotainment and Service Infrastructures. In Proceedings of the 2001 EC/NSF Workshop Universal on Accessibility of Ubiquitous Computing: Providing for the Elderly, Alcácer do Sal, Portugal, 41-50.

Kjærulff, U.B. & A.L. Madsen (2006) Probabilistic Networks for Practitioners – A Guide to Construction and Analysis of Bayesian Networks and Influence Diagrams, Department of Computer Science, Aalborg University, HUGIN Expert A/S.

Klein, M. (2001) XML, RDF, and relatives. In Intelligent Systems, IEEE, Vol. 16, No. 2, March-April, 26-28.

Klein, M. (2002) Interpreting XML documents via an RDF schema ontology. In Proceeding of the 13th International Workshop on Database and Expert Systems Applications, September, Amsterdam, Netherlands, 889 – 893.

Kopp, S. & I. Wachsmuth (2004) Synthesizing multimodal utterances for conversational agents. In Computer Animation and Virtual Worlds, 2004; Vol. 15, 39–52.

Kristensen, T. (2001) T Software Agents In A Collaborative Learning Environment. In International Conference on Engineering Education, Oslo, Norway, Session 8B1, August, 20-25.

López-Cózar Delgado, R. & M. Araki (2005) Spoken, Multilingual and Multimodal Dialogue Systems: Development and Assessment. Chichester, England: John Wiley & Sons.

Lumiere (2009)
http://research.microsoft.com/~horvitz/lum.htm Site visited 16/03/09.

Ma, M. & P. Mc Kevitt (2003) Semantic representation of events in 3D animation. In Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5), Harry Bunt, Ielka van der Sluis and Roser Morante (Eds.), 253-281. Tilburg University, Tilburg, The Netherlands, January.

Martin, J.C., S. Grimard & K. Alexandri (2001) On the annotation of the multimodal behavior and computation of cooperation between modalities. In Proceedings of the workshop on Representing, Annotating, and Evaluating Non-Verbal and Verbal Communicative Acts to Achieve Contextual Embodied Agents, May 29, Montreal, Fifth International Conference on Autonomous Agents, 1-7.

Martin, J.C. & M. Kipp (2002) Annotating and Measuring Multimodal Behaviour - Tycoon Metrics in the Anvil Tool. In Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC'2002), Las Palmas, Canary Islands, Spain, May, 29-31.

Martinho, C., A. Paiva, & M. R. Gomes (2000). Emotions for a Motion: Rapid Development of Believable Pathematic Agents in Intelligent Virtual Environments. Applied Artificial Intelligence, Vol. 14, No. 1, 33-68.

Maybury, M.T. (Ed.) (1993) Intelligent Multimedia Interfaces. Menlo Park: AAAI/MIT Press.

Mc Guinness, D.L., R. Fikes, J. Hendler & L.A. Stein (2002) DAML+OIL: An Ontology Language for the Semantic Web. In IEEE Intelligent Systems, Vol. 17, No. 5, September/October, 72-80.

Mc Kevitt, P. (Ed.) (1995/96) Integration of Natural Language and Vision Processing (Volumes I-IV): Computational Models and Systems. London, U.K.: Kluwer Academic Publishers.

Mc Kevitt, P., S. Ó Nualláin & C. Mulvihill (Eds.) (2002), Language, vision and music, Readings in Cognitive Science and Consciousness, Advances in Consciousness Research, AiCR, Vol. 35. Amsterdam, The Netherlands/Philadelphia, USA: John Benjamins Publishing Company.

Mc Kevitt, Paul (2005) Advances in Intelligent MultiMedia: MultiModal semantic representation. In Proceedings of the Pacific Rim International Conference on Computational Linguistics (PACLING-05), Hiroshi Sakaki (Ed.), Meisei University (Hino Campus), Hino-shi, Tokyo, Japan, August, 2-13.

Mc Tear, M.F. (2004) Spoken dialogue technology: toward the conversational user interface. London, England: Springer Verlag.

MIAMM (2009)
http://miamm.loria.fr/ Site visited 16/03/09.

Microsoft (2009)
http://www.microsoft.com/surface/index.html Site visited 16/03/09.

Mindmakers (2009)
http://www.mindmakers.org/ Site visited 16/03/09.

Minsky, M. (1975) A Framework for representing knowledge. In Readings in Knowledge Representation, R. Brachman and H. Levesque (Eds.), Los Altos, CA: Morgan Kaufmann, 245-262.

MPEG-7 (2009)
http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm Site visited 16/03/09.

MSBNx (2009)
http://www.research.microsoft.com/adapt/MSBNx/ Site visited 16/03/09.

MS .NET (2009)
http://www.microsoft.com/NET/ Site visited 16/03/09.

Murphy (2009)
Website of Kevin Patrick Murphy. http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html Site visited 16/03/09.

Neal, J. & S. Shapiro (1991) Intelligent Multi-Media Interface Technology. In Intelligent User Interfaces, J. Sullivan and S. Tyler (Eds.), 11-43, Reading, MA: Addison-Wesley.

Neal, R.M. (1993) Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report, CRG-TR-93-1, University of Toronto, Canada.

Nejdl, W., M. Wolpers & C. Capella (2000) The RDF Schema Specification Revisited. In Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik, Modellierung 2000, April.

Ng-Thow-Hing, V., J. Lim, J. Wormer, R.K. Sarvadevabhatla, C. Rocha, K. Fujimura & Y. Sakagami (2008) The memory game: Creating a human-robot interactive scenario for ASIMO. IROS 2008, 779-786.

Nigay, L. & J. Coutaz (1995) A generic platform for addressing the multimodal challenge. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 98-105.

Nolle, L., K. Wong & A.A. Hopgood (2001) DARBS: a distributed blackboard system. In Proceedings of ES2001, Research and Development in Intelligent Systems XVIII, M. Bramer, F. Coenen and A. Preece (Eds.), 161-170, Berlin, Germany: Springer-Verlag.

Norsys (2009)
http://www.norsys.com/ Site visited 16/03/09.

OAA (2009)
http://www.ai.sri.com/~oaa/whitepaper.html Site visited 16/03/09.

Okada, N. (1996) Integrating vision, motion, and language through mind. In Integration of Natural Language and Vision Processing (Volume IV): Recent Advances. McKevitt, P. (Ed.) 55-79. Dordrecht, The Netherlands: Kluwer-Academic Publishers.

Okada, N., K. Inui & M. Tokuhisa (1999) Towards affective integration of vision, behavior, and speech processing. In Integration of Speech and Image Understanding, September, 49-77.

Ó Nualláin, S. & A. Smith (1994) An Investigation into the Common Semantics of Language and Vision. In P. McKevitt, (Ed.), 21-30, Integration of Natural Language and Vision Processing (Volume I): Computational Models and Systems. London, U.K.: Kluwer Academic Publishers.

Ó Nualláin, S., B. Farley & A. Smith (1994) The Spoken Image System: On the visual interpretation of verbal scene descriptions. In P. McKevitt, (Ed.), 36-39, Proceedings of the Workshop on integration of natural language and vision processing, Twelfth American National Conference on Artificial Intelligence (AAAI-94). Seattle, Washington, USA, August.

OWL (2009)
http://www.w3.org/2004/OWL/ Site visited 16/03/09.

Oxygen (2009)
http://oxygen.lcs.mit.edu/Overview.html Site visited 16/03/09.

Passino, K.M. & S. Yurkovich (1997) Fuzzy Control. Menlo Park, CA: Addison Wesley Longman.

Pastra, K. & Y. Wilks (2004) Image-language Multimodal Corpora: needs, lacunae and an AI synergy for annotation. In Proceedings of the 4th Language Resources and Evaluation Conference (LREC), Lisbon, Portugal, 767-770.

Pearl, J. (2000) Causality: Models, Reasoning and Inference, New York, USA: Cambridge University Press.

Pearl, J. (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, 2nd edition, San Francisco, USA: Morgan Kaufmann.

Petukhova, V.V. (2005) Multidimensional interaction of dialogue acts in the AMI project. MA thesis, Tilburg University, Tilburg, The Netherlands, August.

Pineda, L. & G. Garza (1997) A model for multimodal reference resolution. Computational Linguistics. Vol. 26, No. 2, 139-193.

Pourret, O., P. Naïm & B. Marcot (Eds.) (2008) Bayesian Networks: A Practical Guide to Applications. Chichester, England: John Wiley & Sons.

Psyclone (2009)
http://www.cmlabs.com/psyclone/ Site visited 16/03/09.

RDF Schema (2009)
http://www.w3.org/TR/rdf-schema/ Site visited 16/03/09.

Reithinger, N., C. Lauer & L. Romary (2002) MIAMM - Multidimensional Information Access using Multiple Modalities. In International CLASS Workshop on Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems, Copenhagen, Denmark, 28-29 June.

Reithinger, N. & D. Sonntag (2005) An integration framework for a mobile multimodal dialogue system accessing the semantic web. In Interspeech 2005, Lisbon, Portugal, 841-844.

Rehm, M. & E. André (2006) From Annotated Multimodal Corpora to Simulated Human-Like Behaviors. ZiF Workshop, 1-17.

Rich, C. & C. Sidner (1997) COLLAGEN: When Agents Collaborate with People. In First International Conference on Autonomous Agents, Marina del Rey, CA, February, 284-291.

Rickel, J., J. Gratch, R. Hill, S. Marsella, & W. Swartout (2001) Steve Goes to Bosnia: Towards a New Generation of Virtual Humans for Interactive Experiences. In AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, Stanford University, CA, March.

Rist, T. (2001) Media and Content Management in an Intelligent Driver Support System. International Seminar on Coordination and Fusion in MultiModal Interaction, Schloss Dagstuhl International Conference and Research Center for Computer Science, Wadern, Saarland, Germany, 29 Oct - 2 Nov. (www.dfki.de/~wahlster/Dagstuhl_Multi_Modality/rist-dagstuhl.pdf Site visited 16/03/09)

Rutledge, L. (2001) SMIL 2.0: XML For Web Multimedia. In IEEE Internet Computing, Sept-Oct, 78-84.

Rutledge, L. & P. Schmitz (2001) Improving Media Fragment Integration in Emerging Web Formats. In Proceedings of the International Conference on Multimedia Modelling (MMM01), CWI, Amsterdam, The Netherlands, November 5-7, 147-166.

Sidner, C.L. (1994) An Artificial Discourse Language for Collaborative Negotiation. In Proceedings of the Twelfth National Conference on Artificial Intelligence, Vol. 1, MIT Press, Cambridge, MA, 814-819.

SmartKom (2009)
http://www.smartkom.org Site visited 16/03/09.

SMIL (2009a)
http://www.w3.org/TR/REC-smil/ Site visited 16/03/09.

SMIL (2009b)
http://www.w3.org/AudioVideo/ Site visited 16/03/09.

Solon, A.J., P. Mc Kevitt & K. Curran (2007) TeleMorph: a fuzzy logic approach to network-aware transmoding in mobile Intelligent Multimedia presentation systems, Special issue on Network-Aware Multimedia Processing and Communications, A. Dumitras, H. Radha, J. Apostolopoulos, Y. Altunbasak (Eds.), IEEE Journal Of Selected Topics In Signal Processing, 1(2) (August), 254-263.

Spirtes, P., C. Glymour & R. Scheines (2000) Causation, Prediction, and Search, 2nd Edition, Cambridge, MA: MIT Press.

Stefánsson, S.F., Jónsson, B.T. & K.R. Thórisson (2009) A YARP-Based Architectural Framework for Robotic Vision Applications. In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP'09). February 5-8, Lisboa, Portugal, 65-68.

Stock, O. & M. Zancanaro (2005) Multimodal Intelligent Information Presentation (Text, Speech and Language Technology), Dordrecht, The Netherlands: Springer.

Sunderam, V.S. (1990) PVM: a framework for parallel distributed computing. In Concurrency Practice and Experience, 2(4), 315-340.

SW (2009)
Semantic Web. http://www.w3.org/2001/sw/ Site visited 16/03/09.

Thórisson, K. (1996) Communicative Humanoids: A Computational Model of Psychosocial Dialogue Skills. Ph.D. Thesis, Media Arts and Sciences, Massachusetts Institute of Technology, USA.

Thórisson, K. R. (1997) Gandalf: An Embodied Humanoid Capable of Real-Time Multimodal Dialogue with People. In the First ACM International Conference on Autonomous Agents, Mariott Hotel, Marina del Rey, California, February 5-8, 536-7

Thórisson, K. (1999) A Mind Model for Multimodal Communicative Creatures & Humanoids. In International Journal of Applied Artificial Intelligence, Vol. 13 (4-5), 449-486.

Thórisson, K. R. (2002) Natural Turn-Taking Needs No Manual: Computational Theory and Model, from Perception to Action. In B. Granström, D. House, I. Karlsson (Eds.), Multimodality in Language and Speech Systems, 173-207. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Thórisson, K. R., C. Pennock, T. List & J. DiPirro (2004) Artificial Intelligence in Computer Graphics: A Constructionist Approach. Computer Graphics Quarterly, 38(1), New York: ACM, 26-30.

Thórisson, K.R., T. List, C. Pennock, & J. DiPirro (2005) Whiteboards: Scheduling Blackboards for Semantic Routing of Messages & Streams, AAAI-05 Workshop on Modular Construction of Human-Like Intelligences, K.R. Thórisson (Ed.), Twentieth Annual Conference on Artificial Intelligence, Pittsburgh, PA, July 10, 16-23.

Thórisson, K. R. (2007) Avatar Intelligence Infusion - Key Noteworthy Issues. Keynote presentation, 10th International Conference on Computer Graphics and Artificial Intelligence, 3IA 2007, Athens, Greece, May 30-31, 123-134.

Turunen, M. & J. Hakulinen (2000) Jaspis - A Framework for Multilingual Adaptive Speech Applications, In Proceedings of the Sixth International Conference on Spoken Language Processing, Beijing, China, October 16-20, 719-722.

Vybornova, O., M. Gemo & B. Macq (2007) Multimodal Multi-Level Fusion using Contextual Information. In ERCIM NEWS, No. 70, July, 61-62.

Vinoski, S. (1993) Distributed object computing with CORBA, C++ Report, Vol. 5, No. 6, July/August, 32-38.

W3C (2009)
http://www.w3.org Site visited 16/03/09.

W3C XML (2009)
http://www.w3.org/XML/Activity.html Site visited 16/03/09.

Wahlster, W., E. André, S. Bandyopadhyay, W. Graf & T. Rist (1992) WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation. In Communication from Artificial Intelligence Perspective: Theoretical and Applied Issues, J. Slack, A. Ortony & O. Stock (Eds.), 121-143, Berlin, Heidelberg: Springer Verlag.

Wahlster, W., N. Reithinger & A. Blocher (2001) SmartKom: Towards Multimodal Dialogues with Anthropomorphic Interface Agents. In: Wolf, G. & G. Klein (Eds.), 23-34, Proceedings of International Status Conference, Human-Computer Interaction. October, Berlin, Germany: DLR.

Wahlster, W. (2003) SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In: Krahl, R. & D. Günther (Eds.), 47-62, Proceedings of the Human Computer Interaction Status Conference, June. Berlin, Germany: DLR.

Wahlster, W. (2006) (Ed.) SmartKom: Foundations of Multimodal Dialogue Systems, Berlin, Germany: Springer-Verlag.

Waibel, A., M.T. Vo, P. Duchnowski & S. Manke (1996) Multimodal Interfaces. In Artificial Intelligence Review, Vol. 10, Issue 3-4, August, 299-319.

Webb, N., M. Hepple & Y. Wilks (2005) Dialog act classification based on intra-utterance features. In Proceedings of the AAAI Workshop on Spoken Language Understanding.

Weilhammer, K., J.D. Williams & S. Young (2005) The SACTI-2 Corpus: Guide for Research Users. Technical Report CUED/F-INFENG/TR.505, Department of Engineering, Cambridge University, England, February.

Wilson, A. & H. Pham (2003) Pointing in Intelligent Environments with the WorldCursor, Interact.

Wilson, A. & S. Shafer (2003) XWand: UI for Intelligent Spaces. In Proceedings of the SIGCHI conference on human factors in computing systems, Ft. Lauderdale, Florida, USA, April 5-10, 545-552.

Zadeh, L. (1965) Fuzzy sets. Information and Control, 8(3), 338-353.

Zarri, G.P. (1997) NKRL, a Knowledge Representation Tool for Encoding the Meaning of Complex Narrative Texts. In Natural Language Engineering, 3, 231-253.

Zarri, G.P. (2002) Semantic Web and knowledge representation. In Proceedings of the 13th International Workshop on Database and Expert Systems Applications, September, 75-79.

Zou, X. & B. Bhanu (2005) Tracking Humans using Multi-modal Fusion. In Computer Society Conference on Computer Vision and Pattern Recognition (CVPRW'05), San Diego, California, USA, 4-11.