MediaHub:
An Intelligent Multimedia Distributed Platform Hub

Glenn Campbell
Supervisors: Dr. Tom Lunney, Prof. Paul Mc Kevitt

First year report
Date: 22 June 2005

Presented as a requirement for Ph.D. in
School of Computing and Intelligent Systems
Faculty of Engineering
University of Ulster, Magee
Email: Campbell-g8@ulster.ac.uk

Abstract

The primary objective of the research presented in this report is the development of an intelligent multimedia distributed platform hub (MediaHub) for the fusion and synchronisation of language and vision data. MediaHub will integrate and synchronise language and vision data in such a way that the two modalities are complementary to each other. The main objectives of MediaHub are to interpret/generate semantic representations of multimodal input/output and perform fusion and synchronisation of multimodal data (decision-making).

Research related to the design and implementation of MediaHub is reviewed, including an investigation into the area of language and vision integration. The area of multimodal semantic representation is considered, with focus being placed on both the frame-based and XML-based methods of semantic representation. An analysis of several existing multimodal platforms is performed, with particular attention given to their method of communication, semantic representation, semantic storage and decision-making. Multimodal corpora and annotation tools are discussed that could be used in the development of MediaHub. Also included is a review of several speech mark-up language specifications and a discussion on various Artificial Intelligence techniques, in particular Bayesian Networks, which allow reasoning and decision-making under uncertainty.

The proposed system architecture of MediaHub is presented and potential tools are reviewed. A unique contribution of the research is identified detailing how MediaHub, using Bayesian Networks for decision-making, improves on current systems. MediaHub will be developed using existing software tools and tested within an existing multimodal platform.

Keywords: intelligent multimedia, distributed systems, multimodal synchronisation, multimodal fusion, multimodal semantic representation, intelligent multimedia interfaces, decision-making, Bayesian networks.

Acknowledgements

I would like to thank my first supervisor Dr. Tom Lunney for his unending support and guidance. I also wish to acknowledge the valuable contribution of Prof. Paul Mc Kevitt, my second supervisor. Both Tom and Paul have been of great assistance to me in helping me adapt to life as a research student and their knowledge in the areas of distributed computing and intelligent multimedia has been greatly beneficial. I am also grateful for the support given to me by all the research students and lecturers in the School of Computing and Intelligent Systems, Faculty of Engineering.

Table of Contents

1. Introduction

In this chapter the background to the current research will be presented, focusing mainly on the areas of intelligent multimedia and distributed computing. A summary of the aims and objectives of the research is also presented.

1.1 Background

The area of intelligent multimedia has in recent years seen considerable research into the creation of user interfaces that can accept multimodal language and vision input. This has led to the development of intelligent interfaces that can learn to meet the needs of the user, in contrast to traditional systems where the onus was on the user to learn to use the interface. A more natural form of human-machine interaction has resulted from the development of systems that allow multimodal input such as natural language, eye and head tracking and 3D gestures (Maybury 1993). Considerable work has also been completed in the area of knowledge representation, with the development of several semantic mark-up languages. Efforts have also been made to integrate natural language and vision processing, and some of the approaches in this field are described in Mc Kevitt (1995/1996). The area of distributed computing has been exploited to create intelligent multimedia systems that are human-centred and directly address the needs of the user. The Oxygen project (Oxygen 2005) is concerned with developing pervasive systems, embedded in the world around us, that adapt to changes in user requirements. Psyclone (2005) is a powerful platform designed to aid the creation of modular, distributed systems. DACS (Fink et al. 1995, 1996) is an equally powerful tool that provides numerous features for the development and maintenance of distributed systems. CHAMELEON (Brøndsted et al. 1998, 2001) is a platform for developing multimedia applications that makes use of DACS for process synchronization and intercommunication.

1.2 Objectives of this Research

The principle aim of this research is to investigate the fusion of multimodal data through establishing more intelligent decision-making in a distributed environment. This will involve the development of a test platform hub, namely MediaHub. The primary objectives of this research are to:

- Interpret/generate semantic representations of multimodal input/output.
- Perform fusion and synchronisation of multimodal data (decision-making).
- Implement and evaluate MediaHub, a multimodal platform hub with a potential new approach to decision-making.

In pursuing the three objectives outlined above, several research questions will need to be answered. For example:

- Will MediaHub use frames for semantic representation, or will it use XML or one of its derivatives?
- How will MediaHub communicate with various elements of a platform?
- Will MediaHub constitute a blackboard or non-blackboard model?
- What mechanism will be implemented for decision-making within MediaHub?

These questions will be answered in the design and implementation of MediaHub – an intelligent multimedia distributed platform hub to facilitate the integration of multimodal data. MediaHub will be tested within an existing multimodal platform such as CONFUCIUS (Ma & Mc Kevitt 2003) using multimodal input/output data.

2. Literature Review

This chapter focuses on reviewing a variety of areas related to the development of MediaHub. The main focus of the chapter will be a review of the various multimodal distributed systems and platforms for intelligent multimedia. First, there will be a discussion on the integration of language and vision. We will then look at the area of multimodal semantic representation, with focus being placed on both the frame-based and XML-based methods of semantic representation. An analysis of several existing multimodal platforms will then be performed, with particular attention given to their method of communication, semantic representation, semantic storage and decision-making. Various existing tools for distributed processing will then be considered, followed by a review of multimodal corpora and annotation tools. Also included in this chapter is a review of several speech mark-up language specifications and a discussion on various Artificial Intelligence techniques, in particular Bayesian Networks, which allow reasoning and decision-making under uncertainty.

2.1 Integration of Language and Vision

Integration of language and vision refers to the fusion of language and vision input and output in multimodal systems. Systems capable of integrating speech and gestures have been proposed since the early eighties. Bolt (1980) introduced the 'Put-That-There' system, which allows the user to move shapes about a graphical display using speech commands and pointing gestures. Mc Kevitt (1995/96) presents a collection of computational models and systems that have been designed with this in mind. Ó Nualláin & Smith (1994) investigates the relationship between the semantics of language and vision, which led to the development of the Spoken Image system (Ó Nualláin et al. 1994).The Spoken Image system allows a user to quickly build an envisaged house or town scene, within a 3D environment, by describing the scene using natural language. The Aesopworld project (Okada 1996; Okada et al. 1999) aims to create an architectural foundation of intelligent agents. The project has involved the creation of a computational agent that simulates various kinds of mental activities. The QuickSet system (Johnston et al. 1997, Johnston 1998) focuses on the integration of speech with pen-based input. QuickSet is a distributed system that consists of a collection of interacting agents that communicate using the Open Agent Architecture (Cheyer et al. 1998, OAA 2005). QuickSet supports more direct manipulation by allowing more complex pen gestures, such as arrows and various types of lines. The QuickSet interface allows users to manipulate an intelligent map using natural language and pen-based gestures. Semantic representation within QuickSet is in the form of typed feature structures (Carpenter 1992). Hall & Mc Kevitt (1995) observe that language and vision are combined by humans in a non-linear fashion. This 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 (2004a) 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) achieved 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 exist for integrating language and vision. Three common integration mechanisms, as discussed in Blattner & Glinert (1996) are frames, neural networks and agents. Frames are commonly used to hold data from different input devices. Neural Networks and agents have also been used for language-vision integration. Neural Networks were first proposed more than 50 years ago and are now used extensively in the area of Artificial Intelligence. An agent is a software program that is used to perform a specific task. Thus it is possible for a system to delegate the task of integrating language and vision data to a software agent or program.

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. The term 'semantic representation' refers to the method employed to represent the meaning of media representation (Mc Kevitt 2003). This semantic representation must support interpretation and generation, multimodal input and output and a variety of semantic theories. The majority of the work in multimodal systems employs

either frames or XML as the method of semantic representation. A frame is a collection of attributes 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 concept of frames is based on human memory and the psychological view that when faced with a new problem, humans select an existing frame (remembered framework) and adapt it to fit the new situation by changing appropriate details. In multimodal systems frames can be grouped into the following three categories:

- Input frames - These come from modules processing perceptual input.
- Output frames - Produced by modules generating system output.
- Integration frames - These are integrated meaning representations that are built through the course of the dialogue (i.e. all other frames).

Besides frames, the other most common method of semantic representation in multimodal systems is XML (eXtensible Mark-up Language) (Klein 2002, W3C XML 2005, TopXML 2005). XML, created by W3C (World Wide Web Consortium), is a derivative of SGML (Standard Generalised Mark-up Language). XML was originally designed for use in large-scale electronic publishing but is now used extensively in the exchange of data via the web. XML documents contain both parsed and unparsed data, with the former being either mark-up or character data (data between a pair of start and end mark-ups). The mark-up encodes a description of the storage layout and logical structure of the document. A mechanism is provided within XML that allows constraints to be imposed on the storage layout and logical structure. The main purpose of XML is to provide a mechanism that can be used in the mark-up and structuring of documents.

XML is different to HTML in that tags are only used within XML to delimit pieces of data. The interpretation of the data is left completely to the application that reads it. Another advantage of using XML is that it is possible to easily create new tags in XML. For example, the following XML tags could be defined to describe Glenn Campbell:

<name> Glenn Campbell </name>
<university> University of Ulster </university>
<title_of_research> An Intelligent Multimedia Distributed Platform Hub </title_of_research>

Any programming language can be used to manipulate data in XML and a large amount of middleware exists for managing data in XML format.

The Synchronised Multimedia Integration Language (SMIL) (Rutledge 2001, Rutledge & Schmitz 2001, SMIL 2005a,b) is a representation language used to encode multimedia presentations for delivery over the web. The SMIL specification is XML-based and was first released by the World Wide Web Consortium (W3C 2005) in 1998. SMIL consists of a collection of XML elements and attributes that describe the temporal and spatial coordination of media objects. The focus of SMIL is on the integration and synchronisation of independent media.

RDF Schema (Klein 2001, 2002, Nejdl et al. 2000, RDF Schema 2005) was proposed to the W3C (World Wide Web Consortium) in February 2004. The Resource Description Framework (RDF) is a general-purpose language for representing information on the Web. RDF Schema, a semantic extension of RDF, provides mechanisms for describing groups of related resources and the relationships between these resources. The RDF class and property system is similar to that of object-oriented programming languages such as Java. However in RDF, instead of defining a class in terms of the properties its instances may have, the RDF vocabulary description language describes properties in terms of the classes of resource to which they apply. The RDF Schema specification is made up of basic classes and properties and it can be extended if necessary to fit different domains.

MPEG-7 (MPEG7 2005), also known as “Multimedia Content Description Interface”, is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). The MPEG-7 standard provides a rich set of tools to describe multimedia content. The aim of the MPEG-7 standard is to allow interoperable searching, indexing, filtering and access of audio-visual (AV) content by enabling interoperability among devices and applications that deal with AV content description. It is expected that MPEG-7 will make the web more searchable by using multimedia

content as the search criteria, as opposed to the traditional method of searching for text. The MPEG-7 library (Fürntratt et al. 2004, MPEG-7 Library 2005), released in 2004, implements the MPEG-7 standard using a set of C++ classes. The library application allows developers to create a multimedia content description, serialise it to, and de-serialise it from, XML.

The Semantic Web (SW 2005) aims to provide a common framework that allows data to be shared and reused across application, enterprise and community boundaries. Based on the Resource Description Framework (RDF), it is a joint effort led by the W3C (World Wide Web Consortium) with participation from other researchers and industrial partners. According to Berners-Lee et al. (2001), "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 project is to put information on the web that can be processed by machines as well as by humans. Semantic Web technologies such as RDF, OIL (Ontology Inference Layer) and OWL (Ontology Web Language) enable rich information networks to be rapidly created and can assist in the generation of hypotheses across large sets of data. OWL (2005) is designed for use by applications that need to process the content of information instead of just 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 along with a formal semantics. OWL is based on DAML+OIL (DAML 2005, DAML+OIL 2005, Mc Guinness et al. 2002) and has three increasingly expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

OIL (Ontology Inference Layer) (Fensel et al. 2001) is an ontology architecture that is being developed for the Semantic Web. An ontology, as defined by Gruber (1993), is " a formal, explicit specification of a shared 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. The foundation for the development of OIL is RDF Schema, which Fensel et al. (2001) have developed into a complete web-based ontology language to express and represent ontologies. The syntax used by OIL is based on a document type definition (DTD) and an XML Schema definition. Fensel et al. (2001) state that existing keyword-based searching for information on the web frequently return irrelevant information that uses the keywords in a different context. Thus users can miss information by using different keyword relating to the desired content. It is hoped that through the use of ontologies, the semantic representations will allow the intelligent searching of the semantic web, instead of the traditional method of keyword matching.

DAML + OIL (DAML 2005, DAML+OIL 2005, Mc Guinness et al. 2002) evolved from the merging of DAML + ONT, an earlier ontology language, and OIL. The Defence Advanced Research Projects Agency (DARPA) launched the DARPA Agent Mark-up Language (DAML) initiative and the first version of DAML + OIL was released in December 2000. DAML + OIL is built upon XML, RDF and RDF Schema. The latest version of DAML + OIL provides a useful set of constructs to create ontologies and mark-up information, making it machine readable and understandable. 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 2005), which provides a set of mark-up language constructs to describe web services in a format understandable to computers. The DAML program now aims to develop a query language and integrate a rule-encoding option. It is hoped DAML-Logic, the next language enhancement, will enable the encoding of inference and general implications.

SOAP (Simple Object Access Protocol) (Chester 2001) is an emerging W3C standard that allows software applications written in different 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. The HTML is used for communication between modules, while the actual network conversation is encoded in XML. SOAP is a network protocol that can be used to call components, methods, objects and services on remote servers. This is done by representing parameters, return values and errors in an XML document known as a SOAP envelop. A

standardised XML vocabulary called WSDL (Web Services Definition Language) is used to describe 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 allows information exchange in a decentralised, distributed environment.

NKRL (Narrative Knowledge Representation Language) (Zarri 1997, 2002) is a language that is used to represent the semantic content of complex multimedia information in a standardised way. That is, it is a language for representing the meaning of natural language narrative documents such as, for example, news stories and corporate documents. Within NKRL there exists a standard set of basic templates that can be used, thus a system-builder need not create the structural knowledge necessary to describe a relatively large class of narrative documents. This catalogue of standard templates also allows previous results to be easily shared and reproduced.

2.3 Multimodal Platforms

Several multimodal platforms have been developed that can assist the creation of intelligent multimodal systems. These multimodal platforms are capable of dealing with input in a wide range of modalities including speech, visual, gesture and tactile, amongst others. These platforms enable systems to be developed that can communicate with users using the entire range of human communication modalities. The result of enabling such rich multimodal input 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.

2.3.1 WAXHOLM

The WAXHOLM Project (Carlson & Granström 1996, Carlson 1996) aims to deliver a generic system that will combine speech synthesis and speech recognition in a human-computer dialogue framework. The demonstrator application, called WAXHOLM, gives information on boat traffic. Besides the spoken language capabilities, the system contains modules to handle graphical information such as pictures, maps, charts and timetables. WAXHOLM uses the standardised query language 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 and the specific application. In order to decide on the topic, each semantic feature found in the semantic and syntactic analysis is considered in the form of a conditional probability. The probability is expressed in the form p(topic | F), where F is a feature vector including all semantic features in the utterance. The topic prediction is trained using a set of utterances taken from the WAXHOLM database. WAXHOLM implements a non-blackboard model of semantic storage.

2.3.2 Spoken Image/SONAS

The Spoken Image project and its successor, the SONAS system, (Ó Nualláin et al. 1994, Ó Nualláin & Smith 1994, Kelleher et al. 2000) concentrate on developing a system for interacting with a 3D environment using natural language, gestures and other modes of communication. The project aims to develop a system that will allow users to communicate and interact with it in a multimodal manner, inspired by the way humans communicate with ease using multiple modalities. The original project, Spoken Image, allowed a user to quickly build a house or town scene by describing the scene using 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 allows a combination of several input modalities. The SONAS system allows objects in a 3D environment to be manipulated using natural language. Consider the following example, as discussed in Kelleher et al. (2000). With the input: “Put the book on the table”, the user should see 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 geometric points, lines, planes etc.

An important theme in the Spoken Image/SONAS project has been an effort to find a common semantics between language and vision. That is, 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). Spoken Image/SONAS uses frames for semantic representation and uses a blackboard for semantic storage. Whilst it could be argued that the human ability of representing language and vision may never be completely replicated by a computer, projects such as Spoken Image/SONAS attempt to move us ever closer to that ideal.

2.3.3 AESOPWORLD

The AESOPWORLD (Okada 1996, Okada et al. 1999) project aims to create an architectural foundation of intelligent agents. The project has involved the creation of a computational agent that simulates various kinds of mental activities. An important objective of the AESOPWORLD project is to develop human-friendly interfaces that can make decisions on a dialogue based on the user's facial expressions, gestures or 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 2.1.

```
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 2.1: An example AESOPWORLD frame (Okada et al. 1999)

In its efforts to develop a truly intelligent agent, the AESOPWORLD project attempts to integrate the following seven intelligent activities: recognition, planning, action, desire, emotion, memory and language.

2.3.4 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, the participants need to have mutual beliefs about the goals/actions to 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. Data structures and algorithms are provided within Collagen to represent and manipulate goals, actions, recipes and SharedPlans. Figure 2.2

illustrates how a user can collaborate with an agent using Collagen, while Figure 2.3 shows the internal architecture of Collagen.

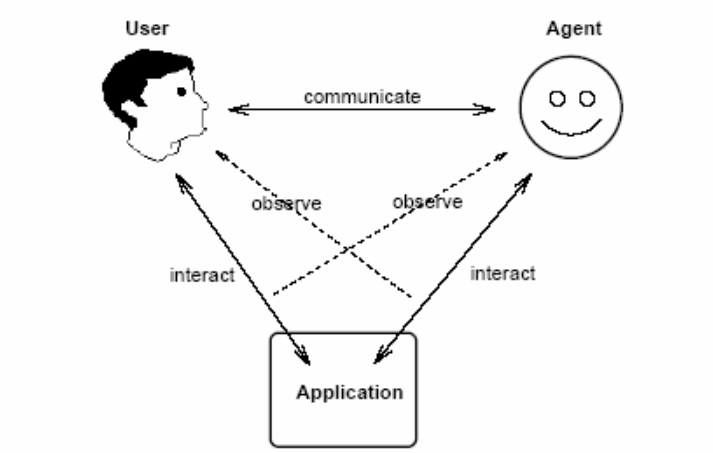

Figure 2.2: User-Agent Collaboration within Collagen (Rich & Sidner 1997)

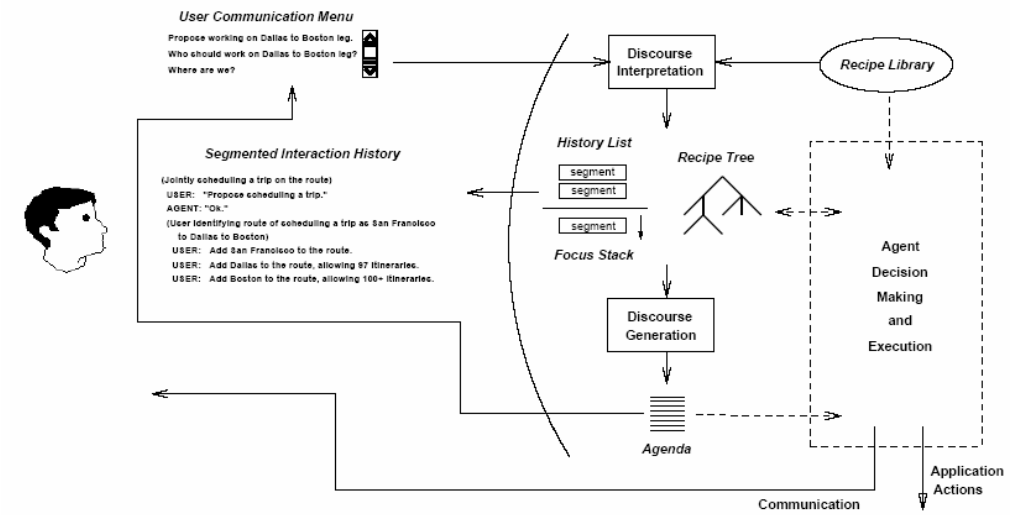

Figure 2.3: Collagen architecture (Rich & Sidner 1997)

Note from Figure 2.3, 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 provide a method of decision-making – this is left to the discretion of the developer. Collagen uses Sidner's (1994) artificial discourse language as the internal representation for agent communication acts. Within the artificial discourse language there is a set of constructors for basic act types, such as 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$_1$* believes *belief*, communicates this belief to *participant$_2$* and intends *participant$_2$* to believe it also. If *participant$_2$* now responds with an AP act (i.e. accepts the proposal), then *belief* is considered mutually believed. There is two additional application-independent operators to model a belief about an action, SHOULD (act) and RECIPE (act, recipe). The rest of the belief sublanguage is specific to each application. Collagen uses a frame-based method of semantic representation and implements a non-blackboard model for semantic storage.

2.3.5 INTERACT

The INTERACT project (Waibel et al. 1996) aims to remove the limitations associated with traditional computer interfaces by allowing input using the entire range of human communication modalities including speech, gesture, eye-gaze, lip motion, facial expression, face recognition, 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) is used to perform recognition of visual data. The network architecture of the system is shown in Figure 2.4. As shown in Figure 2.4, 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 2.5 presents a table of recognition performances for the Speech/Lip system. As illustrated in Figure 2.5, 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.

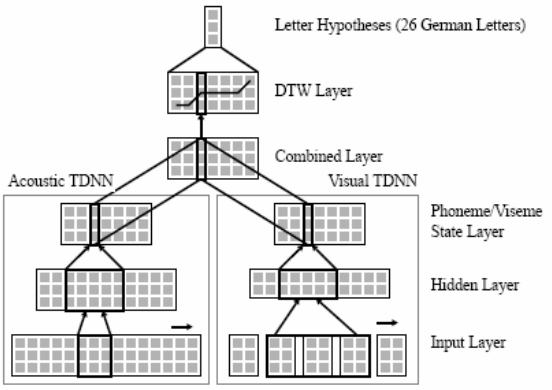

Figure 2.4: Network architecture of INTERACT Audio/Visual ASR system (Waibel et al. 1996)

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

Figure 2.5: Word accuracy of Speech/Lip system (Waibel et al. 1996)

2.3.6 Ymir

Ymir (Thórisson 1999) is a computational model for creating autonomous creatures capable of human-like communication with real users. Ymir represents a distributed, modular approach that bridges between multimodal perception, decision and action in a coherent framework. Within the Ymir architecture, a prototype agent called Gandalf 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 generate suitable responses.
- The Action Scheduler (AS), which coordinates appropriate actions.

There are three main blackboards implemented in Ymir. The first blackboard, called the Functional Sketchboard, is primarily used for information exchange between the Reactive Layer and the Process Control Layer. The second blackboard is called the Content Blackboard. This 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 on the Functional Sketchboard. The third blackboard is called the Motor Feedback Blackboard and is used to keep track of which part of a stream of actions is currently being planned or carried out by the Action Scheduler (AS). Ymir uses a frame-based method of semantic representation. Within the Ymir prototype, most messages take the form [<Message>,<State>,<Timestamp], with state being either true or false. The following is an example of a message posted on the Functional Sketchboard:

(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)

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 effecting the simplicity or performance of the system. According to Thórisson (1999), Ymir looks promising in providing a general framework for communicative, task-knowledgeable agents.

2.3.7 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 learner

The ten modules of CHAMELEON are glued together by the DACS communication system (Fink et al. 1995, Fink et al. 1996). The hub of CHAMELEON consists of a dialogue manager and a blackboard. The role of the blackboard is to keep track of interactions over time, using frames for semantic representation. The architecture of CHAMELEON is shown in Figure 2.6.

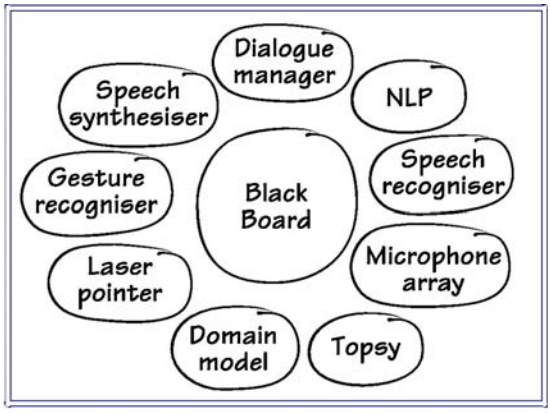

Figure 2.6: Architecture of CHAMELEON (Brøndsted et al. 1998, 2001)

The blackboard and dialogue manager form the kernel of CHAMELEON. The blackboard stores the semantic representations produced by the other modules, keeping a history of all interactions. Communication between modules is achieved by exchanging semantic representations between themselves or the blackboard. Figure 2.7 shows how the blackboard acts as a mediator for information exchange between the modules of CHAMELEON.

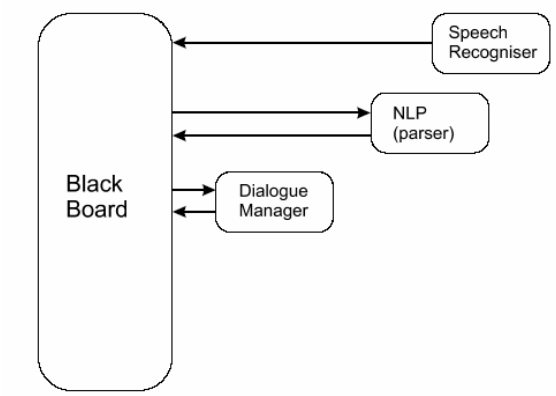

Figure 2.7: Information exchange using the blackboard
(Brøndsted et al. 1998, 2001)

The blackboard consists of the following three main parts:

- SQL database
- Communication interface
- Table of implicit requests

The internal architecture of the blackboard is shown in Figure 2.8.

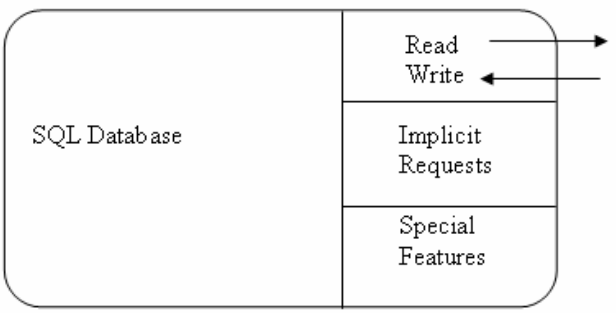

Figure 2.8: Internal blackboard Architecture

The semantic representation is in the form of input, output and integration frames for representing the meaning of user input and system output. Frames are coded as messages and are passed between modules using the DACS communication system. Figure 2.9 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.

An initial prototype application for CHAMELEON is the IntelliMedia WorkBench (Brøndsted et al. 2001). The current domain 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 within CHAMELEON contains all the relevant data for the Campus Information System, including information on rooms, their function and their occupants. Figure 2.10 shows an extract from a file containing the description of the physical environment. Hierarchical levels are illustrated using 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 are used to store information about the occupants of the building. The functionality provided by the domain model is to answer questions relating to the environment using 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 struct holding x and y coordinates, while the last two functions return a string containing the name of the person or place. The domain module is implemented in C. The program is relatively straightforward, with static information stored in simple text files. Inputs to CHAMELEON can include synchronised spoken dialogues and images, and outputs include synchronised spoken dialogue and laser pointing.

```
FRAME          ::= PREDICATE

PREDICATE      ::= identifier(ARGUMENTS)

ARGUMENTS      ::= ARGUMENT
                 | ARGUMENTS, ARGUMENT

ARGUMENT       ::= CONSTANT
                 | VARIABLE
                 | PREDICATE

CONSTANT       ::= identifier
                 | integer
                 | string

VARIABLE       ::= $identifier
```

Figure 2.9: Syntax of messages (frames) within CHAMELEON
(Brøndsted et al. 1998, 2001)

```
area FRB7  0 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 2.10: Physical environment data (Brøndsted et al. 1998)

2.3.8 Oxygen

The Oxygen project (Oxygen 2005), developed by MIT, is motivated towards making computing available to everyone, everywhere in the word – just as accessible as the oxygen we breathe. Some of the aims of the Oxygen project are to develop 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 needed
- adaptable to changes in user requirements
- intentional, i.e. enabling people to name a service or software object by intent, for example, "the closest printer," as opposed to by address

The meeting of these objectives will create 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 will 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. Another important theme that is considered in the design of Oxygen is semantic content, i.e. the ability to understand what the user means, not just what the user says. The semantic representation is in the form of frames, whilst the semantic storage is implemented using a non-blackboard model.

2.3.9 SmartKom

SmartKom (Wahlster 2003, Wahlster et al. 2001, SmartKom 2005) is a multimodal dialogue system that is being developed to help overcome the problems of interaction between people and machines. The project focuses on developing multimodal interfaces for applications in the home, public and mobile domains. The system uses 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. For example, in the public domain, the user can allocate the task of finding a library to Smartakus. Along with its language capabilities, Smartakus uses facial expressions and body language to provide a more natural form of human-computer interaction. It is intended that SmartKom will enable complex dialogic interactions, were 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. The system allows 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), is being developed to provide semantic representation of information that is passed between the various components of SmartKom. M3L covers all data interfaces within the dialogue system and the information exchanged through the various blackboards is encoded in M3L. An example of the M3L code used by SmartKom is shown in Figure 2.11. SmartKom also makes use of the OIL ontology language (Fensel et al. 2001) to represent domain and application knowledge.

SmartKom is based on a distributed architecture and constitutes a multi-blackboard system, including more than 40 asynchronous modules coded in C, C++, Java and Prolog. The integration platform is called MULTIPLATFORM (Multiple Language Target Integration Platform for Modules) (Herzog et al. 2003), which allows the creation of an open, flexible and scalable software architecture. The MULTIPLATFORM Testbed uses a message-based middleware, based on 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 several application scenarios.

```
<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 2.11: Example of M3L code (Wahlster 2003)

2.3.10 DARBS

DARBS (Distributed Algorithmic and Rule-Based System) (Choy et al. 2004a, 2004b, Nolle et al. 2001) is a distributed system that allows 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 allowed one knowledge source to operate at any one time, so 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 2.12 shows the architecture of DARBS. As shown, the system comprises rule-based, procedural, neural network and genetic algorithm knowledge sources operating in parallel. DARBS uses frames for semantic representation.

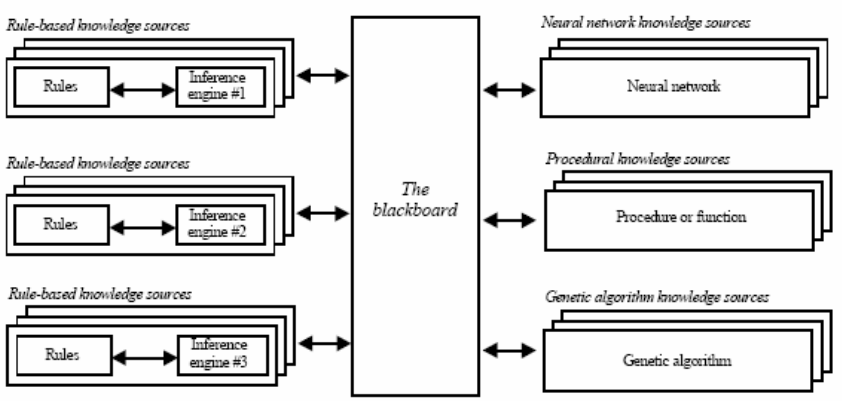

Figure 2.12: Architecture of DARBS (Nolle et al. 2001)

The major advantage that DARBS offers over its predecessor is the introduction of 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 2.13.

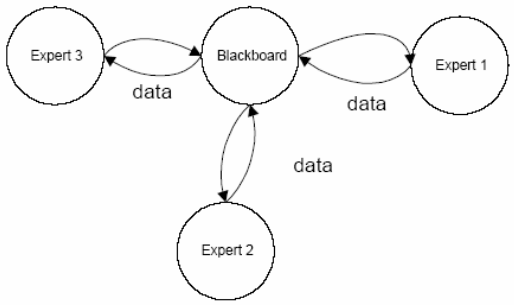

Figure 2.13: Communication within DARBS (Nolle et al. 2001)

The DARBS knowledge sources constantly observe 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 are used to look up information on the blackboard, write information to the blackboard and to make decisions about information on the blackboard. An example of a typical DARBS rule is shown in Figure 2.14.

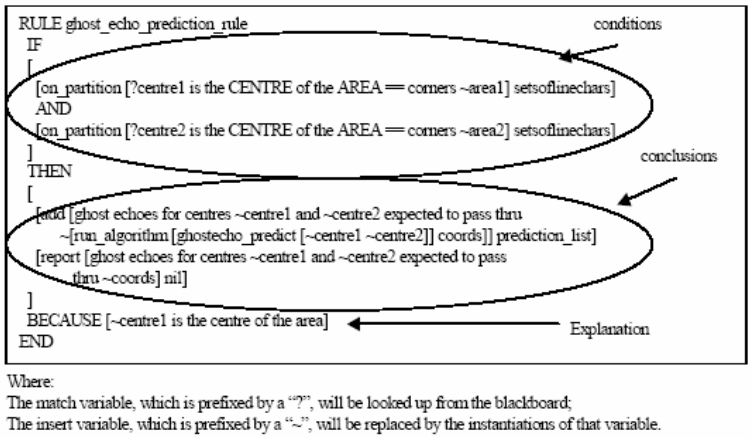

Figure 2.14: A typical DARBS rule (Nolle et al. 2001)

In order to test the system and demonstrate its flexibility, DARBS has been applied to several different AI applications, including interpreting ultrasonic non-destructive evaluation (NDE) and controlling plasma processes.

2.3.11 DARPA Galaxy Communicator

The DARPA Galaxy Communicator (Bayer et al. 2001) program is investigating 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). This is an extension of the Galaxy-II distributed infrastructure for dialogue interaction, which was developed by MIT.

The GCSI is a distributed hub-and-spoke architecture. It implements frames for semantic representation and communication facilitated via message-passing. The architecture of the system is illustrated in Figure 2.15. The system's hub allows 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.

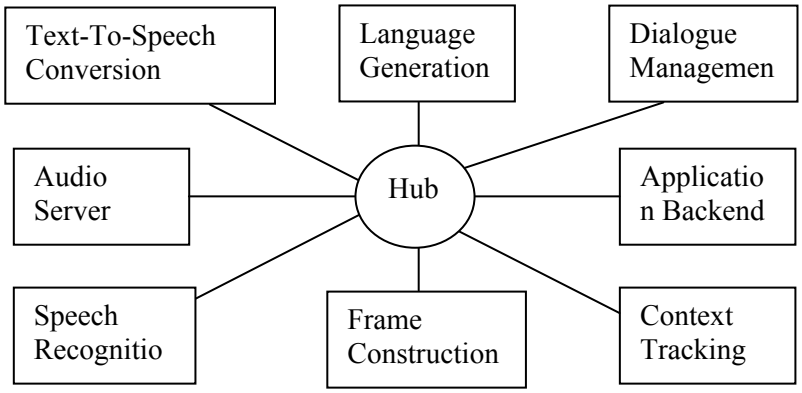

Figure 2.15: Hub-and-spoke architecture of GCSI

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 allows 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.

2.3.12 EMBASSI

The EMBASSI project (Kirste et al. 2001, EMBASSI 2005) aims to provide a platform that will give computer-based assistance to the user in achieving his/her individual objectives. That is, the computer will act as a mediator between users and their personal environment. 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 2.16, which shows the relationship between the user, the computer and the user's personal environment.

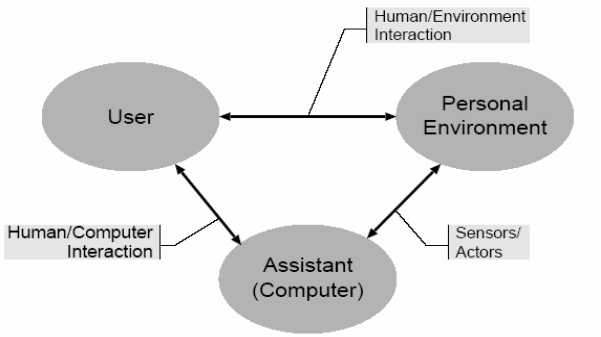

Figure 2.16: User-computer-environment relationship (Kirste et al. 2001)

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 2.17.

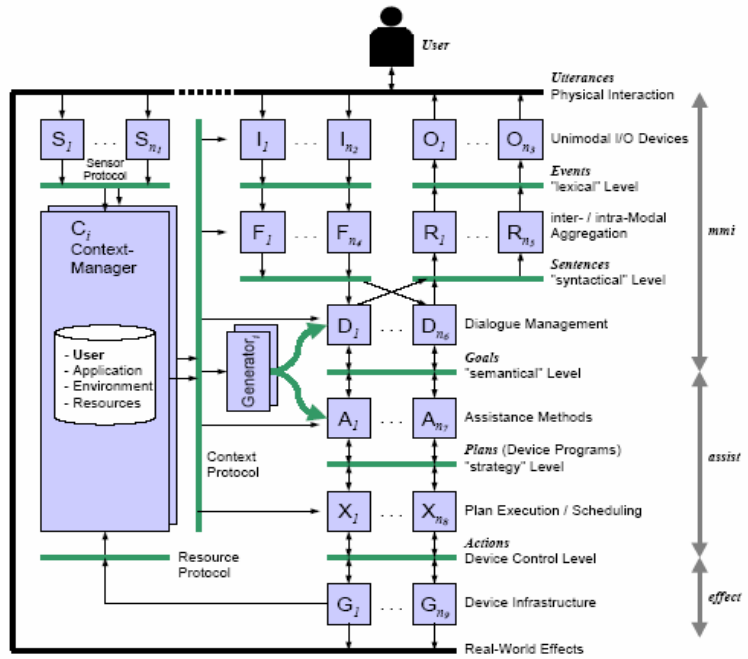

Figure 2.17: Generic architecture of EMBASSI (Kirste et al. 2001)

As shown in Figure 2.17, the MMI levels define 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. The KQML (Knowledge Query and Manipulation Language) Agent Communication Language (ACL) (Finin et al. 1994) is used as a messaging infrastructure, while XML (eXtensible Mark-up Language) (W3C XML 2005) is used as the content language. A non-blackboard 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 the lighting, temperature etcetera 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 music". Many other scenarios are possible where the user can simply express a goal and leave the required technical functionality to the EMBASSI platform.

2.3.13 MIAMM

MIAMM (Multidimensional Information Access using Multiple Modalities) (Reithinger et al. 2002, MIAMM 2005) aims to develop new concepts and techniques that will facilitate fast and natural access to multimedia databases using multimodal dialogues. This will involve the creation of a multimedia framework for the design of modular multimodal dialogue systems. The MIAMM project 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 allows 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 2.18.

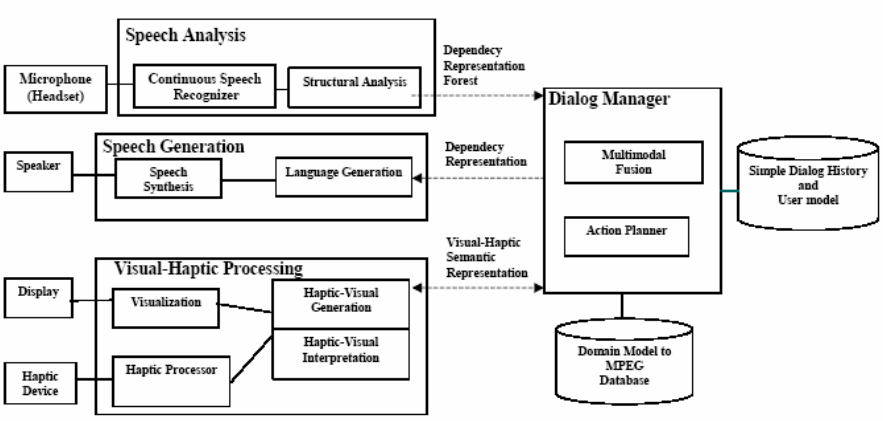

Figure 2.18: MIAMM general architecture (Reithinger et al. 2002)

The exchange of information within MIAMM is facilitated through the Multi-Modal Interface Language (MMIL), which is based on XML. The 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 in the design of the MMIL language is to allow for 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 model of semantic storage. Within MIAMM, a dialogue manager is used to combine information from the underlying application, the language modules, the haptic device and the graphical interface. As an example, suppose the user says the following:

"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 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 2.19 is then presented to the user.

Figure 2.19: Example MIAMM hand-held device (Reithinger et al. 2002)

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. The multimodal fusion then checks time and type constraints of each structure and the action planner uses the domain model to retrieve the relevant information from the database. Finally, the action planner sends a display order to the visual-haptic agent.

2.3.14 Psyclone

Psyclone (Psyclone 2005) is a powerful message-based middleware that can be used to simplify the creation of modular, distributed systems. Psyclone finds use in applications where complexity management or interactivity is of importance. Psyclone allows software to be easily distributed across multiple machines and allows communication to be managed using rich messages. The messages within Psyclone are formatted in XML. A design methodology called 'divisible modularity' is used to allow the incremental construction of multi-granular, heterogeneous systems. Psyclone, written in C++ for robustness, uses XML configuration files and bulletin boards to enable the easy set-up and testing of system architectures. Psyclone will run 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 is used to initialise values at start-up. Figure 2.20 shows an example psySpec.

```
<psySpec name="Project X" version="1.2">
  <global>
    <psyprobe>
      <enabled>yes</enabled>
    </psyprobe>
    <port>10000</port>
  </global>

 <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>
    <spec>
       <context name="System">
          <phase name="Look for System Created">
             <triggers from="any">
                <trigger type="System.Ready" after="100" />
             </triggers>
             <posts>
                <post to="WB1" type="Internal.Context:SoB.Alive" />
             </posts>
          </phase>
       </context>
    </spec>
 </module>

</psySpec>
```

Figure 2.20: XML Code to initialise Psyclone at start-up (Psyclone 2005)

The code in Figure 2.20 creates a whiteboard (WB1) and an internal module called 'Startup'. It is also possible to create multiple whiteboards and multiple modules for a system. In order to allow multiple programs to communicate with each other via the whiteboards, programs can use plugs written in Java, C++, Python and Lisp.

2.4 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 appears 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. 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 tools for distributed computing are available and an overview of a few of these will now be given.

2.4.1 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 caters for the development of large concurrent or parallel applications that consist of interacting, but relatively independent, components. An objective of the PVM project is to allow 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 enhanced overhead during reconfiguration.

2.4.2 ICE

ICE (Amtrup 1995) stands for INTARC Communication Environment and 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 graphics capabilities. At present 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 2.21. A system developed using ICE can be made up of several components that can be written in any programming language. Each component is given a unique name and can communicate with all other components within the system.

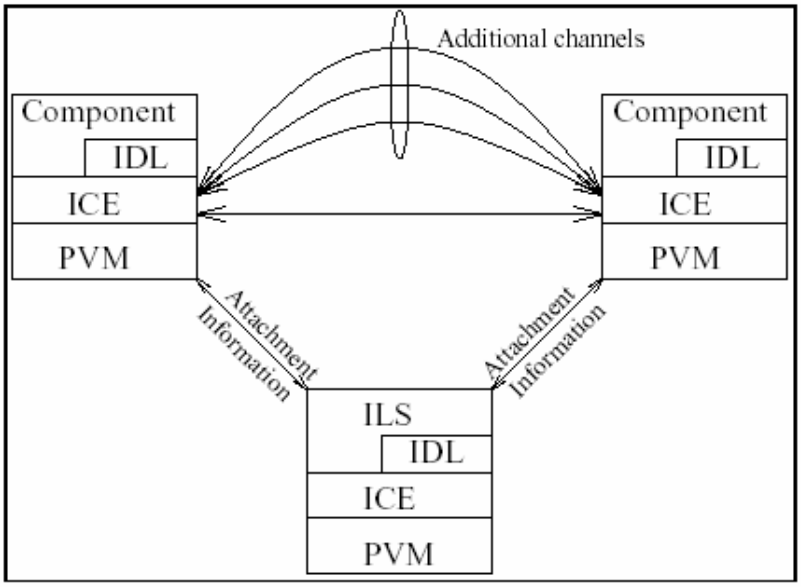

Figure 2.21: Typical structure of an ICE system (Amtrup 1995)

As shown in Figure 2.21, components communicate with each other via channels. ICE provides a base channel between components that uses the eXternal Data Representation (XDR) to encode data hardware-independent.

2.4.3 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 simple asynchronous message passing, with additional extensions to deal with dynamic system reconfiguration during 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 that are passed within DACS are encoded in a Network Data Representation, which makes it possible to inspect data at any point in the system and to develop generic tools capable of processing all kinds of data. Similar to PVM, DACS uses a communication demon that runs on each participating machine. DACS differs from PVM in that the communications demon allows multiple users to access the system simultaneously and does not provide a virtual machine dedicated to an individual user. The purpose of the DACS demon is to act as a router for all internal traffic and establish connections to other demons on remote machines. A central name server is used to keep 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 allows 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.

2.4.4 Open Agent Architecture (OAA)

The Open Agent Architecture (OAA) (Cheyer et al. 1998, OAA 2005) is a general-purpose infrastructure for creating systems that contain multiple software agents. OAA allows 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 2005, p.1). OAA distinguishes itself from other methods of distributed computing, such as the Blackboard approach, in that it allows both human users and software agents to express what they want done without specifying who is to do it 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 2.22 illustrates the agent interaction within an OAA-based system.

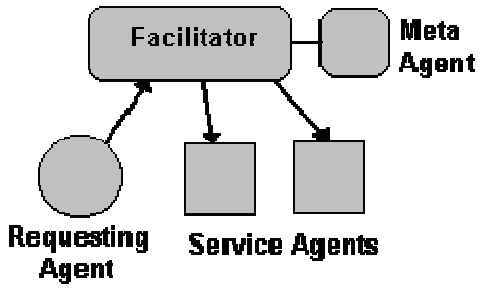

Figure 2.22: Agent interaction in OAA (OAA 2005)

As shown in Figure 2.22, the requesting agent issues a request to the Facilitator, which matches the request with an agent or agents providing that service. All agents interact using the Interagent Communication Language (ICL). ICL is a logic-based declarative language used to express high-level, complex tasks and natural language expressions. A major advantage of OAA is the ability to add new agents on the fly.

2.4.5 JavaSpaces

JavaSpaces (Freeman 2005), developed by Sun Microsystems, are a simple but powerful distributed programming tool that allows developers to quickly create collaborative and distributed applications. JavaSpaces represent a new 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 objects to a space, take objects from a space, or read objects in a space (i.e. make a copy of an object). These operations are illustrated in Figure 2.23.

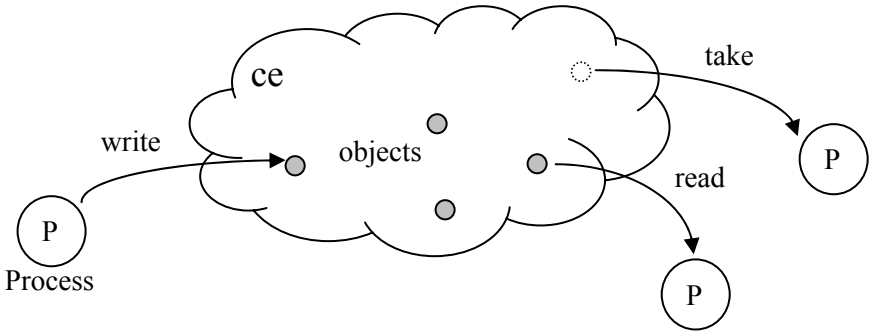

Figure 2.23: read, write and take operations within JavaSpaces

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.

2.4.6 CORBA

The CORBA (Common Object Request Broker Architecture) (CORBA 2005, Vinoski 1993) specification was released by the Object Management Group (OMG) in 1991. CORBA is a standard architecture for distributed object systems, allowing 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, which is defined using the Interface Definition Language (IDL), 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 illustrated in Figure 2.24.

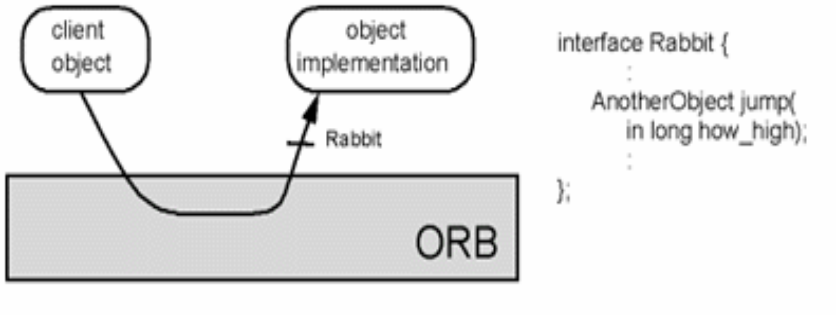

Figure 2.24: A typical object request (CORBA 2005)

Note that the client object holds a reference to the distributed object. This object reference is typed by an interface. In Figure 2.4 the Rabbit interface is used to type the object reference. 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 regardless 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 major objectives of the CORBA specification is to allow 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.

2.4.7 JATLite

JATLite (Kristensen 2001, Jeon et al. 2000), developed by the Stanford university, provides a set of Java packages that enable multi-agent systems to be constructed using Java. JATLite provides a Java agent platform that uses the KQML Agent Communication Language (ACL) for inter-agent communication. A major concept with the JATLite platform is the Agent Message Router (AMR). The AMR is responsible for registering agents with a name and password. JATLite features modular construction 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 message routing and queuing for agents via the AMR.

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

2.4.8 .NET

.NET (MS.NET 2005) is the Microsoft Web services strategy that allows 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 one another. Based on XML, the web services allow 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 .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 runtime, managing basic services such as memory management and error control. The .NET Framework Class Library is a collection of object-oriented types that can be used to develop applications, services and components. .NET allows the components of a system to be shared over the Internet. The framework makes use of Web Services that can be used by both Windows-based applications and applications 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 2.25.

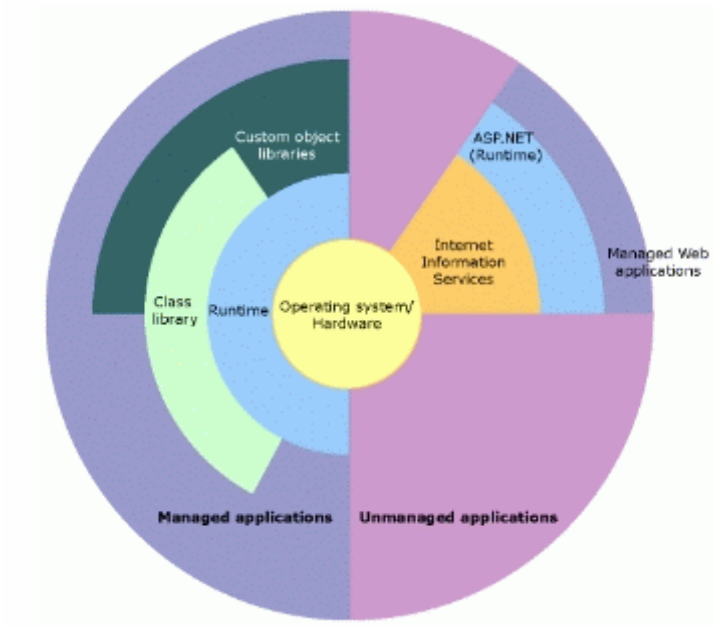

Figure 2.25: Architecture of .NET framework (MS.NET 2005)

As illustrated in Figure 2.25, .NET allows 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.

This section has discussed a few of the tools for distributed processing that could be utilised in the design and implementation of MediaHub. Further analysis of the capabilities of such tools will be performed before a decision is made on which processing tool can be used within MediaHub.

2.5 Multimodal Corpora and Annotation Tools

Many researchers are currently engaged the creation of multimodal corpora (Pastra & Wilks 2004a,b). This involves the recording and annotation of several communication modalities such as language, gesture, facial expression and body posture. Several organisations and projects have focused on this area and more information on multimodal resources can be found at LREC (2005). Pastra & Wilks (2004b) observe that, although there is an increasing need for multimodal corpora, much work is needed in eradicating the current lack of multimodal resources. The authors also make the following observations:

- "There are no systematically collected and annotated image-language corpora for training systems to construct integration resources/multimodal thesauri automatically, or even for facilitating a principled, manual construction of such integration resources that will open the road for reuse across multimodal integration tasks.

- There are no systematically collected and annotated image-language corpora for a principled, safely generalised guidance on media allocation, cross-modal reference resolution/generation and interface design issues in intelligent multimodal systems development.

- There is no gold standard against which the output of multimodal systems involving visual and linguistic modalities could be evaluated/correlated" (Pastra & Wilks 2004b p.2).

Pastra & Wilks (2004b) suggest that focus is needed in both acquiring such multimodal corpora and in finding annotation schemes and tools for performing the annotation. Anvil (Kipp 2001, Martin & Kipp 2002) is one such annotation tool designed for the annotation of videos. Anvil is XML-based and platform independent. It is highly generic in that it can be used with different annotation schemes. Anvil uses an XML-based coding scheme, which is contained in a specification file. According to Martin & Kipp (2002), Anvil can be thought of as a "graphical notepad for parallel events such as speech and gesture" (Martin & Kipp 2002, p.1). The MATE Workbench (Mc Kelvie et al. 2000) is another tool that allows the creation and maintenance of spoken language corpora. Like Anvil, it is implemented in Java, is platform independent and uses an XML file format. The Friedman Osborn Martell (FORM) gesture annotation system (Martell et al. 2002) aims to provide a corpus of speech and its corresponding gestures. The creators of FORM intend to make the corpus available for other researchers to use in their work. FORM is an annotation scheme that describes the kinematical information in a gesture. It is also extensible to allow the description of speech and other conversational information. The envisaged result of the project is to create an extensible corpus of annotated videos that will assist researchers in their understanding of the various aspects of conversational interaction. Martell et al. (2002) describe how the FORM annotation scheme has been implemented using the Anvil tool, as illustrated in Figure 2.26.

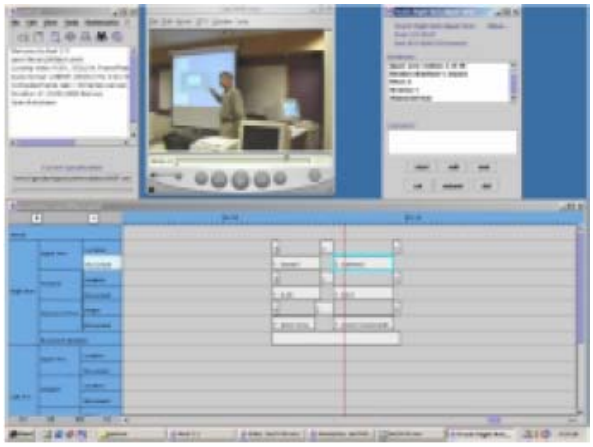

Figure 2.26: FORM annotation implemented using Anvil (Martell et al. 2002)

The Anvil annotation tool is also used to implement the Tycoon framework, as discussed in Martin & Kipp (2002). Tycoon was developed to assist the study and development of multimodal systems, offering an encoding scheme and analysis metrics for multimodal communication scenarios. The Tycoon scheme/metrics are implemented in Anvil for a video sample. The use of Anvil for implementing both the Tycoon framework and the FORM annotation scheme is evidence of its broad capabilities in multimodal annotation.

2.6 Speech Mark-up Language Specifications

The desire for multimodal access to the Internet has led to the development of several technologies specifically designed for this purpose. VoiceXML (2005), XHTML + Voice (Mc Tear 2004), SALT (2005) and EMMA (2005) are all mark-up specifications that can be used to create applications that use voice input and/or voice output. A brief discussion of each of these standards will now follow.

VoiceXML (Voice eXtensible Mark-up Language) (Hansmann et al. 2003, VoiceXML 2005) is a standard that enables access to the web via language and telephone. Developed by AT&T, IBM, Lucent Technologies and Motorola, VoiceXML is essentially a specification language that is an application of XML. A VoiceXML interpreter is used to allow language-based web applications to be developed. The interpreter's role is to execute VoiceXML code and access the speech processing system. VoiceXML has to date been applied in several applications including telephone services, recording of spoken user input, speech recognition and generation, processing of audio files and recognition of dial tone impulses. Over recent years VoiceXML has been developed and refined by industry experts in voice programming, and has been extensively tested in real-world scenarios such as call centres. VoiceXML provides a well-defined standard for combining speech and Internet technology. However, the quality of any voice application strongly depends on the system used for speech processing.

XHTML + Voice (X + V) (XHTML + Voice 2005, IBM X + V 2005, Mc Tear 2004) is a web mark-up language for developing multimodal applications. It is a product of the integration of XHTML (2004) and VoiceXML, with the intention of bringing spoken interaction to the visual web. The X + V specification was proposed to the W3C in December 2001 by IBM, Motorola and Opera Software ASA. IBM and Opera have since developed a multimodal browser based on the proposed specification. X + V is an XML application, with XHTML (eXtensible HyperText Mark-up Language) being the host language. XHTML is an XML-based language that is used for the creation of visual applications. The motivation behind its development is to enable web developers to create increasingly smarter web applications. Using X + V multimodal dialogues can be created that combine the visual input mode, represented by XHTML, and input and output speech, represented by a subset of VoiceXML. One example of where X + V could be applied is in turning a visually oriented web page into a multimodal one. Although X + V is still in the development stage, it's proposers at IBM say that it will "deliver the feature set, flexibility, and ease of use that developers need to write one application that supports visual-only, voice-only, and multimodal interaction" (IBM X + V 2005 p.2). X + V comprises the three core XML standards, XHTML (2005), VoiceXML (Hansmann et al. 2003, VoiceXML 2005) and XML Events (XML Events 2005), to assist in the development of multimodal application interfaces.

The SALT (Speech Application Language Tags) (Mc Tear 2004, SALT 2005) specification allows multimodal and telephony-enabled access to information, applications and Web services from PCs, telephones, tablet PCs and wireless personal digital assistants (PDAs). SALT is an extension of existing mark-up languages such as HTML, XHTML and XML. As declared by the SALT forum, "multimodal access will enable users to interact with an application in a variety of ways: they will be able to input data using speech, a keyboard, keypad, mouse and/or stylus, and produce data as synthesized speech, audio, plain text, motion video, and/or graphics. Each of these modes will be able to be used independently or concurrently" (SALT 2005, p.1). One limitation with SALT, in relation to the previously discussed X + V, is that it does not provide a standard visual mark-up language or eventing model. Instead it provides a low-level set of tags for specifying voice interaction that must be embedded into other environments.

EMMA (Extensible MultiModal Annotation mark-up language) (EMMA 2005) is a specification proposed to the W3C in December 2004. EMMA is a mark-up language for systems that provide semantic interpretations for multimodal inputs. It is hoped that EMMA will be used as a standard data interchange format between the modules of a multimodal system. It will normally be generated automatically by interpretation components to semantically represent user input and will not be directly authored by the developer. A set of elements and attributes is provided that represent annotations on interpretations of the multimodal inputs. EMMA provides a relatively simple syntax for organising interpretations and instances, and an annotative syntax derived from RDF is used to assign annotations to the multimodal input data.

2.7 Artificial Intelligence 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 2.27 provides some clarification by illustrating the spectrum of intelligence based on the level of understanding involved.

Figure 2.27: Spectrum of intelligent behaviour (Hopgood 2003)

Although there has 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. AI technology can be broadly categorised into the following two areas:

- Explicit modelling
- Implicit modelling

With explicit modelling, words and symbols are used to create explicit rules to model the problem, for example:

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. A brief discussion will follow on each of these techniques.

2.7.1 Fuzzy Logic

Fuzzy Logic (Passino & Yurkovich 1997) was first introduced by Professor Lotfi Zadeh in 1969 at the University of California, Berkley. Fuzzy Logic is a problem solving methodology that can be used to arrive at definite conclusions based on vague, imprecise or missing data. This allows 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". Each of these linguistic variables have linguistic values assigned to them. Examples of linguistic values are "very hot", "zero", "positive large", "negative small", "ok", "cold", "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 2.28.

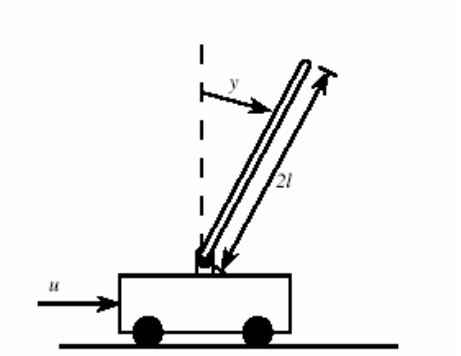

Figure 2.28: Inverted Pendulum (Passino & Yurkovich 1997)

Note that 'u' in Figure 2.28 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:

$$e(t) = r(t) - y(t) \qquad (1)$$

where r(t) is the reference point, indicated by the dashed line in Figure 2.28, 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 abbrieviation 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 2.29. 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) = π/4 then μ(π/4) = 1.0, indicating that we are absolutely certain that e(t) = π/4 is what we mean by “possmall.”

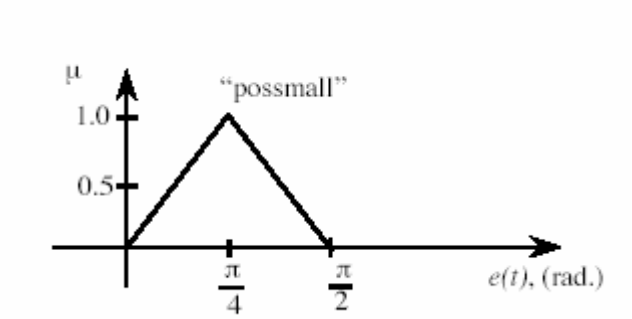

Figure 2.29: Membership function for “possmall” (Passino & Yurkovich 1997)

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

Therefore, using (1):

e(t) = r(0) – y(0) = 0 – 0.1 = -0.1

d/dt e(t) = dy/dt = 0

The membership functions for these input conditions are shown in Figure 2.30. For e(t) = - 0.1 and d/dt e(t) = 0, we can see from Figure 2.30 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

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.

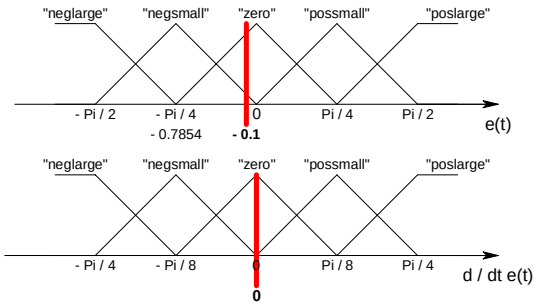

Figure 2.30. Membership function for "error" and "change in error"

It is not necessary to continue this example further. It is suffice to say that 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 language and vision data.

2.7.2 Genetic Algorithms

Genetic Algorithms (GA) (Davis 1991; Goldberg 1989; Holland 1992) are another paradigm of Artificial Intelligence that could be used for decision-making within a multimodal system. GA 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 2.31. Genetic Algorithms were first proposed in 1975 by John Holland at the University of Michigan. They 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. Each 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 optimum solution to the problem in question. So in each generation, new strings are created from the segments of previous strings. New random data is occasionally added in order to keep the population from stagnating. GA are parallel search procedures. That is, they can be implemented on parallel processing machines which can greatly speed up their operation. The main elements of GA 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.

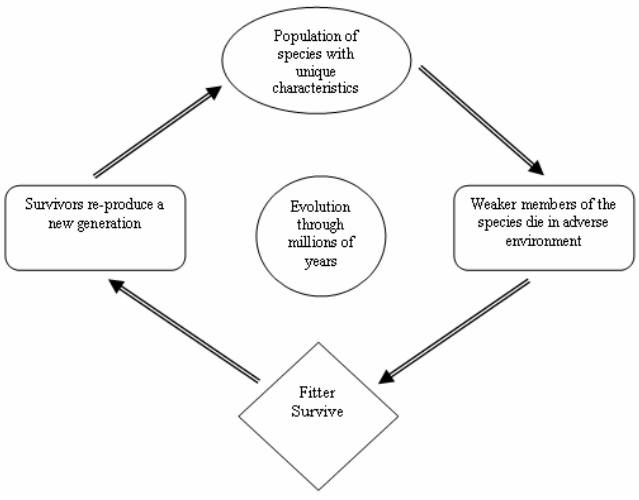

Figure 2.31. Darwin's theory of evolution

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}

There are three basic operators found in every GA. These are selection (reproduction), crossover and mutation. The selection operator allows an individual string to be copied, based on the string's fitness value, for possible inclusion in the next generation. The fitness value is calculated using a fitness function. 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 is used to introduce new genetic material into an existing individual, thus increasing the genetic diversity of the population.

2.7.3 Neural Networks

Neural Networks (Haykin 1999) are information processing paradigms that have been inspired by the way biological nervous systems, such as the human brain, process information. A neural network is made up of a large number of interconnected processing elements (or neurons) that work 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 2.32.

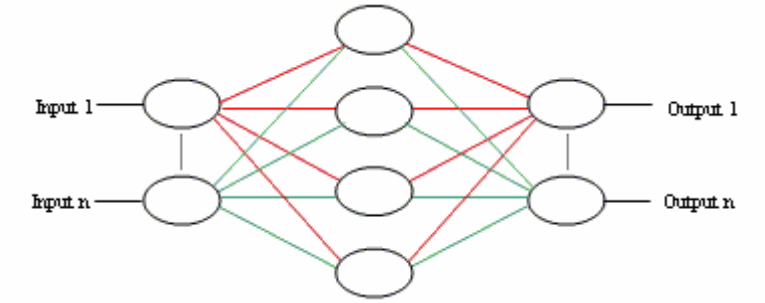

Figure 2.32. Example of a typical neural network structure

Three important properties of Neural Networks are:

- Parallelism
- Learning
- Generalisation

Parallelism allows 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 solve 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 refers to the ability of 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 possible method of Artificial Intelligence for decision-making within multimodal systems.

2.7.4 Bayesian Networks

Bayesian networks (Fenton 2005, Hugin 2005, Jensen 2000, Kadie et al. 2001) are also known as Bayes nets, Causal Probabilistic Networks (CPNs), Bayesian Belief Networks (BBNs) or simply belief networks. A Bayesian Network consists of a set of nodes and a set of directed edges

between the nodes. Each of the nodes represents a random variable, with each edge representing a cause-effect relationship within the domain. 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). Each directed edge models causal impact. As an example consider Figure 2.33, where the directed edges from 'Diet' and 'Exercise' have an impact on 'Weight Loss'.

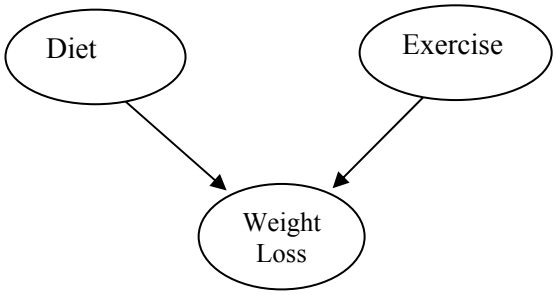

Figure 2.33: Example of a simple Bayesian Network

It can be seen in Figure 2.33 that the nodes are represented by ovals and the directed edges are illustrated by arrow. In Figure 2.33 there is a causal dependency from 'Diet' to 'Weight Loss' and from 'Exercise' to 'Weight Loss'. Each node in a Bayesian Network contains the states of the random variable it represents and a conditional probability table (CPT) or a conditional probability function (CPF). The CPT of a node contains the probabilities of the node being in a specific state given the states of its parents. Note that the effects, represented by the directed edges, are not completely deterministic (e.g. disease -> symptom) and the strength of an effect is modelled as a probability. 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 | represented 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...', there also needs to be a specification of how the two causes combine. That is, one needs the probability of having a fever if both symptoms are present and if the patient is completely healthy. Hence it is necessary to specify the conditional probabilities:

P(temp>37.9 | whooping cough, tonsillitis)

where 'whooping cough' and 'tonsillitis' each can take the states 'yes' and 'no'. Thus it is necessary to specify the strength of all combinations of states for all possible causes. Inference, or model evaluation, means computing the conditional probability for some variables given information (evidence) on other variables. This is easy when all available evidence is on variables that are ancestors (parent nodes) of the variable(s) of interest. But when evidence is available on a descendant of the variable(s) of interest, one has to perform inference against the direction of the edges. To this end, Bayes' Theorem is employed:

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

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.

3. Project Proposal

The proposed project is the design and implementation of MediaHub - an intelligent multimedia distributed platform hub for the fusion and synchronisation of language and vision data. A schematic for MediaHub is shown in Figure 3.1.

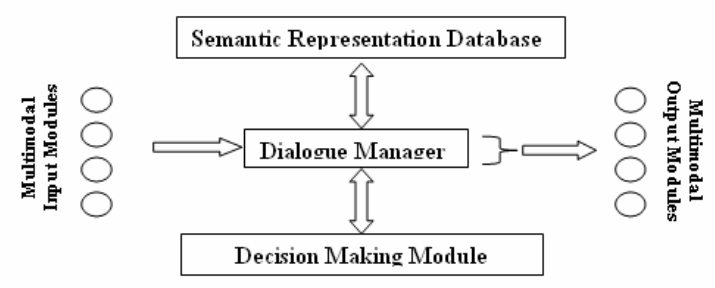

Figure 3.1: Architecture of MediaHub

The key components of MediaHub are:

- Dialogue Manager
- Semantic Representation Database
- Decision-Making Module

The role of the Dialogue Manager is to facilitate the interactions between all components of the platform. It will act as a blackboard module, with all communication between components achieved via the DM. The DM will also be responsible for the synchronisation of the multimodal input and output. During the testing of MediaHub an existing multimodal platform, such as CONFUCIUS (Ma & Mc Kevitt 2003), will be used to perform the processing of the multimodal input and output.

The Semantic Representation Database in MediaHub will use an XML-based method of semantic representation. XML has been chosen due to its widespread use in the area of knowledge and semantic representation in intelligent multimedia. XML's ease of use will allow it to be easily integrated into MediaHub.

The Decision-Making Module will employ an Artificial Intelligence (AI) technique to provide decision-making on language and vision data. Bayesian Networks are currently being investigated to determine if they will be suitable for decision-making. It may also be possible to use other techniques such as Fuzzy Logic, Neural Networks, Genetic Algorithms or a combination of techniques to provide this functionality. The potential for these methods of decision-making under uncertainty will be investigated further before a definitive decision is made on the design of the Decision-Making Module. In regard to multimodal input and output, existing input/output data structures will be assumed.

3.1 Decision-making within MediaHub

Having considered various AI techniques that could be used for decision-making within MediaHub, we will now consider the types of decisions that MediaHub will be required to make. Essentially these can be divided into two main categories:

- Decisions relating to input
- Decisions relating to output

With regard to decisions necessary at the input, these can be further categorised into the following three areas:

- Determining the semantic content of the input.
- Fusing the semantics of the input. That is, fuse the semantics of the language input such as "Who's office is this?" with the visual input (i.e. the pointing).

- Resolving any ambiguity at the input.

An example of ambiguity at the input could be if the user points three times while saying "show me the best possible route from this office to this office". Here, synchronisation (e.g. using timestamps) could be used to determine which two offices the user is referring to. Another example could be in an industrial environment where a control technician could point 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 may be needed to determine which two computers the control technician is referring to. Only two example of ambiguity was given here, though there are many way in which ambiguity could occur. Resolving ambiguity at the input will be a key objective for the decision-making component of MediaHub.

In relation to decisions at the output, synchronisation issues could arise in order to match, for example, a laser movement with a speech output. As is the case in CHAMELEON (Brøndsted et al. 1998, 2001), a statement of the form "This is the best route from Paul's office to Tom's office" may need to be synchronised with the laser output tracing the route between the two offices. 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, while a response to a user's query may be better presented using natural language output. Another example could be when the driver of a car asks an in-car intelligent 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 drivers eyes are pre-occupied on the road), then the system would respond using speech output.

3.2 Comparison with Previous Work

Table A.1 in Appendix A compares MediaHub to the hub of other existing platforms. In the table, platform characteristics are listed, with a tick (√) indicating if the characteristics are present for each of the platform hubs.

As shown in the table, INTERACT (Waibel et al. 1996) uses neural networks, while DARBS (Choy et al. 2004, Nolle et al. 2001) implements a combination of Rule Based, Neural Network and Genetic Algorithm techniques for decision-making. MediaHub will implement a blackboard model of semantic storage and will use an XML-based method of semantic representation. Spoken Image/SONAS, Ymir, CHAMELEON, SmartKom, DARBS, DARPA Galaxy Communicator and Psyclone also use a blackboard model. An XML-based method of semantic representation is also used in SmartKom, EMBASSI, MIAMM and Psyclone. MediaHub will improve on the capabilities of the existing platforms by implementing Bayesian Networks for decision-making.

It can be seen from Table A.1 that each of the systems investigated offer their own advantages in the area of intelligent multimedia. An important characteristic of all of the systems discussed is the ability to facilitate natural language interaction, as this is the most popular method of communication amongst humans. However, as the previous chapter has illustrated, language understanding is the bare minimum that is required in a multimodal system. The main focus of this research is the area of distributed intelligent multimedia platforms and, as the table illustrates, fourteen such platforms have been considered.

From Table A.1, the following observations can be made:

- The capability to facilitate natural language generation and understanding is a fundamental property of a multimedia distributed platform.
- Oxygen, SmartKom and Psyclone were considered the most powerful of the systems reviewed. Psyclone and SmartKom use a blackboard model for semantic storage, whilst

Oxygen implements a non-blackboard model. SmartKom and Psyclone use an XML-based method of semantic storage, whilst Oxygen uses frames.

- None of the systems discussed make use of Bayesian Networks (CPNs) for reasoning under uncertainty.

The last point is particularly relevant, as the potential unique contribution of MediaHub is the use of Bayesian Networks for decision-making in a multimodal distributed platform hub.

3.3 Project Schedule

Table B.1 in Appendix B outlines the plan of work for the completion of this project, together with an indication of when these tasks are expected to be completed. The tasks above the bolded line in Table B.1 have been completed during the previous nine months. We have surveyed research relating to the areas of language and vision integration, multimodal semantic representation, multimodal platforms, distributed processing, multimodal corpora and annotation tools and speech mark-up language specifications. We have also reviewed Artificial Intelligence techniques for decision-making, including Fuzzy Logic, Neural Networks, Genetic Algorithms and Bayesian Networks. The plan for the future development of MediaHub is indicated below the bolded line, with the design and implementation of MediaHub taking focus.

4. Software Analysis and Prospective Tools

Several implementations of XML could be used by the Semantic Representation Database. Initially, XHTML + Voice (XHTML + Voice 2005, IBM X + V 2005) may be a suitable choice, since it combines the vision capabilities of XHTML and the speech capabilities of VoiceXML. Other XML-based languages such as the Synchronised Multimedia Integration Language (SMIL) (Rutledge 2001, Rutledge & Schmitz 2001, SMIL 2005a,b), MMIL, as used in MIAMM (Reithinger et al. 2002, MIAMM 2005) and EMMA (Extensible MultiModal Annotation mark-up language) (EMMA 2005) will also be considered. The HUGIN software tool (HUGIN 2005), a tool for implementing Bayesian Networks as CPNs, will be investigated. An analysis of Hugin is performed in the next section. Other software tools for implementing Fuzzy Logic, Neural Networks and Genetic Algorithms may also be utilised. It is expected that Java will be used as the main programming language within MediaHub, although the decision on which programming language to use will depend on what other processing tools are chosen for use within MediaHub.

4.1 Hugin Graphical User Interface

The Hugin GUI (Graphical User Interface) (HUGIN 2005) is a tool for creating and implementing Bayesian networks and influence diagrams for decision-making. Note that an influence diagram is simply a Bayesian network that has been extended with decision-making facilities (i.e. decision and utility nodes). The GUI provides an easy to use interface to the Hugin decision engine. The Hugin decision engine is available with APIs written in C, C++, Java and as an ActiveX Server for use with Visual Basic. The Hugin tools have been designed to run on the Windows 2000/XP, Solaris, Linux and MAC operating systems. Hugin is also capable of learning 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 4.1.

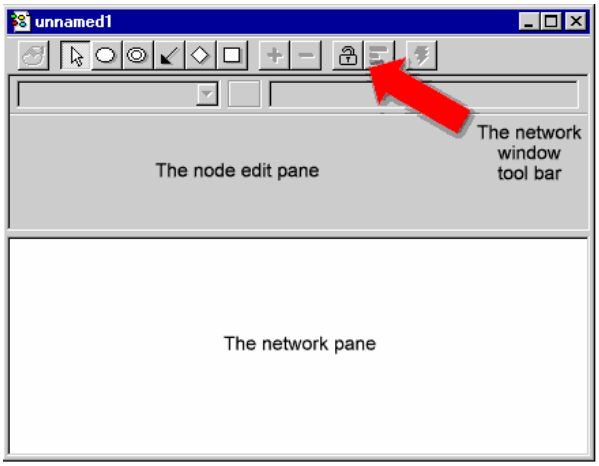

Figure 4.1: Main window of Hugin GUI (HUGIN 2005)

As shown in Figure 4.1, the main window contains a toolbar, a node edit pane and a document pane. The GUI automatically starts up in Edit mode, allowing 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. The Discrete Chance Tool, along with the Node Properties and Link Tool buttons are shown in Figure 4.2. 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 4.3 shows a simple example of a Bayesian network implemented in the Hugin software tool.

Figure 4.2: Discrete Chance Tool, node properties and link tool (HUGIN 2005)

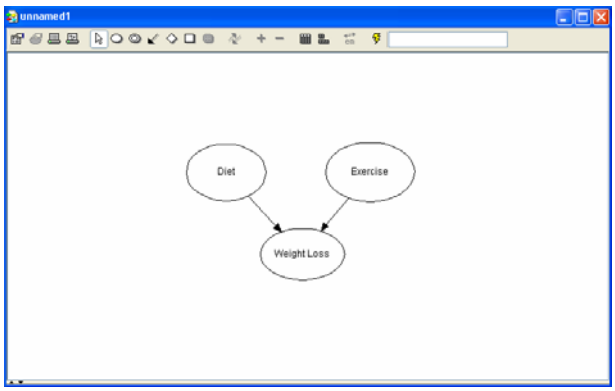

Figure 4.3: Simple example of a Bayesian network in Hugin

Note from Figure 4.3 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. 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 Tables-pane button. To specify the states of 'Diet', we must do the following:

- Select the node 'Diet' by left clicking inside the node. This causes the table for 'Diet' to appear in the tables pane, as shown in Figure 4.4.
- We can now rename the 'State 1' and 'State 2' nodes 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 a value of 1.

The previous procedure would then be repeated for the 'Exercise' and 'Weight Loss' nodes.

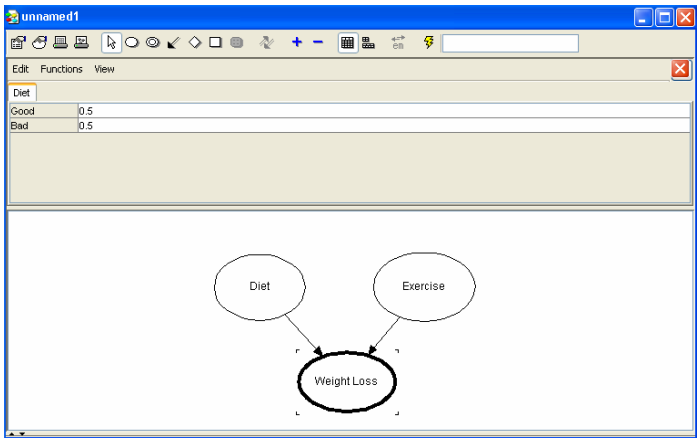

Figure 4.4: View of table for 'Diet' node

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 assigned appropriate values to the states of 'Weight Loss' as illustrated in Figure 4.5.

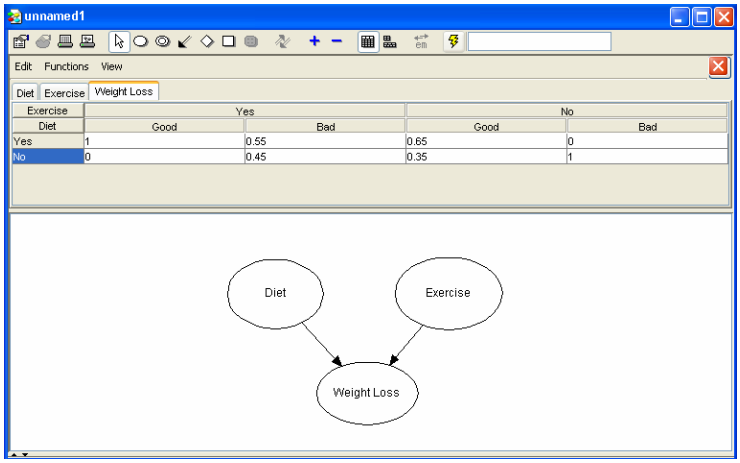

Figure 4.5: CPT of 'Weight Loss' node

Note from Figure 4.5, that the CPT of 'Weight Loss' is larger since it accounts for the parent nodes of 'Diet' and 'Exercise'. Pressing the 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 4.6 shows the weight loss example in Run Mode.

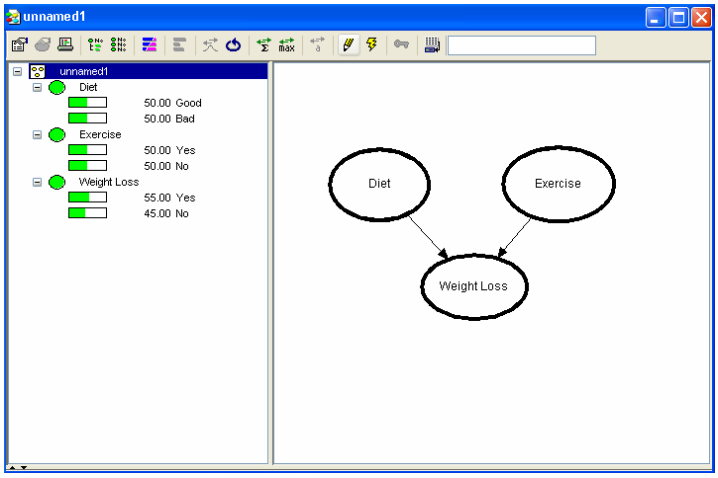

Figure 4.6: Example of Run Mode

We will 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 allows 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. As shown in Figure 4.7 a red coloured letter 'e' appears beside the 'Diet' node in the Bayesian network, indicating that evidence has been added to the node.

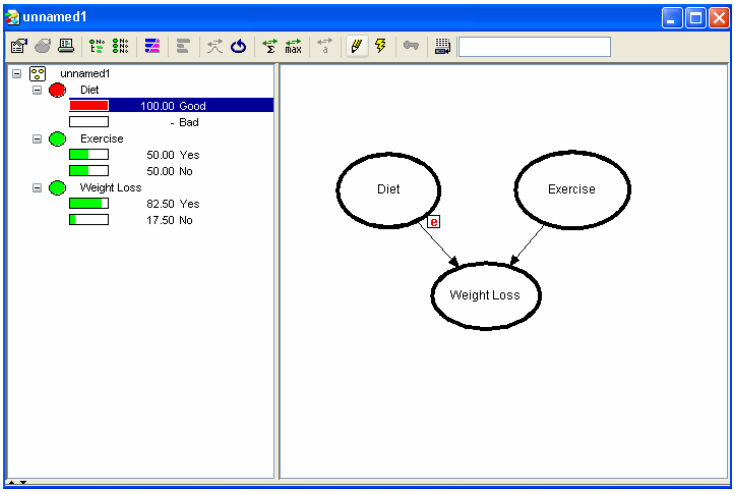

Figure 4.7: Evidence added to the Diet node

See from Figure 4.7 that 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 the diet is accompanied by exercise, the probability of weight loss changes to 100% as illustrated in Figure 4.8.

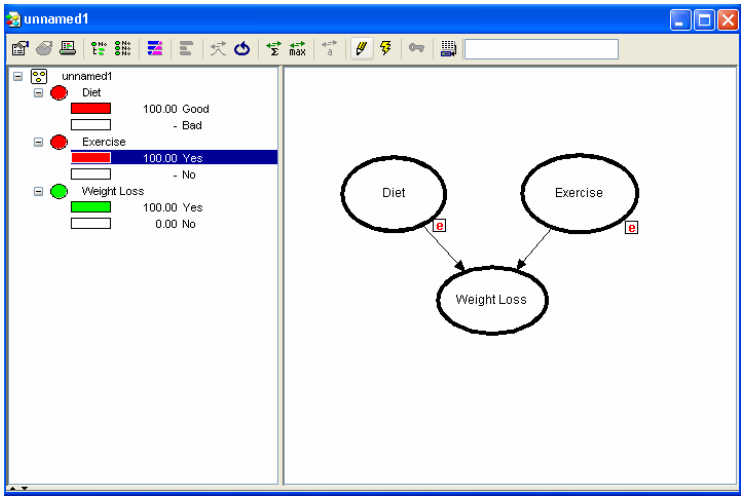

Figure 4.8: Evidence of diet and exercise

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 4.9, where the probability of weight loss has become 27.5%. If we now assert the belief that there is both a bad diet and no exercise, then the probability of weight loss changes to 0%, as shown in Figure 4.10. Therefore we are absolute certainty that a bad diet and no exercise will not result in any weight loss.

Thus the Hugin GUI allows us to create 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' node. When our Bayesian network has been constructed, we can enter evidence to the model and view the results dynamically. Hugin takes care of all the mathematics and computes the new probabilities whenever new information has been added.

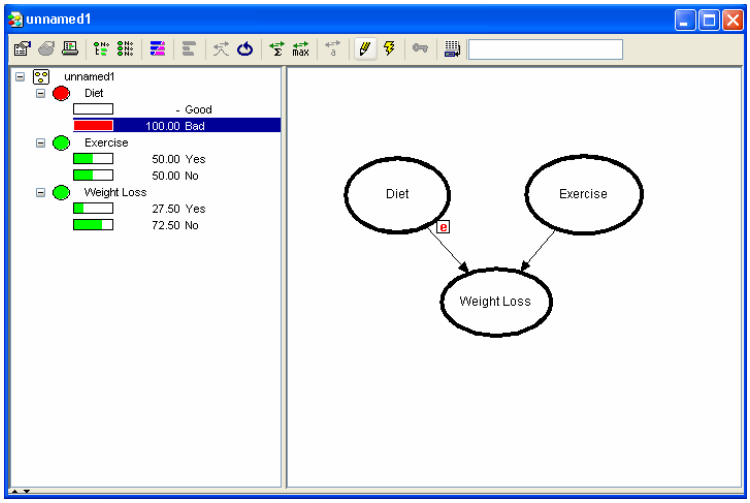

Figure 4.9: Evidence of a bad diet added

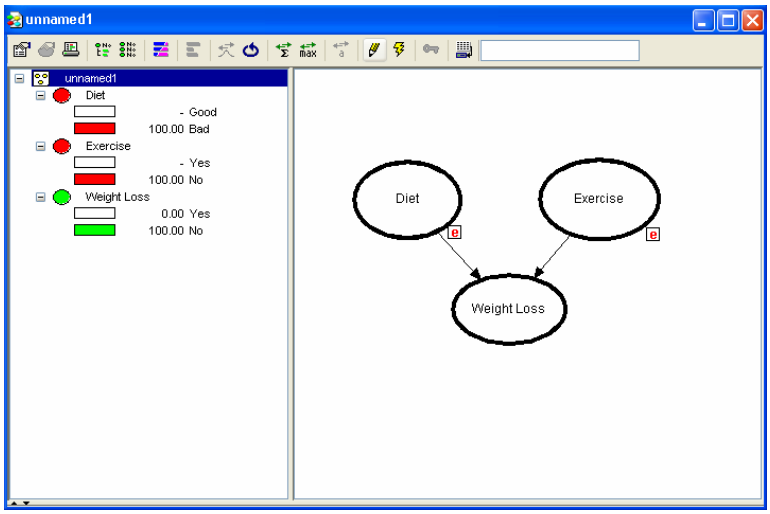

Figure 4.10: Evidence of bad diet and no exercise added

4.2 Hugin API

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, allowing 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. To use the Hugin API in C++, the hugin header file must be included as follows:

include <hugin>

The HAPI namespace defines all of the C++ entities. To access these entities we can use the HAPI : : prefix or place the following line of code before the C++ entities are used:

Using namespace HAPI;

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 in 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).

Within the Hugin API, all objects are represented as opaque pointers. An opaque pointer points to data that is not further defined. Opaque pointers allow the references of data to be manipulated, without having knowledge of the structure of the actual data. The Hugin API allows several data types including integer, string and boolean. Nodes and domains are represented in the Hugin API using the opaque pointer types h_node_t and h_domain_t. The following code is used to create an empty domain:

h_domain_t h_new_domain (void)

Similarly, the following code is used to create a new node of category C and kind K within the domain D:

h_node_t h_domain_new_node (h_domain_t D, h_node_category_t C, h_node_kind_t K)

The following function is used to add a directed edge between a parent and child node:

h_status_t h_node_add_parent (h_node_t child, h_node_t parent)

When the above function is executed, the conditional probability table associated with the child node will be automatically updated. Functions are also provided for adding parents to a node, removing parents from a node, replacing one parent with another parent and reversing a directed edge between two nodes. There are also functions for numerous other operations such as retrieving the current parents and children of a node and specifying the number of states that a node will have.

4.3. Other Bayesian decision-making tools

In addition to Hugin, other potential tools for Bayesian decision-making are:

- MSBNx/MSBN3 (Kadie et al. 2001, MSBNx 2005)
- GeNIe/SMILE (Genie 2005)
- Netica (Norsys 2005)
- Bayes Net Toolbox (Murphy 2005)
- BUGS (BUGS 2005)

MSBNx (Kadie et al. 2001, MSBNx 2005) is a Windows application for creating and evaluating Bayesian networks. It is available for non-commercial use and can be downloaded at MSBNx (2005). Components of this Microsoft application can be integrated into programs, allowing 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. 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. The MSBN3 API manual is available online at MSBN3 (2005).

The GeNIe (Graphical Network Interface) (Genie 2005) software package is the graphical user interface to SMILE (Structural Modelling, Inference, and Learning Engine). It has been developed at the University of Pittsburgh. GeNIe supports all major file types, including Hugin and Nitica. GeNIe is implemented in Visual C++ and only runs under the Windows family of operating systems (98, NT, 2000, XP). SMILE, however, 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.

The Netica application (Norsys 2005) allows problem solving using Bayesian networks and influence diagrams. A free demo version of Netica can be downloaded at Norsys (2005). The demo version has full functionality, but is limited in model size. The complete application allows the building, editing and learning of Bayesian networks and influence diagrams. Netica compiles belief networks into junction trees of cliques to allow fast probabilistic reasoning. Netica also offers the Netica Programmers Library, or the Netica API, to allow programmers to embed the functionality of Netica within their own applications. The API is written in C and can be used with any programming language, provided the language is capable of calling C functions. The Netica API only requires the Standard C library to operate, but can be used in conjunction with any other C or C++ library. Versions of the API are available for Windows, Sun Sparc and Macintosh. The interface for each of these operating systems is identical, so code is fully portable across all of the platforms.

The Bayes Net Toolbox (BNT) (Murphy 2005) 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, which is not free. 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 big strengths 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.

The BUGS (Bayesian inference Using Gibbs Sampling) (BUGS 2005) project is concerned with providing a software tool that can perform Bayesian analysis of complex statistical models using Markov Chain Monte Carlo (MCMC) Methods (Neal 1993). Since the project began in 1989, several versions of BUGS have been developed. 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.

4.4 Comparison of Bayesian decision-making tools

Table C.1 in Appendix C provides a comparison of a few existing tools for Bayesian decision-making. As shown in table C.1, all of the applications discussed provide an API with the exception of BUGS. In the case of the Bayes Net Toolbox (BNT), no API is required since the source code is made available to the developer. It is however necessary to first have Matlab in order to use the BNT. Another disadvantage associated with the BNT is the fact that no Graphical User Interface (GUI) is provided to assist development. As shown in the table, of those applications considered, only BNT and Hugin boast the capability of learning structures. However, this would only be an advantage if there was a need for structures to be learned. Continuous random values are supported by all the applications with the exception of GeNIe/SMILE and MSBNx. GeNIe/SMILE does have the advantage, along with BNT and BUGS, of being entirely free and unrestricted. Hugin, MSBNx and Netica are also free but they are restricted in some way (e.g. in the size of models that can be created). This only becomes an issue if large models are to be created. It is possible to create fairly useful Bayesian models within these systems, without having to pay for the restrictions to be removed. Whilst Hugin seems like a good choice for the development of MediaHub, Microsoft's MSBNx is also a viable option – particularly if the .NET framework is to be used. It should be noted that if .NET is to be utilised, then Hugin could still be used if it offers greater advantages than its Microsoft counterpart. The decision on which Bayesian tool to use will be finalised during the design of MediaHub. -making.

5. Conclusion

Firstly, this chapter summarises the research proposal, the literature review and the main contribution of this project. Next, a software analysis focusing on potential tools and programming languages for possible implementation is discussed. Finally, thoughts on the future research and development of MediaHub are provided.

The objective of the work described in this research report is the proposed development of MediaHub, an intelligent multimedia distributed platform hub for the fusion and synchronisation of language and vision data. The primary objectives of MediaHub are to (1) interpret/generate semantic representations of multimodal input/output and (2) perform fusion and synchronisation of multimodal data (decision-making). In meeting these objectives, four key research issues in the area of intelligent multimedia distributed processing will be addressed: (1) multimodal semantic representation, (2) communication within a multimodal platform, (3) semantic storage and (4) decision-making within a multimodal platform hub. The proposed architecture and core modules of MediaHub have also been presented.

Research related to the design and implementation of MediaHub has been reviewed. This has included an investigation into the area of language and vision integration. The area of multimodal semantic representation has also been considered, with focus being placed on both the frame-based and XML-based methods of semantic representation. An analysis of several existing multimodal platforms has been performed, with particular attention given to their method of communication, semantic representation, semantic storage and decision-making. Multimodal corpora and annotation tools that are of potential use in the development of MediaHub were discussed. Also included in this report is a review of several speech mark-up language specifications and a discussion on various Artificial Intelligence techniques, in particular Bayesian Networks, which allow reasoning and decision-making under uncertainty.

A software analysis has been performed, focusing on potential tools for use within MediaHub. This has included an analysis of several implementations of XML that can be used for semantic representation and consideration into the choice of programming language, which will depend heavily on the final decision on which tools to use within MediaHub. Several existing applications for Bayesian decision-making have been analysed, with a view to their potential employment within MediaHub. The Hugin GUI/API (HUGIN 2005) has been identified as the most promising approach for implementing Bayesian decision-making within MediaHub.

A unique contribution of MediaHub has been identified detailing how MediaHub, using Bayesian Networks for decision-making, will improve on current multimodal systems. A project plan has been presented which details the remaining tasks necessary for the implementation of MediaHub.

Future development involves developing the Bayesian decision-making using the Hugin API, developing the semantic representation database and acquiring multimodal corpora for the testing of MediaHub. When implemented, MediaHub will be tested within an existing multimodal platform such as CONFUCIUS (Ma & Mc Kevitt 2003).

In conclusion, this report has presented a summary of the motivation for, and future direction of, the development of MediaHub, a multimodal distributed platform hub for the fusion and synchronisation of language and vision data.

6. References

Amtrup, Jan W. (1995) ICE-INTARC Communication Environment Users Guide and Reference Manual Version 1.4, University of Hamburg, October.

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, Scientific American, May 17.

Blattner, M.M. & E.P. Glinert (1996) Multimodal Integration. In Multimedia, IEEE, Volume 3, Issue 3, 14-24.

Bolt, R.A. (1980) "Put-that-there" Voice and gesture at the graphics interface. Computer Graphics (SIGGRAPH '80 Proceedings), 14(3), July, 262–270.

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., 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 and Robbert-Jan Beun (Eds.), 217-233. Lecture Notes in Artificial Intelligence (LNAI) series, LNAI 2155, Berlin, Germany: Springer Verlag.

BUGS (2005)
http://www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml Site visited 10/03/05.

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 UP, Cambridge, 1992.

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 18th European Simulation Multiconference (ESM2004), Magdeburg, Germany, June 2004, Horton, G., (Ed.), 129-135.

CORBA (2005)
http://java.sun.com/developer/onlineTraining/corba/corba.html Site visited 01/02/05.

DAML (2005)
http://www.daml.org/ Site visited 15/02/05.

DAML-S (2005)
http://www.daml.org/services/owl-s/ Site visited 15/02/05.

DAML+OIL (2005)
http://www.daml.org/2001/03/daml+oil-index Site visited 15/02/05.

Davis, L. (Ed.) (1991) Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

EMBASSI (2005) http://www.embassi.de/ewas/ewas_frame.html Site visited 09/04/05.

EMMA (2005) W3C Working Draft.
http://www.w3.org/TR/2004/WD-emma-20041214/
Site visited 21/01/05.

Fenton (2005) Norman Fenton's web resource.
http://www.dcs.qmw.ac.uk/~norman/BBNs/BBNs.htm Site visited 04/03/05.

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.

Freeman (2005) Make Room For JavaSpaces Part 1.
http://www.javaworld.com/javaworld/jw-11-1999/jw-11-jiniology.html Site visited 16/02/05.

Fürntratt, H., H. Neuschmied & W. Bailer (2004) MPEG-7 library: MPEG-7 C++ API Implementation, Institute of Information Systems & Information Management, Joanneum Research, Austria, May.

Genie (2005)
http://www.sis.pitt.edu/~genie/ Site visited 07/03/05.

Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimisation and Machine Learning. Addison-Wesley.

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, Chapter 20.

Gruber, T.R. (1993) A translation approach to portable ontology specifications. In Knowledge Specification, vol. 5, 199-220.

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.

Hansmann, U., L. Merk, M.S. Nicklous & T. Stober (2003) Pervasive Computing Second Edition, Springer, Berlin, Germany.

Haykin, S. (1999) Neural Networks, A Comprehensive Foundation. Prentice Hall, Upper Saddle River, NJ.

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. 267, No. 1, July, 66.

Hopgood, A.A. (2003) Artificial Intelligence: Hype or Reality? In Computer, Vol. 36, No. 5, IEEE Computer Society, 25-28, May.

Hugin (2005) Hugin Expert Developers Site. http://developer.hugin.com/ Site visited 30/01/05.

IBM X + V (2005)
IBM's proposal to the W3C.
http://www-106.ibm.com/developerworks/wireless/library/wi-xvlanguage/ Site visited 03/04/05

Jensen, F.V. (2000) Bayesian Graphical Models, Encyclopaedia of Environmetrics, Wiley, Sussex, UK.

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. (1998) Unification-based multimodal parsing. In Proceedings of the 36th conference on Association for Computational Linguistics, Montreal, Quebec, Canada, 624-630.

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.

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.

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.
Klein, M. (2001) XML, RDF, and relatives. In Intelligent Systems, IEEE, Volume 16, Issue 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.

Kristensen, T., T Software Agents In A Collaborative Learning Environment. In International Conference on Engineering Education, Oslo, Norway, August 2001, Session 8B1, 20-25.

LREC (2005) http://www.lrec-conf.org/lrec2002/ Site visited 05/02/05.

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.

Martell, C., C. Osborn, J. Friedman & P. Howard (2002) The FORM Gesture Annotation System. In Third International Conference on Language Resources and Evaluation (LREC): Multimodal Resources and Multimodal Systems Evaluation Workshop, June 1, Las Palmas, Canary Islands, Spain, 16-23.

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.

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 Kelvie, D., A. Isard, A. Mengel, M. B. Moller, M. Gross & M. Klein (2000) The MATE workbench - an annotation tool for XML coded speech corpora. In Speech Communication, Vol. 33, No. 1-2, 97-112.

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. (2003) MultiModal semantic representation. In Proceedings of the SIGSEM Working Group on the Representation of MultiModal Semantic Information, First Working Meeting, Fifth International Workshop on Computational Semantics (IWCS-5), Harry Bunt, Kiyong Lee, Laurent Romary, and Emiel Krahmer (Eds.), Tilburg University, Tilburg, The Netherlands, January, 1-16.

Mc Tear, M.F. (2004) Spoken dialogue technology: toward the conversational user interface. Springer Verlag: London.

MIAMM (2005) Multidimensional Information Access using Multiple Modalities. http://miamm.loria.fr/ Site visited 09/02/05.

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 Library (2005)
http://iis.joanneum.at/MPEG%2D7/ Site visited 31/01/05.

MPEG7 (2005)
http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm Site visited 04/04/05.

MSBNx (2005)
http://www.research.microsoft.com/adapt/MSBNx/ Site visited 02/03/05.

MSBN3 (2005)
http://research.microsoft.com/adapt/MSBNx/MSBN3Manual.asp?path=msbn3/main.htm Site visited 16/03/05.

MS.NET (2005)
http://www.microsoft.com/Net/Basics.aspx Site visited 19/04/05.

Murphy (2005)
Website of Kevin Patrick Murphy. http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html Site visited 18/04/05.

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.

Nolle, L., K. Wong & A.A. Hopgood (2001) DARBS: a distributed blackboard system. In Proc. ES2001, Research and Development in Intelligent Systems XVIII, M.Bramer, F. Coenen and A. Preece (Eds.), 161-170, Berlin, Germany: Springer-Verlag.

Norsys (2005)
Norsys Software Corp. http://www.norsys.com/ Site visited 07/03/05.

OAA (2005) http://www.ai.sri.com/~oaa/whitepaper.html Site visited 18/01/05.

Okada, N. (1996) Integrating vision, motion, and language through mind. In Integration of Natural Language and Vision Processing (Volum 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, 1999. Proceedings, September, 49-77.

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.

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.

OWL (2005)
W3C OWL overview. http://www.w3.org/2004/OWL/ Site visited 04/05/05.

Oxygen (2005) http://oxygen.lcs.mit.edu/Overview.html Site visited 22/03/05.

Passino, K.M. & S. Yurkovich (1997) Fuzzy Control. Menlo Park, CA: Addison Wesley Longman.

Pastra, K. & Y. Wilks (2004a) Vision-Language Integration in AI: a reality check. In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), Valencia, Spain, 937-941.

Pastra, K. & Y. Wilks (2004b) 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.

Psyclone (2005) http://www.cmlabs.com/psyclone/ Site visited 21/03/05.

RDF Schema (2005) http://www.w3.org/TR/rdf-schema/ Site visited 04/02/05.

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.

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.

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.

SALT (2005) http://saltforum.org Sites visited 01/02/2005.

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 (2005) http://www.smartkom.org Site visited 12/02/05.

SMIL (2005a)
SMIL 1.0 WWW Consortium Recommendation, June 1998. http://www.w3.org/TR/REC-smil/
Site visited 02/03/05.

SMIL (2005b)
http://www.w3.org/AudioVideo/ Site visited 02/03/05.

Sunderam, V.S. (1990) PVM: a framework for parallel distributed computing. In Concurrency Practice and Experience, 2(4), 315-340.

SW (2005)
Semantic Web. http://www.w3.org/2001/sw/ Site visited 04/02/05.

Thórisson, K. (1999) A Mind Model for Multimodal Communicative Creatures & Humanoids. In International Journal of Applied Artificial Intelligence, Vol. 13 (4-5), 449-486.

TopXML (2005) http://www.topxml.com/xml/default.asp Site visited 23/02/05.

Vinoski, S. (1993) Distributed object computing with CORBA, C++ Report, Vol. 5, No. 6, July/August, 32-38.

VoiceXML (2005)
http://www.voicexml.org Site visited 30/03/05.

W3C (2005)
W3C homepage. http://www.w3.org Site visited 01/05/05.

W3C XML (2005)
W3C XML Activity Statement. http://www.w3.org/XML/Activity.html Site visited 29/03/2005.

XHTML (2005)
XHTML™ 2.0. http://www.w3.org/TR/xhtml2 Site visited 03/04/2005.

XHTML + Voice (2005)
XHTML+Voice Profile 1.2. http://www.voicexml.org/specs/multimodal/x+v/12/ Site visited 03/03/05.

Wahlster, W. (2003) SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell. In: Krahl, R., Günther, D. (eds), 47-62, Proceedings of the Human Computer Interaction Status Conference, June. Berlin, Germany: DLR.

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.

Waibel, A., M.T. Vo, P. Duchnowski & S. Manke (1996) Multimodal Interfaces. In Artificial Intelligence Review, Vol. 10, Issue 3-4, August, 299-319.

XML Events (2005)
An Events Syntax for XML. http://www.w3.org/TR/xml-events/ Site visited 09/03/05.

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.

Appendix A: Comparison of Intelligent Multimodal Platforms

Category	System	Year	Semantic Representation		Semantic Storage		Decision-making (Fusion and Synchronisation)					Multimodal Interaction							
												Input Media				Output Media			
			Frames	XML	BB	Non BB	Bayesian Networks (CPNs)	Fuzzy Logic	Genetic Algorithms	Neural Networks	Rule Based	Text	Speech	Gesture	Vision	Text	Speech	Gesture	Graphics
Intelligent Multimodal Platforms	WAXHOLM	1992	√			√					√		√			√	√	√	√
	Spoken Image / SONAS	1994	√		√						√	√	√	√					√
	AESOPWORLD	1996	√			√					√		√		√		√		√
	COLLAGEN	1996	√			√	N/A					√	√			√	√	√	
	INTERACT	1996	√			√				√		√	√	√	√				
	Ymir	1997	√		√						√		√	√	√		√	√	√
	CHAMELEON	1998	√		√						√	√	√	√	√		√		
	Oxygen	1999	√			√					√		√	√	√	√	√	√	√
	SmartKom	2000		√	√						√	√	√	√	√	√	√		√
	DARBS	2001	√		√				√	√	√	√	√			√	√		
	DARPA Galaxy Communicator	2001	√		√						√		√				√		
	EMBASSI	2001		√		√					√		√	√	√		√	√	
	MIAMM	2001		√		√					√		√	√	√	√	√		√
	Psyclone	2003		√	√						√	√	√		√	√	√		√
This Project	**MediaHub**			√	√		√					√	√	√	√	√	√	√	√

Table A.1: Comparison of Intelligent Multimodal Platforms

Appendix B Project schedule

Research Activities	Year 1			Year 2			Year 3		
	Oct '04 - Jan 05	*Feb '05 - May '05*	*June '05 - Sept '05*	*Oct '05 - Jan '06*	*Feb '06 - May '06*	*June '06 - Sept '06*	*Oct '06 - Jan '07*	*Feb '07 - May '07*	*June '07 - Sept '07*
Literature survey									
Writing Chapter 2 'Literature Review'									
Analysis and selection of tools									
Analysis of Semantic Representation Languages and speech mark-up languages (e.g., XML, SMIL, X + V)									
Analysis of suitable programming languages (e.g. Java, C, C++)									
Review of AI techniques & other reusable system components (e.g. FL, NNs, GAs, Bayesian Networks)									
Design of MediaHub									
Architecture implementation									
Develop Semantic Representation Database									
Implement AI technique for decision-making (e.g. FL, NNs, GAs, Bayesian Networks e.g. CPNs)									
Develop a prototype application for MediaHub									
Integration and testing									
Improving system									
Write up PhD thesis									

Table B.1: Project Schedule

Appendix C Summary Table of Bayesian decision making tools

Name	Source Code	API	GUI	Continuous Random Values	Can Learn Structures	Free
Bayes Net Toolbox	Matlab	Not Needed[1]	No	Yes	Yes	Yes
GeNIe/SMILE	No	Yes	Yes	No	No	Yes
Hugin	No	Yes	Yes	Yes	Yes	Restricted
MSBNx	No	Yes	Yes	No	No	Restricted
Netica	No	Yes	Yes	Yes	No	Restricted
BUGS	No	No	Yes	Yes	No	Yes

[1] API is not needed since the source code is available

Table C1: Summary of Bayesian decision making tools