Adaptive Ontology-Driven Personalised News Services

Shane Tallon

A dissertation submitted to the University of Dublin,
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science

2005

Declaration

I declare that the work described in this dissertation is,
except where otherwise stated, entirely my own work
and has not been submitted as an exercise for a degree
at this or any other university.

Signed: ____________________

Shane Tallon
September 2005

Permission to lend and/or copy

I agree that Trinity College Library may lend
or copy this dissertation upon request.

Signed: ______________________

Shane Tallon
September 2005

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor Dr. Owen Conlan, whose expert advice, insight and guidance proved invaluable in the writing of this Thesis. I would also like to thank Ian O'Keeffe whose technical expertise was of great help. Thank you also to the members of the Knowledge and Data Engineering Group in Trinity College, some of whom were guinea pigs for the personalised news service, and others who provided even more insight and advice.

I would like to thank my Mam and Dad, Lorraine and Andy, for their constant love, encouragement and support through the year, and thanks to Dad for proof reading my terrible English. Thanks also go to my brother Barry, my two sisters, Aoibhínn and Elenore, and to my Grandparents for providing me with all of those lunches and dinners, and their prayers.

I would like to thank the NDS class, for making the year a very enjoyable experience, and to my friends from home, who made sure I never missed a night out regardless of what work had to be done.

Finally, I would like to dedicate this thesis to the memory of my good friend and Granddad, Dan.

ABSTRACT

Ontologically Driven Adaptation is a challenging field within the area of Personalised News. Ontologies provide a structured, semantically rich methodology for the modelling of a domain. Personalised News is an area of Adaptive Hypermedia which is only beginning to take shape. There is a broad range of news that can be personalised, and a variety of ways in which a Personalised News System can be developed. Adaptive Hypermedia can be leveraged to provide a Personalised News Service for users who have interests in particular domains. Based on the user model, the domain, and the system itself, the news delivered can be high level general news, or low level detailed news, depending primarily on which type a user wishes to receive.

This Thesis looks at the tangible differences between the results provided by two differently modelled domain models, which represent the same information space. One domain model is a structured view of the information space, and it contains rich semantic meaning. By this it is meant that it has rich semantic knowledge about the concepts residing within the domain, and relationships between those concepts. This domain model is expressed as an ontology and will be known from here on as a Strict Ontology. The second domain model, is a taxonomy of the information space with little or no semantic detail of the information space. It knows nothing of the relationships between objects other than the fact that they are related. This domain model will from here on be known as a Loose Ontology.

This Thesis will evaluate the relative benefits of the two different approaches to representing an information space by examining the personalised user experiences offered. This will be carried out against the backdrop of an innovative approach to offering personalised ontologically-driven news.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS .... **III**

ABSTRACT .... **IV**

TABLE OF CONTENTS .... **V**

TABLE OF FIGURES .... **VIII**

1 INTRODUCTION **1**

1.1 MOTIVATION 1
1.2 RESEARCH QUESTION 2
1.3 OBJECTIVES AND GOALS 2
1.4 CONTRIBUTION TO STATE OF THE ART 3
1.5 OVERVIEW 3

2 STATE OF THE ART **5**

2.1 ADAPTIVE HYPERMEDIA 5
2.2 SURVEY OF ADAPTIVE HYPERMEDIA SYSTEMS 7
2.3 ANALYSIS OF ADAPTIVE HYPERMEDIA SYSTEMS 10
2.4 PERSONALISED ELEARNING 12
2.4.1 Adaptive Authoring 14
2.5 ELEARNING ANALYSIS 15
2.6 SURVEY OF PERSONALISED NEWS 15
2.7 ANALYSIS OF PERSONALISED NEWS 18
2.8 ARTIFICIAL INTELLIGENCE METHODS FOR PERSONALISED NEWS 20
2.9 AI METHODS ANALYSIS 22
2.10 THE SEMANTIC WEB 22
2.11 ONTOLOGICALLY DRIVEN ADAPTATION 23
2.12 TECHNOLOGIES 24
2.12.1 XML 24
2.12.2 RSS 24
2.12.3 RDF 24
2.12.4 Ontology Structure (OWL/DAML+OIL) 24
2.12.5 Protégé 25
2.12.6 JESS 25
2.12.7 JATHA 25
2.12.8 LISP 26
2.13 SUMMARY 26

3 DESIGN **27**

3.1 INTRODUCTION 27

3.2 Influences from State of the Art 27
3.2.1 Adaptive Axes 27
3.2.2 User Model 28
3.2.3 Ontologies 30
3.2.4 Choice of System 31
3.3 Domain of News 32
3.3.1 Modelling of the Domain 33
3.4 Framework Design 34
3. 5 Service Design 36
3.5.1 Personalisation 36
3.5.2 Ontology 38
3.5.3 News Feed Capture and Personalisation 41
3.6 Modelling the User 41
3.6.1 Separating Interests from Level of Interest 41
3.6.2 Dynamism of User Modelling 42
3.7 Summary 42

4 IMPLEMENTATION 43

4.1 Introduction 43
4.2 Modelling the Domain 43
4.2.1 Protégé 46
4.3 Framework Implementation 46
4.3.1 RSS News Reader 46
4.3.2 Personalised News Service 46
4.3.3 Ontology Reasoning Web Service 47
4.3.4 News Feed Web Service 48
4.4 Service Implementation 49
4.4.1 Personalisation 50
4.4.2 Ontology Reasoning 54
4.4.3 News Feed Capture and Personalisation 55
4.5 Technological Architecture 55
4.6 Summary 56

5 TRIAL AND EVALUATION 58

5.1 Introduction 58
5.2 User Satisfaction 58
5.2.1 Trial Setup 60
5.2.2 Overview 60
5.2.3 Use of News Services 61
5.2.4 Comfort with using News Services on the Web 62
5.2.5 Comparison of Results with Interests Questionnaire 62

5.2.6 Users satisfaction with the Service *65*

5.2.7 Comparison of Results with Expected Results *68*

5.2.8 Satisfaction with the User Model *68*

5.2.9 Comparison of two Ontologies *69*

5.2.10 Additional Features which could be added *70*

5.2.11 Key Findings *71*

5.3 SERVICE ORIENTED ARCHITECTURE 72

5.4.1 Personalised News Service *72*

5.4.2 Ontology Reasoning Web Service *73*

5.4.3 News Feed Capture and Personalisation Web Service *73*

6 CONCLUSION 75

6.1 INTRODUCTION 75

6.2 OBJECTIVES AND ACHIEVEMENTS 75

6.3 CONTRIBUTION TO STATE OF THE ART 77

6.4 FUTURE WORK 78

6.4.1 Different Sporting Domains *78*

6.4.2 More Diverse Domains *78*

6.4.3 Ontologies *79*

6.4.4 Adaptation Mechanisms *79*

6.4.5 User Model Control *79*

BIBLIOGRAPHY 80

APPENDICES 84

APPENDIX I – JSP PAGES 84

APPENDIX II – NARRATIVE MODELS 86

APPENDIX III – CONCEPTUAL MODEL (LEVEL OF INTEREST MODEL) 90

APPENDIX IV – USER MODELS 93

APPENDIX V – INTEREST QUESTIONNAIRE 94

APPENDIX VI – EVALUATION QUESTIONNAIRE 96

APPENDIX VII –SCREEN SHOTS OF FEED READER USING THE PNS 98

TABLE OF FIGURES

FIGURE 1 – AHA! ARCHITECTURE 8
FIGURE 2 – ADAPTIVE ENGINE 3 ARCHITECTURE 9
FIGURE 3 – PERSONALISED NEWS SERVICE ARCHITECTURE 34
FIGURE 4 – ONTOLOGY CLASSES AND RELATIONSHIPS 40
FIGURE 5 – NARRATIVE STEP BY STEP FLOWCHART 47
FIGURE 6 – ONTOLOGY REASONING WEB SERVICE FLOWCHART 48
FIGURE 7 – NEWS FEED CAPTURE WEB SERVICE FLOWCHART 49
FIGURE 8 – TECHNOLOGICAL ARCHITECTURE 55
FIGURE 9 – FRAMEWORK IMPLEMENTATION 57
FIGURE 10 – USERS PRIOR EXPERIENCE WITH ONLINE NEWS SERVICES 61
FIGURE 11 – COMFORT USING NEWS SERVICES ON THE WEB 62
FIGURE 12 – DID THE NEWS GENERATED REFLECT QUESTIONNAIRE FOR STRICT ONTOLOGY? 63
FIGURE 13 – DID THE NEWS GENERATED REFLECT QUESTIONNAIRE FOR LOOSE ONTOLOGY? 64
FIGURE 14 – DID THE NEWS GENERATED REFLECT NEWS YOU FOUND INTERESTING? 65
FIGURE 15 – DID THE NEWS GENERATED OMIT NEWS YOU WANTED? 66
FIGURE 16 – DID THE NEWS GENERATED REFLECT NEWS YOU DID NOT WANT (STRICT)? 67
FIGURE 17 – DID THE NEWS GENERATED REFLECT NEWS YOU DID NOT WANT (LOOSE)? 67
FIGURE 18 – DID THE NEWS GENERATED CONTAIN THE CONTENT YOU EXPECTED? 68
FIGURE 19 – WOULD YOU HAVE LIKED TO CHANGE YOUR USER MODEL? 69
FIGURE 20 – USERS PERCEIVED DIFFERENCE BETWEEN THE TWO ONTOLOGIES 70

1 Introduction

1.1 Motivation

Personalisation of information is an area of research which is provides an alternative to the "one-size fits all" [19] view of today's World Wide Web. Personalisation of information allows a user of a system to have a more enriching experience as they are less likely to be sifting through information which is of no interest to them. Personalisation provides a user with a unique view of information, as the information is adapted towards a user based on information about that user such as prior knowledge, interests, and learning goals.

The primary benefit of personalisation is that it provides each user with their own view of information, adapted towards that user in the best possible way. In providing a user with a personalised piece of information, they can then process that information better than if it were presented the same way for all other users. Personalisation provides a methodology for introducing users to information in different ways based on approaches a user would normally take in gathering the information.

Ontologies provide a structured, semantically rich way of modelling a domain. They are an extension of the eXtensible Markup Language XML [1], and can be built using description languages such as OWL [26], and DAML+OIL [45]. Ontologies count classes, inheritance, relationships between classes and instances as some of their major benefits.

Personalisation can only be done however, by incorporating models for personalisation. Such models include the user model, which is used to model a user's interests, the domain model, which describes a domain of interest, and an adaptation model which provides the personalisation. Models such as these provide support for personalisation, as they represent the different parts of a system which can provide personalisation. These models are the primary focus of any personalised service, regardless of the domain, the user, and the adaptation provided by the system.

1.2 Research Question

The research question posed in this Thesis is whether there are tangible differences in the user experiences of a personalised news service using two different ontologies as domain models. One ontology is what is known as a Strict ontology, and is a semantically rich ontology, containing information about the concepts and relationships between those concepts. This ontology is consistent with what an expert in a particular domain would be expected to develop manually. The other ontology is a Loose ontology and has little or no semantic meaning attached to it and apart from using the syntactical structure of ontologies could be considered a taxonomy. It recognises relationships between concepts, but has no knowledge of the kind of objects they are or their associated gravity within the domain they reside. This ontology could, conceivably, be developed through mining an existing information resource about the domain.

1.3 Objectives and Goals

In order to answer the research question posed, the objective of the thesis is to evaluate the tangible differences between the results supplied by two different domain models. The main goals of the system being produced are that it

- Delivers a satisfactory news service to the users of the system
- Separates the adaptive logic for personalized news from the news content
- Does not require a priori 'mining' of the news content
- Utilises ontologies as the basis of news domain representation
- Is extensible, allowing additional adaptive axes to be accounted for
- Has reusable, independent components

The thesis will investigate if there is a difference in the results provided by the two separate domain models, and if the results are tangible, however small. The primary goal of the thesis is to establish the benefits and drawbacks of each approach, and come up with a reasonable conclusion of which type of ontology would work better in the area of personalised news

1.4 Contribution to State of the Art

The two different types of ontologies must be compared and contrasted to establish the relative benefits between the two, when set against their development costs.

A Strict ontology provides a structured, semantically rich, view of a particular domain. This type of view is something similar to a domain experts view of the domain. The domain expert would have an in depth knowledge of the domain, and would be able to classify, and weight concepts within that domain, providing a domain model which accurately describes most, if not all, of the facets of the domain. The strict ontology is accurate, but it takes a lot of time and effort to develop.

A Loose ontology provides a taxonomic view of a domain. It is a view that could be developed in one of two ways, one, by looking at a strict ontology, and stripping away all of the semantic meaning from it. The second way, is by means of trawling articles within the domain space, to develop links or relationships between concepts in the domain. The time and effort that goes into developing a loose ontology is not as much as a strict ontology, but in choosing a loose ontology, the accuracy of the domain model tends to suffer, as there is little or no semantic meaning, and the concepts appear to have little weight attached to them.

This thesis aims to contribute an analysis and evaluation of the relative benefits of the two types of ontology models.

1.5 Overview

This thesis proposes a service oriented, ontology driven methodology for providing Personalised News Services.

The first chapter provides an insight into the State of the Art in the areas of adaptive hypermedia and personalised news. Various adaptive hypermedia systems and personalised news systems are introduced and critiqued, as it is those systems on which the basis of this thesis is built. These systems are analysed by means of metrics available within adaptive hypermedia such as how users are modelled, how the domain is modelled and the adaptive axes and techniques in use in each system.

Following on from this is the design chapter, which analyses and produces a design for a personalised news system which best fits the modelling of the domain. It outlines a framework, and a service architecture for such a personalised news service, and discusses how the State of the Art has impacted on the design of the system.

The implementation chapter provides an implementation of the design of the Personalised news service. The chapter outlines the technological architecture of the system, the technologies which were used to develop the system, and a guide to the different services which formed the basis of the service oriented architecture.

The next chapter provides an evaluation of the system, with respect to two things, the service oriented architecture and user trials of the personalised news service. The user trials are based on a questionnaire filled out by users to determine their satisfaction with the service. The evaluation of the service oriented architecture is based on object oriented programming paradigms such as reusability and independence.

The final chapter concludes with a discussion of the contribution to the State of the Art and some future work which could be carried out in the area.

2 State of the Art

2.1 Adaptive Hypermedia

There are six different kinds of adaptive hypermedia systems [18]: educational hypermedia, online information systems, online help systems, information retrieval hypermedia, institutional hypermedia, and systems for managing personalised views in information spaces. This project will look at the sixth type of system, although it should be noted, that (far) more emphasis has been put on the first four systems in the past. Now however, the sixth type of system is becoming more prominent.

Adaptive Hypermedia is an application area for user modelling, and an Adaptive Hypermedia System (AHS) tries to adapt information for a user based on a model of that particular user. The user model allows the system to produce hypermedia which best fits the user, by deducing from the user model, aspects of the hypermedia which the user may find interesting or useful to satisfy a particular goal. User Models present to an adaptive system a balance of control over the adaptivity of a system. The user can have control over how the system adapts, or the system can have control itself. This balance can have an effect on how well the system can adapt hypermedia provided to it.

Adaptive Hypermedia is an area which can be quantified and analysed by metrics such as –

- Which adaptive axes/techniques are leveraged to produce adaptation
- Where the adaptivity occurs
- How users are modelled and represented in the system
- How the domain in which the adaptivity occurs is represented.

The adaptive axes are the part of the system which will provide the adaptation. Adaptive Techniques provide the methods around which the axes will be adapted toward. Adaptive axes can be divided into two different categories, the learner and the context. User axes are aspects such as knowledge, culture and learning style, which are unique for each user, and which can have an effect on how the user will go about using the system. For example, if a user has a good prior knowledge of a content domain, they may not want high level information about that particular domain, rather they would want more specific information regarding the domain. User axes provide ways in which modelling of users can be done. If there is a concentration of axes which are important to a user, then it is likely that the user model for the system will be complex. If very few of the axes have a significant effect on the user, then the likelihood is that a simple user model will be employed by the system. The other axis which has an effect is the context axis. Context refers to the aspects which do not directly relate to a user. Examples of context axes are bandwidth, input, and screen size. Context axes are independent of the user, and context is not modelled at all.

Adaptive techniques are the methods which will provide the adapted hypermedia for a user. There are four such techniques, which are adaptive navigation, adaptive presentation, structural adaptation, and historical adaptation. Adaptive navigation provides a way for the user to step through the system by tailoring the link format to the user based on their model. This type of tailoring can be done by way of link annotation, link hiding, or link sorting, providing a unique view of the system for a particular user. Adaptive presentation is intuitively related to how the hypermedia is presented to the user. The hypermedia or content is adapted towards the user model provided, and can be done in many ways such as inserting or removing information, altering the information provided, or compiling the information in a different order. Structural Adaptation allows a user to have an idea of where they are in terms of the hypermedia environment. Using such methods as fisheye views can provide the user with a spatial view of the content domain, and where they can go to next. Historical Adaptation is the final technique provided, and as expected, relates to tracing the history or path which a user has travelled to get to a particular point in the system. By providing milestones or landmarks for a user, the system can adapt to a user's path, and provide adaptation of information.

User models depend on the system itself, and the extent to which information should be personalised. A simple user model will generally tend to provide generic personalisation, whereas a complex user model will tend to provide in depth personalisation. Some systems don't necessarily require complex user models and can benefit from simple models. There are different ways of modelling a user for an adaptive hypermedia system. Modelling a user in an adaptive system should generally come from the user's adaptive axes, and make use of the most important axes relative to the system. For instance, cognitive style and prior knowledge of a topic may be important when modelling a user for an adaptive learning course, and should have a higher weight in the user model than something like communication style. User Models will tend to be different for different types of adaptive hypermedia systems.

The bootstrapping problem is one which crops up again and again in Artificial Intelligence in Adaptive Hypermedia. Projects such as [9, 33] have tried to find a way around it. The bootstrapping problem is the problem of new users entering a system, and how to go about adapting information to new users effectively and without error. As a new user enters a system, interests and attributes have to be attached to a user model, but usually, these interests and attributes don't really give a good enough idea about the type of information an adaptive system should obtain or provide for a user.

An adaptive system can also depend on the presence of a domain model, which outlines objects and relationships between those objects in a particular concept domain. A domain model can help adaptivity, by providing an overall view of the content domain. If there is an overlay between the user model and the domain model, a more intelligent adaptivity is available to the system. This is because the user model will have some sort of knowledge of the domain, even if it is only very small.

2.2 Survey of Adaptive Hypermedia Systems

AHA! [20] was primarily built to support online courses, but now provides a way to add different adaptive features to online information systems. AHA!s major advantage is that it is simple. The AHA! architecture is one which is replicated in many other systems due to its simplicity and its generality. The AHA! architecture consists of a server, containing domain models and user models, which interact with local or remote pages to produce servlets of personalised information. Its simplicity and generality are also a drawback of the system. Due to the lack of specificity on the domain in which it provides personalised services, these services can be considered simple in their basic adaptation. AHA! employs both adaptive presentation and adaptive navigation techniques to provide an adaptive service. Adaptive presentation is performed using the conditional inclusion of fragments of text. Link Hiding and link annotation is also used, to provide the user with a set of links which fall into specified groups. Adaptivity occurs in the domain model, as a combined model. Users are represented in AHA! with a user model based on concepts. Attributes are associated with those concepts, and are updated dynamically by the system as the user navigates their way around the concepts available to them.

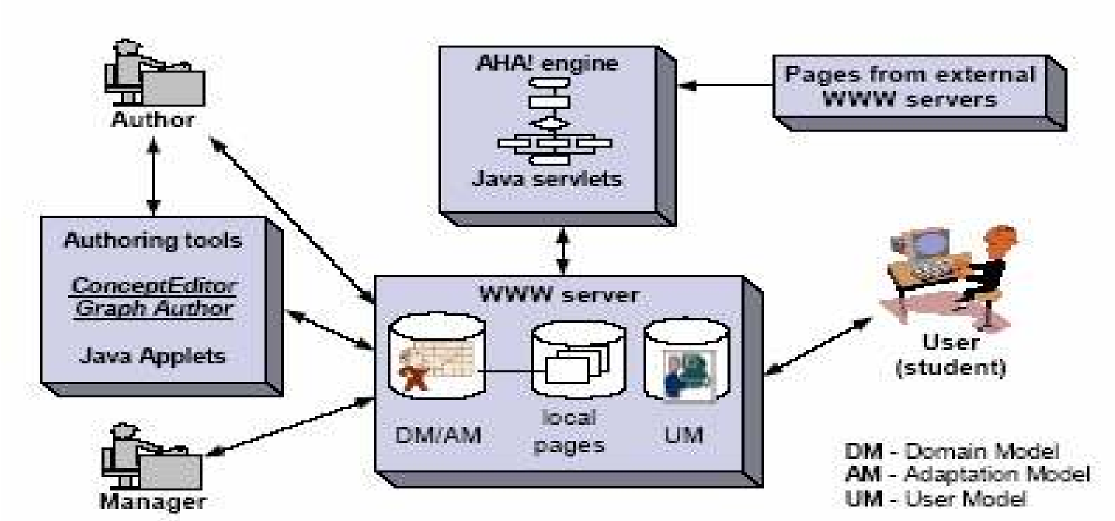


Figure 1 – AHA! Architecture

ELM-ART [38] is another system for adaptation, which was built as a web implementation of an Intelligent Tutoring System. ELM-ART focused on users who wished to learn the programming language LISP. ELM-ART provides the user with both a textbook and a reference manual for consumption. ELM-ART's primary benefit is its knowledge of the LISP language. Due to its knowledge of the content domain, it can provide a better, more intelligent form of adaptation for the user of the system. ELM-ART provides adaptive navigation, based on link sorting and link annotation, as a technique for adapting information towards a user. As ELM-ART is an intelligent tutoring system, the axes of Adaptivity will lean towards learning oriented axes such as cognitive style and prior knowledge rather than maturity and cultural background. The user is represented as a knowledge base, having knowledge about problem solving in LISP. The knowledge base consists of a network of concepts, and in placing the Adaptivity in the users knowledge base, this lessens the amount of models that are needed to provide adaptivity. The domain model is based around that of a regular textbook, with chapters and subsections, with the knowledge base working with the domain model to analyse the amount of knowledge a user has.

Adaptive Engine 3 (AE3) [41] is a system which uses the multi-model approach to provide adaptive services for a user. AE3 takes an approach which emphasises reusability of objects, and attempts to have smaller services which can do tasks rather then having one large system which can do everything. This type of system promotes component development which makes the system easily extendible. In taking the multi-model approach, the system treats all models in the same way. AE3 has five main components, the Data Manager, the Engine Manager, the Model Manager, the adaptive engine itself and the utilities component. The data manager provides a method of retrieving models independently of where they are stored, be it a local file system, or a remote XML database. The engine manager allows the use of multiple engines running at the same time. The model manager provides internal representation of models within memory, allowing for easier access. The utilities component allows for the inclusion of new technologies. Perhaps the most interesting part of the system is its ability to make use of Web Services. Web services provide another outlet for the adaptive engine and allow the manipulation of data which would otherwise not be possible using only the adaptive engine itself. AE3 makes use of both adaptive presentation and adaptive navigation when adapting information. The adaptivity occurs in a separate model, the narrative model, creating an independent and reusable model for adapting information. The user modelling is independent of the system, and the user model can be determined by the type of system that is being developed. The domain model can be represented in different ways, such as a domain ontology.

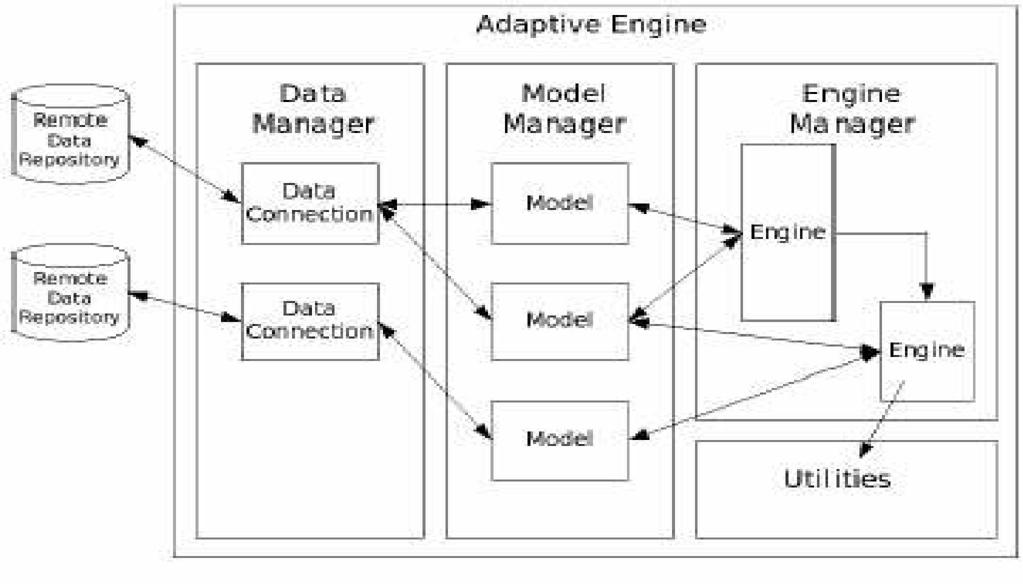


Figure 2 – Adaptive Engine 3 Architecture

There is as yet, no system which can provide a high level of adaptivity on a non specific domain of content. Adaptive hypermedia systems sometimes have a domain model, and a set of rules governing adaptivity which are independent of the domain model. There are also adaptive hypermedia systems which have rule sets which are directly dependant on the domain model provided. Without a domain model, it is very difficult to adapt information specifically for a given domain. Given a certain amount of knowledge of a domain, an AHS can perform a good personalisation of content in that domain e.g. ELM-ART. Given no knowledge of a domain, an AHS can only provide basic personalisation on content in that domain e.g. AHA!. AE3 provides both methods of personalisation, but cannot improve on giving highly personalised information without suitable knowledge of the content domain. So it can be inferred, that to have little or no knowledge about a content domain, yet still provide good personalisation is a very big jump to make.

2.3 Analysis of Adaptive Hypermedia Systems

In order to analyse adaptive hypermedia systems, it is important to compare and contrast them against the metrics mentioned earlier. This will provide a simple but effective analysis on the benefits and drawbacks of each system.

In keeping with the AHA! philosophy of everything at a high level being as simple as possible, the user models which AHA! uses are not complex in any way. This is not to say the user models are not good enough for the system, but may merely mean that all that is needed is a simple user model. For a generic AHS, simple user models would be more prevalent, whereas for something like ELM-ART, where there is a specific type of information to be personalised, the user model is more complex and refined. AE3 provides a developer with the option of how specific a user model may be. Depending on the content, the developer has the power to simplify, or increase the complexity of the user model. Something like this provides a distinct advantage over other adaptive hypermedia systems, as it gives the developer a lot of flexibility in how to model a user. By not constraining the developer to use a particular user model, the system provides room for more adaptive solutions. Extreme flexibility can be a problem however. If a developer is overloaded with choice, then a situation similar to cognitive overload can cause problems.

Adaptive Axes are characteristics towards which a system will perform its adaptation. Adaptive axes come in two forms, the user, and the context. One user may have different characteristics to another such as their interests in the information being adapted or their prior knowledge of the content domain. The context axis is determined by things that have nothing to do with the user itself, such as the input to the system, the device on which the output will reside, and the cost of providing adaptation of information.

The location of adaptivity, as mentioned before, can have an impact on the performance of a system. The Adaptation Model in the AHA! system, is combined with the Domain Model, to provide adaptation of information. The combination of the two models keeps with the simple aspect of AHA! and this simplicity goes even further by having parts of the Domain Model/Adaptation Model also in the User Model. ELM-ART places the adaptivity in the knowledge base of a user, so as a user develops more knowledge of a particular content domain, this knowledge is taken into account when producing an adaptive response for an information request. AE3 places the adaptation firmly in the narrative model, which frees all other models from multiple responsibilities. In placing the adaptation model with the domain model as AHA! does, it creates dependencies between the models. It also provides an approach which does not have many reusable elements. Placing the Adaptivity in the knowledge base for a user is a similar approach in terms of its faults. Placing Adaptivity in the user model prevents reuse. AE3 provides a reusable, independent model driven approach, and provides adaptivity with its own model.

The domain model is combined with the adaptation model in AHA!, and provides a generic and simplified output for a user. ELM-ART, as it is a content specific system, provides a fine grain adaptation for a user, as it has a domain model, which is very specific, and relates well with the content model. AE3 again, proves its diversity by providing both options to a developer. AE3 allows the use of a simple domain model, a complex domain model, or something in between if that is what is required. Building a domain model provides a designer with a time consuming task. A domain model also requires an author with significant expertise in the area of interest to provide an accurate domain model. Providing a user with a domain model allows information to be more accurate, and closer to a users goals or interests. Without a domain model, a designer has one less thing to implement, but the implication at runtime is important. Information adapted to a user without a domain model will be generic, and may not represent a user's interests to the best possible degree.

Overlay is the presence of metadata in more than one model, so as to improve adaptation. The overlay of the user model and the domain model is something which is used quite a lot in AHA!. The overlay occurs with the attributes of concepts of the Domain Model also appearing in the User Model. This can provide some knowledge about the domain in the user model and make adaptation a little more accurate. The system provides attributes which are common to both the user model and the domain model/adaptation model. ELM-ART tries to keep the overlay of the domain model and the user model as small as possible. ELM-ART is based on its knowledge base of problem solving in LISP. AE3 allows the developer to decide the amount of overlay between the user model and the domain model. Overlay allows for knowledge about a certain domain (or domains) to be shared, and used by other parts of the system. Better knowledge of a model improves adaptivity. It is important to make sure however, that overlay is not used too much, as too much overlay can degenerate into models having too many dependencies on each other. If there are too many dependencies within the system it doesn't make for a well structured, reusable system.

Looking at the analysis, AE3 comes out on top when comparing metrics, and comparing the scale of possible extensions to the three systems mentioned. In allowing more freedom to develop the kind of system that is more beneficial to the user, it provides a good opportunity to develop an adaptive element which is not restricted by constraints such as only having a generic output for an information request.

2.4 Personalised eLearning

Personalised eLearning is a primary subject in the field of Adaptive Hypermedia. The major focus in all eLearning systems is the Learning Goal of a user. The Learning goal of a user can incorporate many things, including developing analytical skills, communication skills and learning skills, as well as progressing knowledge of a particular topic. If the system can adapt to any users' learning goal, then it can be said that the system is a highly adaptive one. eLearning also requires pedagogically sound rules, which adds a difficult constraint to its implementation.

Different models form the basis of personalisation in [24]. The Learner, Narrative and Content models are treated as separate entities which promote reuse of models. An adaptive engine reconciles the three different models at runtime to provide a personalised course for a learner / user. The learner model consists of concepts the user is familiar with, is not familiar with, and needs to learn to finish the course provided. The narrative model consists of a definition of the rules which allow a personalised course to be derived for a user. The rules maintain the range and scope of topics. The content model is simply that, a model of the content which needs to be learned by the users of the system. These three models combine at runtime to provide each user a personalised course based on previous knowledge, and what they have to learn.

The aLFanet project [35] also uses a multi model approach, going further by applying more models to the content which is to be adapted. User interests are not the only characteristics which determine the adaptation, as the context of the document can also determine the amount of personalisation applied to a particular article. In order to do this, the extra models are needed so as to get an even more accurate response. The user models used in the system are a combination of machine learning techniques, and knowledge base techniques.

In Specht et al. [36], two pedagogical approaches to adaptive learning are discussed / explored. The WINDS system tries to employ reusability of its learning objects as much as possible. The learning objects form a hierarchy, making it easier to establish a pedagogical approach to take. The two different pedagogical approaches are expository learning, which exposes the user to, and directs the user through the course, and exploratory learning which encourages the user to explore the course and find their own way around the course. An adaptive course can also have external resources attached to it, for further interest one may have in the topic, and this is not unusual in eLearning courses in general. The user model used in this system is regularly updated by the various methods available to the system including submission of tests and topics covered over a duration of time.

2.4.1 Adaptive Authoring

Adaptive Authoring is primarily a feature of eLearning. Its primary focus is on the authoring of online teaching courses, in a wide range of disciplines [16, 23]. Adaptive authoring helps with the establishment of achieving the user's learning goal.

NetCoach [17] provides a solution to the problem of Adaptive Authoring. In the past, simple adaptive systems could be built by people who were not very technical, but by this token, the systems would be as the name suggests, simple, and not much good for someone learning a new concept on their own. In contrast, good adaptive systems tended to be built by technically savvy programmers, and cost a lot of money. NetCoach allows non-technically gifted people to be able to build highly adaptive courses for learning. NetCoach allows the user to find the optimal path to achieve their learning goal. NetCoach allows the authoring of learning material, tests, learning goals and layout of the interface.

In Christea et al. [21], a set of development guidelines are put forward for adaptive authoring systems. This is because it is not realistic that teachers will have the know-how to be able to apply adaptation to their online courses. The guidelines are based on general course structure, text presentation and structure, lesson composition, authoring views and student adaptation facilitation. These guidelines are based on the comparison and contrast of two authoring systems, and can, in the first instance (course structure) at least, be thought of as similar to what is done in [15]. It must also be noted, that it is mentioned that pure traditional editing must also be allowed, but only in extreme cases.

The ACCT [42] is an authoring tool which places its focus firmly on the pedagogical aspects of eLearning. It is noted that pedagogy tends to take the back foot in the design of eLearning systems, and the ACCT aims to prevent such a direction being taken. The ACCT also allows the creation of a knowledge domain for eLearning courses, providing the course developer with a proper way to model the domain. The ACCT provides the developer with structured, sound and reusable pedagogic models around which to build an adaptive eLearning course. An example of these models are case-study, and didactic strategies.

2.5 eLearning Analysis

eLearning provides possibly the most obvious use of adaptive hypermedia. It provides a strong pedagogical way of helping a user become more enlightened on a particular subject. There are some good systems out there which facilitate adaptive learning on a range of courses. Adaptive Hypermedia systems do suffer in comparison to Learning Management Systems however [37]. Learning Management Systems tend to provide a variety of teaching services, whereas AHS tend to focus on presentation and organisation of content. LMS also require little or no overhead when introducing new content, whereas it can be somewhat costly to introduce new information to an AHS. It also remains to be seen how broad the range of courses becomes in the future. While teaching SQL [23] to users is a success, it is a very technical course compared with something like English or Theological Studies. If eLearning were applied to courses such as those which require a different mindset, a different pedagogic approach, and a different teacher, it would be interesting to see the implementation and more importantly the evaluation of such a resource.

AHA! mentioned above, is simple, and this is the reason behind its popularity, but this can be a disadvantage [22]. If the system complexity is low, then the authoring efficiency can't be very high. If the authoring efficiency is low, then it makes for little or no room for good adaptation within a learning course. Like all adaptive systems, it is important to find the right balance between system complexity and the authoring efficiency.

2.6 Survey of Personalised News

Personalised News is the area in which this thesis is focused, so this section will provide an insight into personalisation of news with respect to different systems available in the area of Adaptive Hypermedia.

In Merialdo et al. [30], a novel approach is taken to the adaptation of news. In this case, the approach is to adapt video to users' needs. This however, is not very different to adaptation of text, as the video can be indexed or annotated with metadata providing a good source of uniqueness of the data stream. The project deals with a very large scope of news however, including international politics, national politics, international society, national society, economy, culture and sport. The user model provided by the system combines a level of interest in a category (or multiple categories), and uses a labelling mechanism to annotate a story with an associated importance level. By combining the two, and using a probabilistic formulation, a simple yes or no answer can be given to determine a users interest in a particular story. An added constraint in such a system is that the combined personalised news broadcast is less than the allotted time a user has to view the broadcast. The number of adaptive axes used in this system, is relatively small. Prior knowledge and preferences are likely to be the two most interesting axes in use. Adaptive Presentation is in use, as articles of interest are compiled together so the user can view the video articles. The users were modelled in a simple way, by providing a level of interest in particular topics. Yet the simple user model did not significantly affect the news provided for the users. The domain models, were also simple enough, being divided into several categories, one of which each story would be associated with. Having each story carrying a weight from 1 to 100 allowed for a good probabilistic chance that the user would not receive information which they would deem unsuitable to their needs.

In Jokela et al [31], structured content, in the form of domain ontologies, are used. News content is provided with a semantic structure, which is then compared to a user profile to establish relevant news articles. It is noted however, that interests can change and shift over time, and to provide a mechanism to adapt to such changes, user feedback is employed. In gathering user feedback, the ontologies can change, and become even more powerful than static ontologies. SmartPush aims to get rid of some of the shortcomings available in commercial systems, such as the lack of customisable ontologies and relative depth for experts. The SmartPush system applied weights to semantic relationships between objects, culminating in more power and expressiveness in the system. The SmartPush system concentrates on one primary axis in the user context, the preferences axis. In making this axis adaptable itself (in that it can change based on knowledge and behaviour), SmartPush concentrates its efforts at design time on the most important part of the system. SmartPush uses Adaptive Presentation as a technique, only providing a user with information that it deems to be satisfactory, or interesting. It was also noticed that depth of interest, especially for experts in a given domain was an important factor in modelling the user. The domain model, was represented as an ontology, providing semantic meaning to the domain model. As the concepts carried weights, to distinguish between important and unimportant concepts, the domain model had more power over the information provided by the content.

Billsus et al. [32] look at adapting news from a different point of view. Using machine learning techniques, news is adapted for users, but the users may be using one of two types of hardware, high powered machines like desktop or laptop computers, or lower power devices such and PDAs or mobile phones. User feedback is employed in this project also, and is stored with the news stories themselves. The machine learning techniques are relatively simple, but powerful. Because of the two different types of device in use, different machine learning techniques are used to derive feedback from users. For the high powered devices, explicit feedback is preferred, where users rate the item, and additional feedback methods include the amount of time spent reading a particular article or amount of further information requested is deemed as a topic of high interest for the user. The machine learning technique for the low powered devices however is not as sophisticated, and is by and large, implicit behaviour. If a user requests a story based on a headline, then this can be considered a topic of interest for the user. The AIS system also takes into account the short term and long term interests of a user, using a different AI method for each. Nearest neighbour algorithm is used for short term interests, as stories which are similar, will be interesting to users. Naïve Bayesian classifier algorithm is used for the long term interests, as it would represent a users general interest in a category or topic. The contextual axes are placed at the forefront of the Adaptive Information Server. The system was available on both regular desktops, and low powered palm devices. The contextual axes such as bandwidth, screen size and cost have an effect on the way the system works and adapts information. The user axis also plays a part in the system, as the users are modelled based on preference, and also on past behaviour regarding interests in particular articles. Scalability is noted as a design issue with respect to user models within the system. Adaptivity is performed at the central server, which also stores all the of the user models, so that as little of the device's power is being used to provide the service as is possible.

SeAN [33] is an adaptive system which starts by classifying documents into a tree, made up of sections and subsections. The hierarchical nature of the system is in parallel with the same kind of structure associated with newspaper editorial systems. SeAN attempts to be able to personalise the detail level of a news document based on the user model. The user model is an ontology, which is rather different to most ontologies, as most ontologies represent content or conceptual domains. The user model is broken down into different dimensions, providing an altogether different view of a user than is usual. These dimensions are Interests, Expertise, Cognitive Characteristics and Lifestyle. Behaviour tracking is used to a great extent in the system also. Such behaviour as the time spent reading a news article, false positives and misses are taken into account, and these instances are learned from. User axes such as preferences, cognitive style, and domain knowledge form the basis of the user modelling approach. Stereotypes such as these can provide the system with enough knowledge to base its first few adaptations, with adaptability becoming more focused as more use is made of the system. Due to the domain being represented as a hierarchical structure of articles, from high level concepts to lower level niche topics, adaptive presentation is used where only information relating to the user model is provided, and other redundant information is not presented to the user.

PIN [34] is an adaptive system which uses neural networks to learn user profiles. User profile learning is done quickly and easily using this method. User profiles grow and adapt to new interests of the user. User feedback also helps with the dynamism of the user models. Adaptive Navigation forms the basis of the adaptive techniques used in PIN. Links are sorted based on relevance, in decreasing order. Users are modelled by associating concepts from the domain, with interest weights provided for those concepts. Concepts have semantic meaning, providing more weight for the information which is of interest to the user. User Models are also updated “on the fly” as the system is being used, providing an adaptive user model, which is of better use than a static model.

2.7 Analysis of Personalised News

In Merialdo et al. [30], the user model is a simple one, which only has a users interest in a category (or categories), and matches the stories annotated importance level with the level of interest in a category. This is a simple but effective method of personalising news for a user. There are however constraints involved in such personalisation, more so in the content, when annotating a news article. The annotation process should be very accurate, and not give an importance value to an article which is way off the importance value it should be given. The way in which the articles are chosen however, is an effective method of personalising news for a user.

Jokeli et al [31] constructed their own ontology, after sifting through a years worth of news articles to find the concepts which would be of major importance when creating a domain model. Once the domain model / ontology was in place, the personalisation process was able to begin. One problem which occurred was the broad range of news which was available to be personalised. For domain specific articles, the ontology was not appropriate as it only provided a generic view of multiple domains such as politics, sport and entertainment. In order to have a better news service, domain specific ontologies would need to be created to account for as many domains as possible. This provides acceptable personalisation independent of the type of news article being presented. Intuitively, this can however, take time to develop. The ontology provides a good semantic structure to the domain model, by having weights attached to relationships, rather than having one simple "related-to" relationship. The ontology was a rigid one, where it did not adapt over time, growing with the content which it described.

Using ontologies to describe user models is an interesting approach to personalising news. SeAN [33] uses ontologies to represent users rather than having a complete knowledge base which can incorporate all facets of a user's interests. Using an ontology for user modelling creates a reusable, easy way to provide a user's interests in a model. The user models SeAN used modelled four separate dimensions, and used separate knowledge bases for each dimension. The user models are changed with the ending of each session, as the system tracks a user's behaviour and modifies the model based on the behaviour (such as clicking on lots of financial news links)

Merialdo et al. [30] used a simple user model, as does the SmartPush system, and these simple methods of representing a user do not have an adverse effect on the results. In using an ontology to represent a users interests, SeAN uses a complex method of modelling a user, albeit one which can be refined as the system is used more and more by a user. The difference in the end result may not depend entirely on the user model, and how it is represented. All of the systems find the need for a representation of the domain in terms of some kind of model. Merialdo et al. provided a high level breakdown of categories of news such as politics, entertainment and sport. A domain ontology was created for the SmartPush system, and SeAN had a hierarchy of news concepts, which broke down further into other sub concepts. It seems that however one could look at a personalised news service, a representation of the domain, whether it be fine grain or abstract level, should be present to achieve good results. Adaptive navigation and adaptive presentation appear to be right techniques to use when delivering personalised news, as they provide enough adaptivity towards a user. Contextual axes do not seem to have much of an impact on the systems which have been mentioned, but as systems develop, and context becomes more important, they will have to be looked at in the same capacity as user axes.

The most significant thing which can be learned from looking at these Personalised News Systems is that it is important to have a good understanding of the domain(s) in which the news is available. Generic understanding will only give generic and non specific results back to a user. If there are a number of domains, then having a content specific domain model for each domain will provide better personalisation, and ultimately, lead to happier users. Domain models which have a good semantic structure will also provide better news than domain models which have basic relationships between objects or instances. Finally, it should be noted that user models may not have to be complex in order to provide an adaptive service. Complex user models are not needed in all systems.

2.8 Artificial Intelligence Methods for Personalised News

AI methods are the most common way to personalise hypermedia documents [18], and are generally used for web recommendation systems [19], although they are used in other scenarios also. There are generally thought to be two main methods of AI personalisation, content based filtering, and collaborative filtering. Content based filtering primarily focuses on information the user has provided in the past. Collaborative filtering focuses on similar users and their interests. Like many AI methods, both of these have their strengths and weaknesses, but generally once there is a large enough test set and the system has had plenty of use, the weaknesses are not as apparent.

AI methods generally use learning based methods to personalise information for a user or group. PTV [4] is a good example of this. PTV is a personalised TV Guide, which, based on profile editing, can provide an accurate television viewing guide for a user. PTV learns the majority of its information through grading feedback from the user. Each program can be graded by the user through a grading icon or link available to the user. The main disadvantage of manual grading, is that users tend to find it a burden.

A prime example of promising test set is UseNet News. In [5], collaborative filtering is applied to a subset of UseNet News. UseNet provides a high number of posts for its users, and sparsity is a problem. It is therefore valuable to filter information. In this project, there are four types of scenario that can occur, hits, misses, false positive and correct rejections. Hits and correct rejections should be maximised, while misses and false positives are to be minimised. It should be noted however, that the effects of a high rate of misses and false positives depend on the type of domain in which the user actively uses. For example, the result of missing a post on a good movie is not necessarily a big deal, but missing a post on an expensive restaurant (e.g. don't try the beef) could be costly, both in terms of price, and time spent. UseNet has a low cost, in that the cost of a false positive is annoying but nothing more, as it only takes a couple of seconds to gather that the item you are reading is not suitable for the topic you want to discuss. Misses can also be determined as low cost as truly valuable articles will tend to reappear in follow-up discussions.

In addition to collaborative filtering based on a single space, it is also possible to combine methods of collaborative filtering as shown in Unison-CF [6]. Unison-CF combines the advantages of the collaborative filtering methods it chooses, user based filtering and item based filtering. User based filtering is based on similar users, whereas item based filtering is based in similar items. Recommender systems such as Unison-CF are common in web based environments such as Amazon [8].

TV Scout [9] attempts to work around the bootstrapping problem of new users in a recommendation system. The system slowly unveils the filtering characteristics to the users with more use. The first time a user uses the system, it takes on an information retrieval look, and then progressively becomes more adaptive to the users needs.

Collaborative Filtering and Case Based filtering are not the only approaches for learning however. An interesting project [7] uses a completely different approach to adapting hypermedia documents. In the project, interest is determined by four main characteristics, time between seeing a headline and clicking on it, time spent reading an article, number of embedded links followed, and percentage of topical articles read. This is a different take on how to adapt documents for users.

2.9 AI Methods Analysis

AI Methods provide a good all round way of adapting information for a user. There are a variety of methods and techniques which can be used, depending on the needs of the end user, the temporal constraints, and other such factors. AI methods can use comparison based on similar items, or users, hybrid approaches, and all of these different methods work in its favour. AI methods do suffer however, with small sets of information. Given a small test set, or a small amount of training data, the ability of AI to adapt information can leave a lot to be desired. AI methods require large test sets to reason against and consult before adapting information to a user. While this is common in systems with great stature such as recommender systems, it can be less common for new systems coming online with little or no good diverse test subjects.

2.10 The Semantic Web

The Semantic Web is an extension of the current implementation of the Web [10], and provides meaning for information, making for easier and more meaningful manipulation of data. The Semantic Web enables automatic processing of data, which is not possible on the World Wide Web. RDF is used to express meaning on the Semantic Web, adding to the structure provided by XML. Ontologies are also very prevalent in the Semantic Web, as they express the relationships between terms.

The Semantic Web aims to add to the human interpretable languages like XHTML by making the web interpretable by machines [11]. This is done with ease through the use of XML based software and the XML language.

It must however be noted, that the Semantic Web may not yet be ready to take over, and is not likely to be accepted as the way forward for a number of years [12]. Two reasons are noted for this statement, one is that most of the information on the web has no semantic meaning, and the second is information sources are likely to change their structure. It is the vast difference between human readable web content and machine readable content, and the difficulty to minimise that difference that suggests the Semantic Web is a long way off. The project in [12] aims to provide access to semantically meaningful data to pages with no semantics.

An interesting project is undertaken in [15], because it attempts to form a map between different ontologies on the Semantic Web. This is due to the number of Ontologies on the Semantic Web at the moment, and the relative explosion of ontologies which will more than likely occur if the Semantic Web comes to prominence. The GLUE system, which was developed as part of this project, attempts to create semantic mappings across ontologies using machine learning techniques.

2.11 Ontologically Driven Adaptation

Using Ontologies to adapt information is the way in which adaptation will be done in this project. Ontologies describe relationships between terms in a document or set of documents. There are primarily two methods of adaptation using ontologies, one being automatically generated, and the other being manually generated. Both have pros and cons associated with them. Ontologies can take a significant amount of time to construct [13], and automatic ontology creation can suffer from the restricted ontology problem. The restricted ontology problem is due to a concept that is needed not being available to a user. This problem is overcome in [13] through users being able to add more concepts to the ontology.

Ontology maintenance is also an important factor when taking into consideration adaptation using this technology. In [14], a lightweight ontology is used so as to help with updating the ontology as is needed. Ontology maintenance is however, still a research area in its early stages, and has a while to go before it makes it way into the mainstream of ontology driven adaptation.

2.12 Technologies

2.12.1 XML

XML stands for eXtensible Markup Language, and is a W3C recommendation [1]. XML was designed to describe data, (*unlike most other markup languages (e.g. HTML), which were designed to display data*). XML uses tags to describe data, and these tags are not predefined. You must invent your own. XML will also be used quite a lot in the future, and this gives the opportunity for a project such as this, to have a longer lifespan. Due to the degree of use of the language, and its simplicity, it will be used to a great extent in this project. XML will form the input, output, user models and ontologies which will be in use within the project.

2.12.2 RSS

RSS stands for Really Simple Syndication, and is an extension of XML. RSS will be used in the project as it will be the method of describing news items. RSS is easy to manipulate given that it is an extension of XML, but does have some faults associated with it such as only showing an abridged version of the content, and not having a history.

2.12.3 RDF

RDF stands for Resource Description Framework and is a general purpose extension of XML used for representing information on the web. It is a W3C Recommendation [25]. RDF Schema (RDFS) is RDF's vocabulary description language, and is an extension of RDF. RDF gives semantic meaning to description of resources and relationships between these resources, and is the primary technology in use in the Semantic Web.

2.12.4 Ontology Structure (OWL/DAML+OIL)

OWL stands for Web Ontology Language and is used to describe classes and relationships between ontologies. It is also a W3C Recommendation [26]. OWL is a vocabulary extension of RDF, and is derived from the DAML+OIL Web ontology language. It provides additional vocabulary as well as semantics. Vocabulary such as cardinality, disjointness, and relationship characteristics allow for a richer description language than could be provided by other technologies. OWL provides a methodology for producing documents which are intended to be processed by applications rather than interpreted by humans. OWL has three sublanguages, OWL Lite, OWL DL, and OWL Full, with OWL Lite having the simplest description logic. OWL DL provides more description, and OWL Full provides the maximum description and expression of the three

DAML stands for DARPA Agent Markup Language, and it is not, like the others mentioned above, a W3C recommendation. It does however, offer the same functionality of OWL, and like OWL, is built on top of RDF.

2.12.5 Protégé

Protégé [39] is a tool developed at Stanford University, which aids in the semi automatic construction of ontologies. Protégé works with OWL DL, and provides a simple and easy to use method for the creation of ontologies. Ontologies are difficult to create manually, and are time consuming to create, so the employment of a good tool to help with the creation is important. It allows the creation of classes, class relationships, and class instances. The tool however, does not employ anything for easy visualisation of the ontology, which not so much causes a problem, but does not allow novice users to garner an understanding of the overall picture of the ontology being created

2.12.6 JESS

JESS is the rule based engine which will be used in the project. The system is written entirely in Java [3]. JESS allows the deletion, creation, and modification (amongst other things) of models, which are in the form of XML files.

2.12.7 JATHA

JATHA [27] is a common LISP library in Java. It provides an almost complete subset of common LISP, and due to the fact that it’s written in Java, has the advantage of portability. A separate Java class is used for each LISP primitive, and the user can implement their own new primitives as they see fit. It is also a very fast engine, making it useful for use in today’s world. Algernon [29] is a rule based inference system. It is built in Java, and performs rule based processing, both forward and backward. Algernon is also interfaced with a tool already in use on this project, Protégé. JATHA has helped change Algernon from a LISP engine, to a Java engine, line for line. Given that Algernon is a medium sized rule engine, the job JATHA did is impressive to say the least.

2.12.8 LISP

LISP is a rule based engine for programming. Common LISP has an abundance of data types, such as lists, hash tables and structures, and these are represented as objects, pushing forward the object oriented programming paradigm. This is one of a number of advantages to the language. Other advantages include dynamic patching, extensive control structures. A full list can be found in [28].

2.13 Summary

In this chapter, the State of the Art has been reviewed to determine the possible ways in which different systems can influence the design of the Personalised News Service with respect to the goals of this Thesis.

The thesis aims to present a satisfactory news service for the users of the system. It is important to note that various aspects have to come into consideration when providing a satisfactory news service, such as the user model, and the domain model. In separating adaptive logic from news content, one must look at systems which provide independent reusable components, which allow the separation of such components, rendering them reusable and independent. Ontologies, and knowledge of the domain provide a methodology for providing a domain model, which does not require a priori mining of information. It is also important to have a system which can be easily extended, and it is vital to look at systems which provide that extensibility.

3 Design

3.1 Introduction

The goal of this work is to produce a personalised news service that adapts the news articles delivered to the user's interests. These interests are defined in terms of ontological instances and abstract relationships that exist between them. The depth of interest is determined by the number and type of relationships traversed in the selection of appropriate articles. The delivered personalisation will be in the form of a tailored news feed derived from existing live news sources.

This chapter will focus on the main design issues that occurred in the development of the personalised news service. It will outline how the state of the art has influenced the design of the system. The chapter will outline the domain of news, and why that particular domain was chosen. It will also provide an overall view of the system, from high level components to the finer grained technological decisions made about the system.

Following on from this, the ontological aspect of the design will be looked at. Namely, how the ontologies will impact on the system; how they were created, and why they are as specific as they are will be looked at. Issues surrounding the modelling of user data will be the final aspect that will be examined in this chapter, as this will have a significant impact on the system also.

3.2 Influences from State of the Art

3.2.1 Adaptive Axes

The adaptive axes and techniques presented in the State of the Art chapter provide an insight into what the system will be adapting towards, and of course how to adapt information. The axes will have an effect on the information that will be adapted to a user, and the techniques involved will be based on the axes of adaptivity which are most significant. The user, in this instance, is not trying to develop knowledge through learning, merely through reading through information provided to them. Therefore, cognitive styles and learning goals are less significant than in the case of personalized eLearning systems. On the other hand, the preferences of a user, specifically their interests and prior knowledge are of importance. These axes will have a bearing on the type of information a user will receive.

The techniques involved in providing information for the user are primarily based on adaptive presentation. Structural presentation and historical presentation are of little use when presenting to a user articles of up to date news. Adaptive presentation will allow the insertion and deletion of articles based on the interests of a user. Adaptive navigation, as a technique, cannot be leveraged in the domain of personalised news as it is the news reader client application that determines how news articles are navigated. Taking into account the AHA! system, it provided information based on adaptive presentation and adaptive navigation. Adaptive presentation used in the system decides based on the user model whether to include or not include a fragment of information for the user. These fragments were part of the information in the content domain. Adaptive Engine 3 allows the use of adaptive presentation, providing the designer with the opportunity to include or not, information from the content domain. Merialdo et al. [30] use adaptive presentation for the video which is to be produced for a user. This video contains articles of news based on the users interests. Jokela et al. [31] also provide information for users based on adaptive presentation, with SeAN [33] also making use of adaptive presentation. Looking at the three personalised news services just mentioned, they all make use of the adaptive presentation technique.

3.2.2 User Model

The user model will influence the design of the system, as it will have a large bearing on how the information will be chosen to be presented to the user. The user's interests in a news domain, or lack thereof, must be modelled as well as possible to provide the user with a satisfactory service. The user model must only be as complex as is needed, and given that the domain of news which will be presented to the user is focused, rather than being generic, the user model should be focused on that domain. The domain model will also influence the design of the system, as it is important to have a well structured model on which to base information. The domain model must be structured well and be easily modified to represent any future changes in the domain itself.

The primary influence from the state of the art is to have an accurate model for a user. The more accurate the user model, the more accurate the adaptation, and therefore the likelier it is for a user to be satisfied with the content they are receiving. The user model will be influenced by the system it is being used by however. If a news service has generalised topics of interest, then the user model should have generic, or slightly domain specific interests. This is because there is no need for specific interests, as specific interests will not likely be present in the articles provided for the user. Taking for example, a user with a general interest in politics, who might have generic interests such as Bertie Ahern and Mary Harney. The user is not likely to be interested in a Fine Gael backbencher as it is likely that any news containing that backbencher would have a niche following, and there would be very few articles about that backbencher. The same can be said for highly domain specific news services, the user interests must be domain specific to achieve a satisfactory output from the service. In a domain specific news service, the user interests will have to be specific as these interests will correspond to the type of article they will receive. The more specific the domain, the more knowledge a user will have about the domain, and therefore, the user will have specific interests in instances appearing in news articles.

Adapting and changing a user model over time is something, which given the time constraints of the project, is not feasible, but should be considered important. The user model may change over time. For example, a user may be interested in a team e.g. Manchester United, and the user model may change during the summer as the team buys new players, or sells others. Also, the users interests may change as new players from the youth team begin to play for the first team.

AHA! models users based on concepts, or interests, with attributes forming part of the user model also. This is simple user model, yet it works effectively for the AHA! system. AE3 allows the system designer to determine the complexity of the user model. In allowing the designer to choose the complexity of the user model, AE3 provides an advantage over systems such as AHA!. Merialdo et al. [30] use a simple user model also. The user model is simply based upon an interest in a topic, and a depth or type of interest in that topic. The simple user model, did not affect the news provided in a significant manner. Jokela et al. [31] noted that depth of interest in a particular domain held another key to personalising news as well as possible. This is especially true for experts in a given domain (or domains). By providing a level of interest, an inference can be made into the depth of information a user would like to receive. SeAN [33] also noted that interests and expertise in a given domain were important when providing information about that domain. In conclusion, a simple user model can have just as good an effect on personalisation of information if used properly.

3.2.3 Ontologies

Ontologies provide the semantic relationships between objects and instances in a particular domain. It is important to have an ontology which is as easy to maintain as possible. Lightweight ontologies make this feasible. Using a tool such as Protégé [39] also allows the maintenance and updating of an ontology in quick and easy manner. Classes, relationships and instances can be quickly added and deleted as needs be. The important thing to note about ontologies is that they can provide a model which has semantic meaning. Semantically strong ontologies provide a structured approach to looking at the instances of the domain.

Jokela et al. [31] make use of an ontology to describe the domain. The domain ontology provided a structured view of the domain, with concepts carrying weights to determine their importance within the domain. The weights allow even more inferences to be made about the suitability of a news article when combined with the user model. In using an ontology to represent a user model such as SeAN [33], an overall, structured, and weighted view can be placed on the user's interests. The structure, and the semantic meaning which an ontology can provide, prove to be two of the greatest assets of using ontologies, regardless of the type of component they aim to model. The reason for this is because ontologies provide a view of a domain which would be similar to that of a human's view. For example, knowing which are the more important concepts in a domain, the important relationships between concepts and the weight the concepts carry within the domain are examples of a domain experts, or even a regular persons view of the domain. Having a structured view such as that provides an easier way to model the domain, and makes it easier to see the differences between concepts, and relationships between those concepts.

The Metasaur system [13] performs automated ontology building. The system allows the insertion of objects into a data dictionary, which is analysed and incorporated into a domain ontology. This kind of automatic creation provides the developer with a much less time consuming method of creating a domain model using an ontology. This method also removes the restricted ontology problem, which is when all the concepts in the domain are not available for markup. The disadvantage with this method, is that while it provides an ontology, there is little or no semantic meaning within the ontology. The domain is trawled, and links are inferred from objects, which lead to relationships being provided between those objects. Without much semantic meaning however, objects which have a higher importance than others, may be looked at in the same way by the ontology created by Metasaur.

3.2.4 Choice of System

In making the choice of which adaptive system to use as the basis for the personalised news service, there are a number of factors which must be taken into account. Looking firstly at the flexibility of the system, AHA! does not appear to be very flexible, the user models are simple, and don't allow for more complex models to be built and used. This lack of flexibility and the associated constraints in how the domain is modelled precludes using AHA! outside of adaptive hypermedia applications. ELM-ART is a tutoring system, and would not be of much use for any type of adaptive system which is not based on learning as it intertwines adaptive logic with the content that logic manipulates. The expressed goal of this thesis is to separate the adaptive mechanism from the content over which it acts. AE3 is very flexible, because it allows the developer to decide on the complexities of each model, and is not tied down to a learning based approach. AE3, as an incarnation of the multi-model, metadata driven approach, maintains explicit separations between the models over which it adapts.

Another factor of importance are the number of adaptive techniques available for use by each system. ELM-ART provides adaptive navigation, which cannot be leveraged well in the personalised news context, as it is tied directly to the content it is providing a navigation structure over. AHA! provides both adaptive navigation and adaptive presentation, as does AE3. The importance of having different adaptive axes available for use cannot be understated. Although adaptive navigation is not a good way of adapting personalised news, it is important that it is an avenue that is kept open for further research.

The domain model, is another important factor in the choice of an adaptive system to use. Placing the domain model and the adaptation model together, as AHA! does, provides overlay, but creates a dependency between modelled parts of the system. This dependency makes extracting or modifying adaptive logic difficult. ELM-ART uses a textbook like approach to modelling the domain, with chapters and subsections forming the basis of the domain model. AE3, with its flexible nature, allows the developer to once again, make a choice on how to model the domain.

In conclusion, AE3 would seem to be the best option to base an adaptive service on. AE3 provides good flexibility with its approach, good choice in terms of adaptive techniques available for use, and a good choice of how to represent the domain. One drawback of AE3 is that it may almost be too flexible in its instantiation. Giving the developer too much choice may result in a scenario similar to cognitive overload, although it is better to have too much flexibility than to have none. Flexibility allows the developer to provide a system best suited to the user, rather than suited to the constraints set by the system itself

3.3 Domain of News

The domain of news which was chosen for personalisation was the sport of Formula 1. This domain was chosen as it is an area of sport which doesn’t change remarkably during the course of the summer months unlike other sports such as football. By change, it is implied that drivers don’t move between teams, teams don’t change tyres, nearly everything about the domain is the same at the start of the season as at the end of the season. It is also a relatively small domain, with 20 drivers, 10 teams, and 19 races during the course of the year, thus making it a manageable domain to model manually. Even considering the size of the domain, it was still time consuming to construct the ontology. There is the opportunity to establish more classes, relationships between classes, and provide new instances which have a basis on previous Formula 1 championships which one may find in news about the domain e.g. Jenson Button becomes first British Driver since Damon Hill to win at Circuit X. The main drawback of such an opportunity is the time that goes into the construction of a fully structured ontology. Having said that, the major benefit is that you can have a historic view of a domain, which can be important for some users. It also gives the benefit of having more potential articles of interest as the domain has a temporal dimension added to it.

3.3.1 Modelling of the Domain

Modelling a domain of news with an ontology is a structured way of providing information about the domain. In modelling the domain of Formula 1, it is important to break the sport down into concepts which make up the sport. Concepts such as Teams, Drivers, Engines, for example, are concepts which form part of the interest a user could have within the domain. The concepts are represented as classes in OWL, and instances are created of each class e.g. Ferrari is an instance of a Team. The complete list of concepts used are Team, Driver, Engine, Chassis, Tyre, Team Principal and Team Technical Director. These concepts form the main interests a user could have of the Formula 1 domain. Relationships between the instances are modelled as relationships between the classes. Relationships provide the link between classes of interest. An example of this is a Driver who has the relationship "drives for" with a Team. The relationships may be semantically meaningful if the ontology developer decides so at the time of design. Now that instances of classes and relationships are available, it is possible to link instances with one another, to provide an overall view of the domain.

3.4 Framework Design

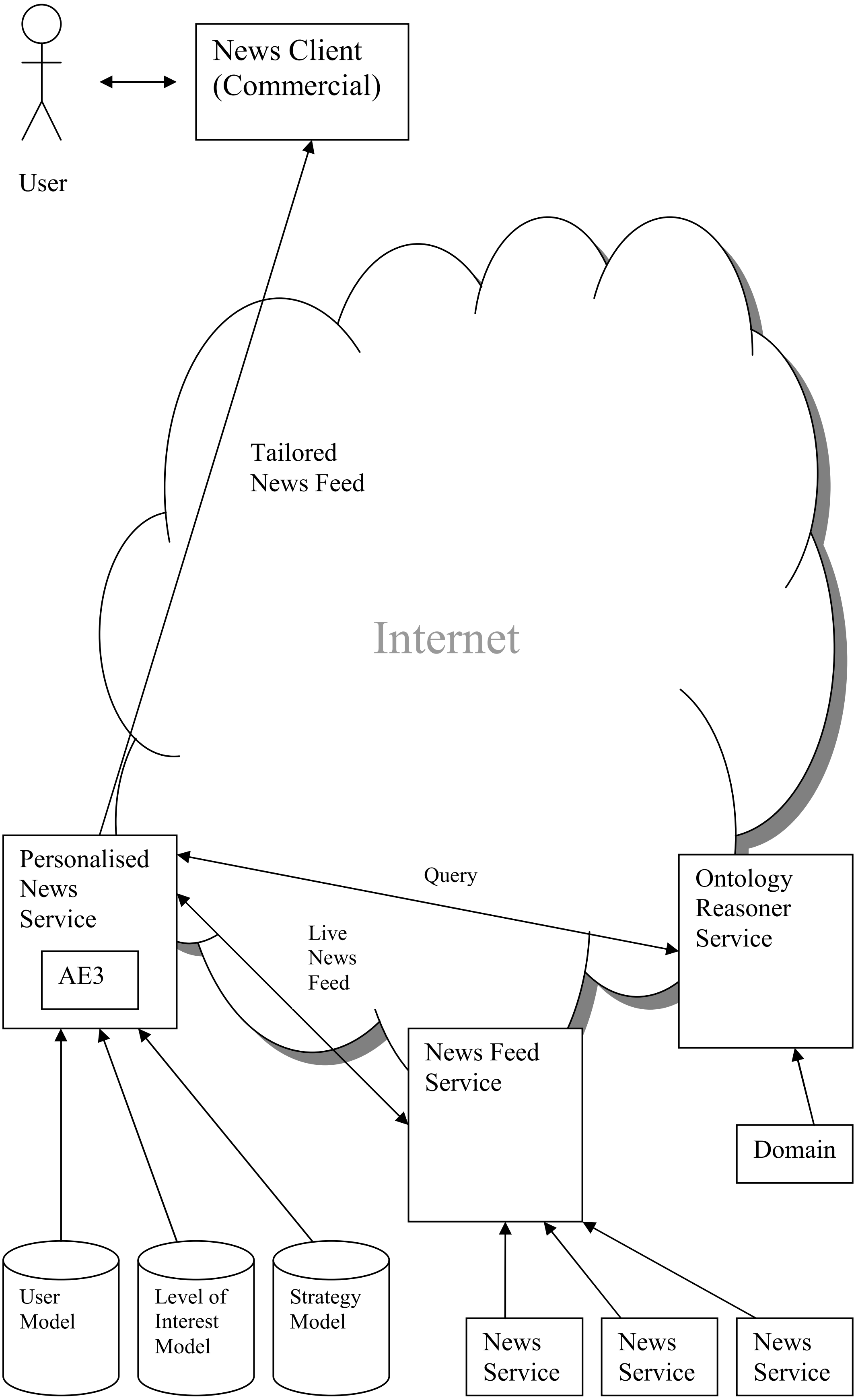


Figure 3 – Personalised News Service Architecture

The Personalised News Service comprises interactions with a number of different components. These components include the different models needed to provide an adaptive service on top of the AE3 engine, Web Services, and an RSS News Reader or client.

The Models are firstly loaded into the Adaptive Engine, before any work can be done with them. The Adaptive Engine forms the basis of the Personalised News Service. AE3 allows the multi-model driven approach, and the Personalised News Service is an extension of the Adaptive Engine. The Narrative Model, User Model, and Conceptual Models are all represented in XML, and the User Models and Conceptual Models are provided as internal instances in memory for easy manipulation of the data residing in them. The Narrative Model provides the adaptivity, as it takes in the User Model and the Conceptual Model, reconciles the two to provide queries, which can then be used to query the ontology available.

Once a query has been created, the Narrative calls an ontology reasoning web service, which provides a method of querying an ontology and provides information based on the query presented. An example query may ask for metadata about the drivers which drive for a particular team. The ontology reasoning web service provides metadata for each instance to the Narrative Model.

Once the metadata has been collected for all of a users interests, the next step is to compare this metadata with the information from a news source to establish which articles are of interest to a user. The Personalisation Service provides a method of creating a personalised RSS feed from a live news service, as designated in the narrative model. The personalisation service downloads the RSS from the designated news service directly, and sifts through the articles provided by the news service. Any articles that are of interest are passed back to the narrative model which can then provide the personalised feed on a webpage.

The final step, is simply for an News Client to access the page, and provide the content in a readable form for the user. As this is the purpose of a News Client, there is no need to modify or transform the content in any way.

3. 5 Service Design

3.5.1 Personalisation

The personalisation is performed by means of multiple, independent models, each of which has an important role within the personalised news service. These include the narrative model, which contains the strategy, the user model, which contains the users interests, the conceptual model, or level of interest mapping model and the content model. The conceptual model simply provides a mapping of a level of interest such as a high level of interest to relationships which may be of interest given that level of interest in a concept

3.5.1.1 Narrative Model

The Narrative model is where the adaptivity is placed in the system. The model performs the adaptation based on the information provided to it by other models. The Narrative Model will need information about the relationships which will interest a user depending on their depth of interest. This is provided by the conceptual model. The conceptual model is the model which contains the relationships of interest for each concept e.g. Team. If, for example, a user has a high level of interest in a concept, then the number of relationships that the user will be interested in that are associated with that concept will be high, and the type of relationship with other concepts will determine its relevance.

There is only one narrative model covering all steps involved when using the engine. The Model simply reconciles the user model with the conceptual model, providing information for a query to query the ontology. It then provides the query itself to establish what terms in news feeds would be of interest to the user. If the news service had multiple domains, it would be better to have many narratives which understand one domain, and a super-narrative providing a multiple domain news service.

3.5.1.2 User Model

The user model chosen is a simple one. Due to the domain in which the news appears, and the fact that there is only one domain of news which is personalised, there is no need for a complex user model. A simple user model such as the one in use provides more than enough information with which to personalise news feeds. The user model provides information such as a subject of interest (e.g. Ferrari), that subjects concept (Ferrari is a Team), and a level of interest in that subject (e.g. medium). The concept is provided in the user model as this provides knowledge for the model. The knowledge that Ferrari is a Team allows the narrative model to establish the relationships with objects which will be of interest to the user. This information is all that is needed to provide enough adaptation to the news feeds for a particular user.

3.5.1.3 Content Model

The content model is in the form of RSS feeds, which can be taken from an XML database, or wrapped as a web service for live feeds. This project will incorporate live feeds, as this will give the user the most up to date news, and will make the feed more appropriate to a users needs. The project will connect to ITV's RSS feed [43], and will use that feed as the source, inserting into the new RSS feed, any news articles which are deemed to be interesting for a specific user. Old news is not something that will be of interest to a user. RSS feeds have a temporal constraint which places an impetus on the content itself to be up to date. Content delivered to a user which is out of date is not of much use to the user. Due to the design of the system and the proposed idea to take live feeds and personalise them, this constraint should not be a problem.

3.5.1.4 Conceptual Model

The conceptual model provides a number of concepts, and depending on the level of interest of users, a list of the types of relationships which may interest them. A user having a low interest in a concept will have few (if any) relationships associated with that concept. Equally, a person with a high interest in a particular concept will have an interest in a reasonably large list of relationships attached to it. The relationships mentioned are relationships which are between two ontology concepts or classes. Intuitively, the higher the interest the user has in a given concept, the more numerous the relationships they are interested in will tend to be. For example, a user with a high interest in a team, will more than likely be interested in not only the drivers, but the engine, the chassis and the tyres, things which would not be of particular interest to someone with only a medium interest in a team. The conceptual model however, does not provide links to step through concepts. A driver may be associated with a team , as may be a Team Technical Director, but if there is no relationship property linking a driver to a Team Technical Director, there is no immediate way in which the two can be associated with each other. This is where the depth of interest can come into play. If the depth of interest is high for a particular concept such as a team, there is the possibility that when querying the ontology, one could step through relationships. Taking for example, a high interest in Ferrari, would correlate to an interest in Michael Schumacher, which would correlate to an interest in a championship rival such as Fernando Alonso, who has no link with Ferrari. Stepping through relationships can have a downside however, in that after stepping through maybe two or three relationships, everything can be related to everything else, providing a mass of interconnected instances.

3.5.1.5 Tailoring the News

Time is a contextual parameter which can be looked at. The duration of a news item can be determined by a number of factors, namely the area of interest (politics, sport, etc.), the subject of interest (Bertie Ahern, Manchester United, etc.), and of course the level of interest in a particular subject. The example of a team introducing a new front wing aerodynamic component for the next Grand Prix, will probably have a time range of interest lasting from the time of the announcement to the time of the next Grand Prix. After the Grand Prix, such a news item will not be of much interest to a user.

3.5.2 Ontology

There are primarily two methods of constructing ontologies, being manual creation and automatic creation. Manually building an ontology requires the identification of concepts and properties within the domain of the ontology. There is also the need to populate this ontology with instances of the classes (and possibly properties), which are generally temporal instances specific to the ontologies content domain. The primary type of ontology defined by manual ontology construction is a strict ontology, or an ontology rich with semantics.

The other method of ontology construction is automatic or semi automatic ontology creation, similar to the Mercureo System [13]. Semi automatic ontology construction can be implemented by trawling the ontologies hyperlinked web resources. This allows the derivation of relationships between concepts or instances of concepts. The relationships derived however, have little meaning associated with them, as the nature of relationships through trawling are difficult to obtain. This can result in what can be termed a Loose ontology, or an ontology with little semantic meaning between the relationships. It should also be noted that it is possible to derive a loose ontology from a strict ontology, by stripping away the meaning from the strict ontology. A loose ontology can be helpful in a domain which is small, and has a relatively small number of objects and instances. A content domain which had a large amount of instances and objects would benefit more from a strict type of ontology, as it would have more control over relating instances.

The ontology can then be considered in two forms, a Strict ontology, and a Loose ontology. The Strict ontology provides meaning for things, such as a driver is a sub class of a person, as is a team technical director. It also gives semantically meaningful relationships between classes, such as “team A has driver B”, or “engine C is built for team D”. The strict ontology provides a rigid, yet easily understood, description of the formula one domain.

The Loose ontology strips the meaning from the Strict ontology, leading to a mass of things and things they are related to. Instances are provided, but there is nothing to differentiate a team from a driver, as they are both just things, albeit with different names. The semantically rich relationships are also removed from the ontology, leaving a simple “thing A is related to thing B” relationship.

For the evaluation of this project, there are two ontologies developed. One ontology provides a domain model which is semantically rich in meaning, sometimes referred to as a Strict ontology. The other ontology provides a domain model which has no semantic meaning, sometimes referred to as a Loose ontology. The Strict ontology has information about the domain, such as knowing the difference between a driver and a team, and what the relationship between to two is. The Loose ontology however, has no knowledge of the domain, and does not know the difference between the driver and the team, and simply knows the two are related, but not how they are related. In evaluating this project, the aim is to establish if there is any difference between the results provided by both ontologies.

The ontologies don't take into account relationships which may come easily to humans, such as "used to drive for", "team based in" or "was winner of". The ontology instance is, like most ontology instances, temporal. Relationships such as the ones described above are deemed unnecessary, as they would lead to more relationships and more instances than are necessary for the purpose of the project. For a commercial system however, these may be of interest.

One important thing to note about the ontology is that the classes and relationships are not likely to be subject to a temporal constraint. In the sporting domain, the fundamental principals of each sport are likely to change slowly over time. By this it is meant that for example, twenty years ago in Formula 1, there were still drivers who drove for teams, team cars had to have tyres, an engine and a chassis etc. although the instances (and possibly number of instances) would of course be different. In this sense, ontologies are an effective way of describing a domain, especially in terms of sport, each of which generally has a specific structure with which to model.

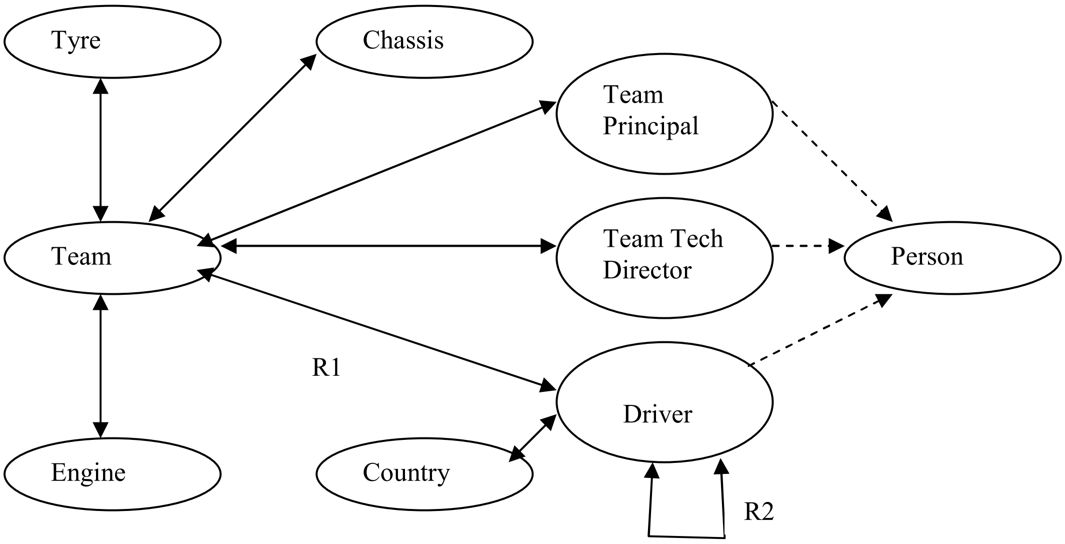


Figure 4 – Ontology Classes and Relationships

Figure 4 shows the classes provided by the strict ontology and which classes are related to each other. The broken line from person to the Driver, Team Principal and Team Technical Director imply that the three classes are a subclass of the class

Person. OWL provides an object oriented method for looking at a domain and creating a model for that domain. The relationships between classes are symmetric, for example, if a driver drives for a team, then the team employ that driver. The loop back arrow on the driver class simply means that a driver can be related to another driver, and there are two ways in which this can happen. A driver can be a team mate of another driver, and can be a competitor of another driver, and in some cases, can be both. Transitive relationships are not of much use in this ontology, but may be useful in other scenarios. A transitive relationship would be along the lines of "A related to B, B related to C, therefore A related to C". Looking at the two relationships R1 and R2 in Figure 4, the two are symmetric. R1, which is from driver to team implies that a Driver "drives for" a Team, but also that Team "has driver" Driver. Instance examples of such a relationship are Michael Schumacher "drives for" Ferrari and Ferrari "has driver" Michael Schumacher. R2 provides one of two possible relationships, Driver "has team mate" Driver, and/or Driver "is competitor of" Driver. Instance examples of this relationship are Michael Schumacher "has team mate" Rubens Barrichello and Michael Schumacher "has competitor" Fernando Alonso.

3.5.3 News Feed Capture and Personalisation

The news feeds needed are captured from a live feed. This feed is simply downloaded, and parsed, and when parsed is reconciled with information provided by the personalised news service. Once this information is reconciled, a personalised feed is produced, and is provided to the personalised news service by the news feed web service.

3.6 Modelling the User

3.6.1 Separating Interests from Level of Interest

Separating the interests from the level of interest provides independence for the user model. This independence can be very useful when presented with users who would define levels differently, and also when presented with different domains. For example, User A's perception of medium, may be different to User B's perception. By having the interests separated from the level of interest, there is the opportunity to provide users with their own perception of a level of interest. Quantifying this may be something along lines of User A perceiving that medium interest in a Team would include knowledge about all of the team, from the sporting front, to the political and administrative front to the management front. User B may not perceive this to be part of the information garnered from a medium interest in a team.

Also, taking into account different domains, a high interest in a Football Team and a high interest in a Formula 1 team would not give the same type of information, or have similar relationships from the team to related concepts. The level of interest would be independent of the domain, but the level of interest (or conceptual) model would contain information regarding whichever sport was being provided for.

3.6.2 Dynamism of User Modelling

The user model would change little in a multiple domain sports news service, and is highly scalable. Changes may include the addition of a type of sport, so as to reference a domain specific narrative (e.g. Formula 1, Football, Athletics). The ability to provide a multiple domain news service would prove a little trickier, as the user model for politics or entertainment could be different to the one in use for sports. The dynamism of the model is of interest when considering how to model the user. The news itself, being in the Formula 1 news domain is relatively dynamic. Most of the time, different drivers win races, different teams win races, and the news differs in a day to day context. The other dynamic quality the user model must take into account is the relationship between the users interests and the content domain. A user may receive an article based on having a high interest in a particular concept, whereas having a medium interest would not have produced that article.

3.7 Summary

This section describes design issues which have come under scrutiny with respect to the goals of the thesis. The AE3 System is incorporated into the design, as it enables the separation of the adaptive logic, or narrative model, from the content model. Ontologies are used as the basis of news domain representation, as they do not require the mining of information to provide a domain model. The AE3 System also allows for an extensible system, which allows other axes of adaptation to be taken into account. AE3 provides independent reusable components, which provide the basis for the Personalised News System.

4 Implementation

4.1 Introduction

This chapter will discuss the implementation of a Personalised News Service, as described in the design chapter. This chapter will look at each aspect of the design, from the service oriented architecture, through personalisation to the final result of a personalised news feed for a user. Technologies will be looked at which help implement such a service.

4.2 Modelling the Domain

The modelling of the domain is performed using a semi automatic ontology creation tool Protégé, which builds ontologies in the Web Ontology Language OWL. OWL provides the creation of classes to represent objects or concepts in a domain. OWL allows the ontology developer to determine themselves the complexity, and the structure that will be needed for the building of the domain model. In providing a structured methodology for creating classes, relationships, and instances of classes, OWL proved a good choice to develop the domain model in.

OWL provides semantic meaning for all classes created, and allows them to be disjoint from one another i.e. no two classes could be the same e.g. a Team was not the same as a Driver. The relationships provided by OWL were semantically rich, and provided a view similar to a domain experts' view of the domain. OWL, if chosen to be so by the developer, also allows little or no semantic meaning between classes and relationships, as shown by the loose ontology, which was built with the same tool. It has the same structure i.e. 10 teams, 20 drivers etc., but has no classes disjoint from one another and all relationships being a simple "is related to" relationship.

The domain was modelled according to information provided by the ITV Formula 1 site [44], and comprised of the following classes: Team, Driver, Team Principal, Team Technical Director, Engine, Chassis, Tyre and Country. Relationships were defined between classes which merited relationships, and instances were provided as per the site mentioned above.

The following short piece of OWL code, represents the Class TeamTechnicalDirector and the code shows that it is a subclass of Class Person, and that it is disjoint with TeamPrincipal and Driver, who are other classes. If classes are disjoint, it means they cannot be the same as one another. Therefore, a Team Technical Director is not the same as a Driver, and both are not the same as a Team Principal.

```
<owl:Class rdf:about="#TeamTechnicalDirector">
     <owl:disjointWith rdf:resource="#TeamPrincipal"/>
     <rdfs:subClassOf rdf:resource="#Person"/>
     <owl:disjointWith rdf:resource="#Driver"/>
</owl:Class>
```

The next piece of code shows the relationship "hasDriver" and is explained as follows. The range of the relationship, is Driver, meaning that the right hand side of the relationship will always have an instance of the Driver Class. The domain of the relationship, is Team, meaning that the left hand side of the relationship will always have an instance of the Team Class. If looking at the relationship in English, one would see Team "hasDriver" Driver. The inverse of property, implies that the relationship is symmetric. While Team "hasDriver" Driver, the inverse of the relationship is Driver "drivesFor" Team.

```
<owl:ObjectProperty rdf:ID="hasDriver">
     <rdfs:range rdf:resource="#Driver"/>
     <owl:inverseOf>
          <owl:ObjectProperty rdf:ID="drivesFor"/>
     </owl:inverseOf>
     <rdfs:domain rdf:resource="#Team"/>
</owl:ObjectProperty>
```

This final piece of OWL code presents instances of the above piece of OWL code. The Team Ferrari, has two "hasDriver" relationships, one for Driver MichaelSchumacher, and the other for Driver RubensBarrichello. The label applied to the team Ferrari is for the purpose of extracting information regarding the Team.

```
<Team rdf:ID="Ferrari">
     <rdfs:label>Ferrari</rdfs:label>
     <hasDriver rdf:resource="#RubensBarrichello"/>
     <hasDriver rdf:resource="#MichaelSchumacher"/>
</Team>
```

The above pieces of code represent parts of the strict ontology. The following pieces of code will represent the loose ontology, and will show the difference between the two.

```
<owl:Class rdf:ID="Thing"/>
<owl:SymmetricProperty rdf:ID="isRelatedTo">
<owl:inverseOf rdf:resource="#isRelatedTo"/>
<rdfs:range rdf:resource="#Thing"/>
<rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#ObjectPropert
y"/>
<rdfs:domain rdf:resource="#Thing"/>
</owl:SymmetricProperty>
```

It can be seen here, that the OWL Class Thing, is being represented. Every concept in the loose ontology is an instance of the Class Thing. The only relationship which is available is the "isRelatedTo" relationship. It carries no semantic meaning, and provides little information about the classes it relates. This is because the range (the left hand side) and the domain (the right hand side) of the relationship are the same Class. A relationship would look like this: Thing isRelatedTo Thing. This marks quite a difference between the semantically rich Strict ontology.

```
<Thing rdf:ID="Ferrari">
<rdfs:label>Ferrari</rdfs:label>
<isRelatedTo rdf:resource="#MichaelSchumacher"/>
<isRelatedTo rdf:resource="#RubensBarrichello"/>
</Thing>
```

The piece of OWL code above shows the Loose ontology's version of the Strict ontology's view of the Ferrari concept, and the two drivers which are associated with that team. The difference is that Ferrari is now merely a Thing, rather than a Team, and the relationship provides no meaning, as it is known that the two are related, but it is not known how the two are related.

4.2.1 Protégé

Protégé is an open source tool, developed at Stanford University, which can be used for the semi automatic creation of ontologies. Protégé is based upon Java [3], and serves as a platform which can be extended to include media, storage formats and graphical depictions. Protégés API allows for it to interface with other applications, and provides a "plug-and-play" environment for the development of applications and speedy prototyping.

4.3 Framework Implementation

The Framework was implemented as per the design in the following way:

4.3.1 RSS News Reader

The RSS News Reader in use was Feed Reader [2]. Feed Reader is an RSS client which simply connects to the feed provided by the user, and presents it to the user in an aesthetically pleasing form, by parsing the tags in some predefined way. The User was required to download Feed Reader, and set up three different feeds, one for the Strict ontology, one for the Loose ontology, and finally one for the source content which contained all the news provided for the personalised news service. See Appendix VII – Screen Shots of Feed Reader using the PNS.

4.3.2 Personalised News Service

The Personalised News Service provided, is an extension of the Adaptive Engine AE3. The Service provides personalised RSS news feeds for users, and the RSS news Reader is required to connect to a Java Server Page which holds the personalised feed for the user. This is done over HTTP. The adaptive engine sits on a Tomcat Web Application Server, which allows the creation of the personalised feeds through the Java Server Page.

The following is a step by step flow chart as to what the narrative model does in providing the personalised feed for the user

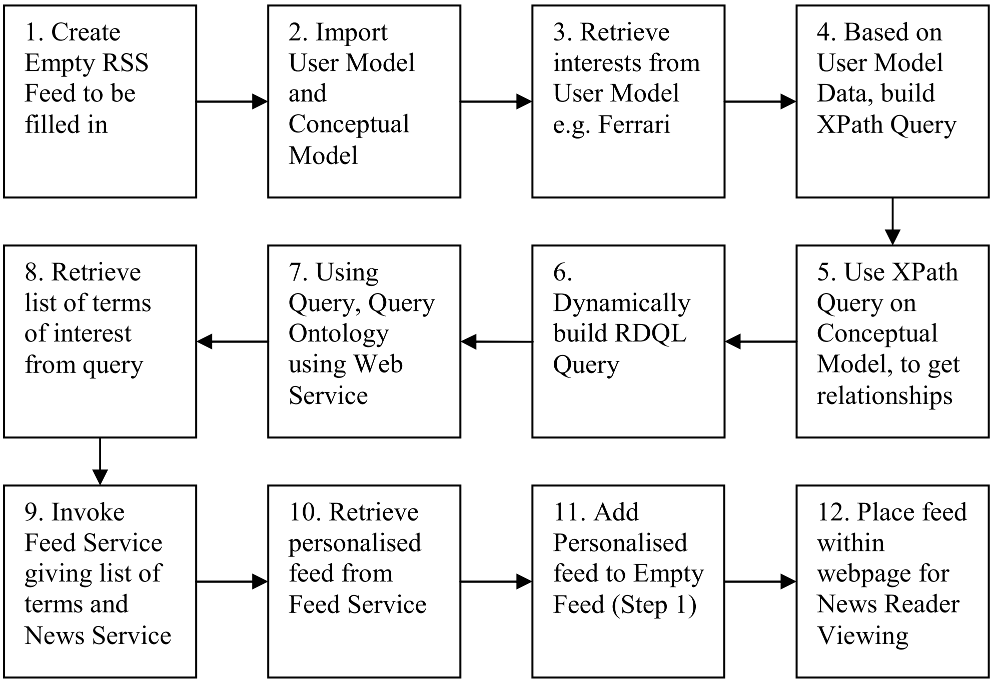

1. Create Empty RSS Feed to be filled in
2. Import User Model and Conceptual Model
3. Retrieve interests from User Model e.g. Ferrari
4. Based on User Model Data, build XPath Query
5. Use XPath Query on Conceptual Model, to get relationships
6. Dynamically build RDQL Query
7. Using Query, Query Ontology using Web Service
8. Retrieve list of terms of interest from query
9. Invoke Feed Service giving list of terms and News Service
10. Retrieve personalised feed from Feed Service
11. Add Personalised feed to Empty Feed (Step 1)
12. Place feed within webpage for News Reader Viewing

Figure 5 – Narrative Step by Step Flowchart

The personalised news service interacts with two web services, an ontology reasoning web service, and a news feed capture and personalisation web service. This is done through the use of Web Service calls. The calls are invoked using JESS, in the narrative model. See Appendix II – Narrative Models for the code.

4.3.3 Ontology Reasoning Web Service

The Personalised News Service creates an RDQL query, which is passed over SOAP to the ontology reasoning web service. The RDQL query is similar to a SQL query, and queries a given ontology, asking for information about classes. The Web service queries the ontology, receives the desired information, and this information is passed back to the personalised news service. The information passed back is a list of items of interest to a user.

The following is a work flow diagram of what the Ontology Reasoning Web Service does in providing information for the Personalised News Service

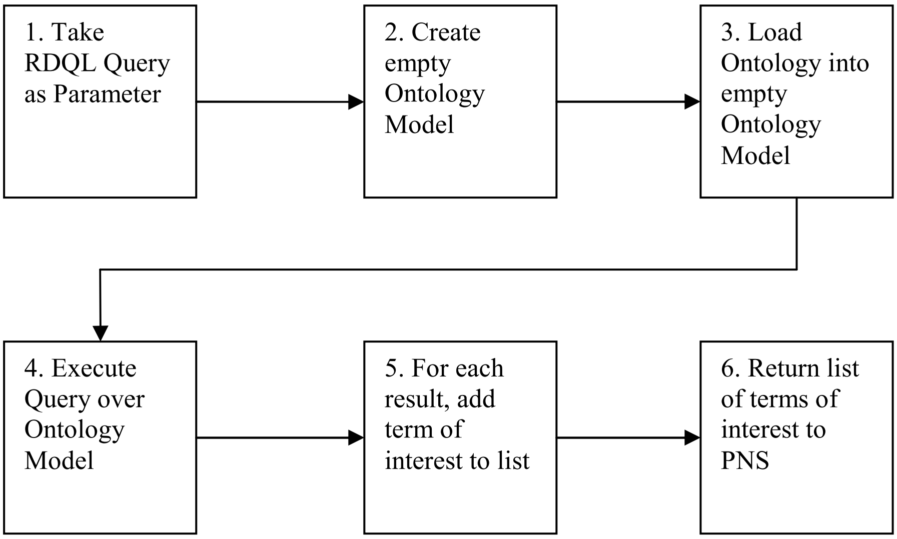

1. Take RDQL Query as Parameter
2. Create empty Ontology Model
3. Load Ontology into empty Ontology Model
4. Execute Query over Ontology Model
5. For each result, add term of interest to list
6. Return list of terms of interest to PNS

Figure 6 – Ontology Reasoning Web Service Flowchart

The Ontology Service takes the RDQL Query as a parameter in the method invocation. Creating an empty ontology model in memory allows the OWL ontology to be referenced, once loaded into memory, creating a quick and easy way to query the ontology. Once the ontology is loaded into memory, the query must be invoked across the ontology, and the results obtained. The results are gathered, and added together in the form of a list, which comprises terms of interest which will be presented to the other web service for personalisation of web feeds.

4.3.4 News Feed Web Service

The News feed web service is a service which is passed two parameters, a list of items of interest to a user, and a news service from which to base it's personalisation on. The news service is simply a link to a live RSS feed. The news feed web service then filters the news based in the list of items, including articles of interest, and discarding articles which are deemed not interesting to a user. This is compiled into a new RSS feed, which is provided to the personalised news service.

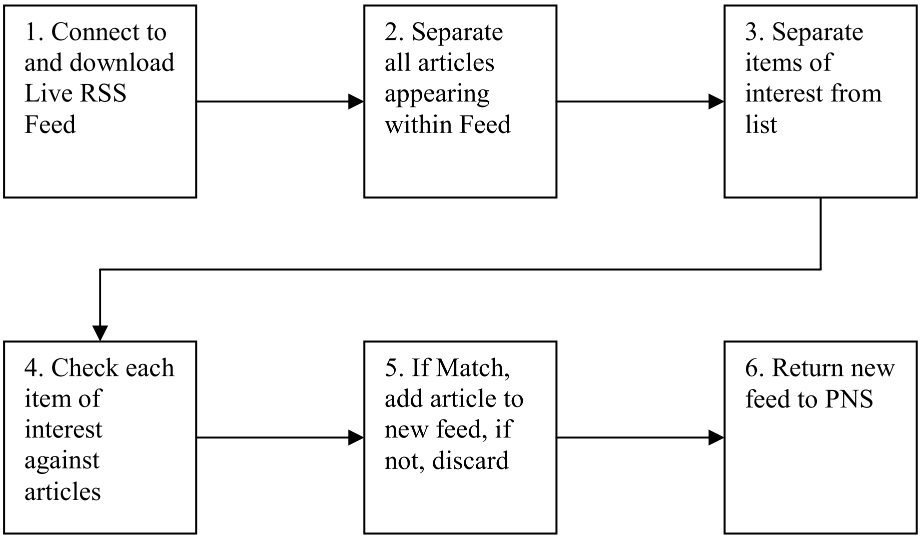

1. Connect to and download Live RSS Feed
2. Separate all articles appearing within Feed
3. Separate items of interest from list
4. Check each item of interest against articles
5. If Match, add article to new feed, if not, discard
6. Return new feed to PNS

Figure 7 – News Feed Capture Web Service Flowchart

The news feed capture and personalisation web service must first connect to and download the feed specified by the Personalised News Service. All articles must then be separated, as they are unique articles referencing a single topic within the domain. All items of interest must be detached from the list so there is two sets of variables, one set of articles and one set of terms of interest which could appear within those articles. The checking process is then performed, where articles are searched for the terms of interest, and if the terms appear then the article is of interest to the user and is added to the new feed. Any terms which do not match with an article are simply discarded as they are of little or no interest to a user. Once the checking process is complete, the personalised feed is returned to the personalised news service, which can then provide this information as an RSS feed for a user.

4.4 Service Implementation

The Service is implemented using various different technologies, depending on the part of the system being used. As the system can primarily be broken down into three different components, this is how the service implementation will be dealt with. The three separate components are the personalised news service, the ontology reasoning web service and the news feed capture and personalisation web service

4.4.1 Personalisation

In order to personalise content, there must be the facility to personalise that content. Adaptive Engine 3 provides that facility, using multiple models. The following models are put to use in the personalised news service, the Narrative Model, the User Model, the Content Model and the Conceptual or Level of Interest Model. All of the models are represented in the same way, using XML.

4.4.1.1 Narrative

The Narrative Model, which was the model responsible for the adaptation, has snippets of JESS code in the XML. The JESS code is responsible for performing the adaptation, as it checks through the other models and performs operations on the data members within those models.

JESS provides the narrative model with a lot of functionality, through its own implementation and through the building of custom functions. The custom functions allow the manipulation of models to provide an adaptive service. Custom functions such as the call-web-service function provide a much needed outlet for the engine, as services can be called to manipulate data which the adaptive engine would not be able to do. Another custom function is the xpath-query-model function which allows a model to be queried via XPATH.

4.4.1.1.1 Querying the User Model and the Conceptual Model

The User Model and the Conceptual model are reconciled at runtime, but there is a vital ingredient needed to provide the right output. XPATH provides a method for extracting information from XML files. XPATH allowed the comparison of elements within both the user and concept models to provide the dynamically filled parts of an RDQL query. The XPATH query is housed in a JESS function call.

4.4.1.2 User

The User model, is simple in its instantiation. One Basic Interest in the user model, which may have many interests, would look like this

```
<user>
  <interest>
```

```
    <name_of_interest></name_of_interest>
    <depth_of_interest></depth_of_interest>
    <concept_of_interest></concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>
</user>
```

The name of interest, is the name which the concept will have, e.g. Ferrari, Michelin etc. The depth of interest, will have one of three values, low, medium, or high, as non interest in a concept results in it not being included at all in the user model. Concept of interest is the class the instance belongs to, so concept for Ferrari is Team, concept for Michelin is Tyre. The source of interest is something which was implemented in the system due to time constraints. The source of interest is intended to provide the user within information based on their chosen source. This could be for reasons such as better coverage of the domain, more in depth knowledge of the domain or other such strengths of a news service. The user model comprises of one or more interests, and provides an accurate model of the user. See Appendix IV – User Models for an example User Model

4.4.1.3 Content

The content model is a model which is built by the provider of the RSS feeds. The RSS schema is the same for all RSS feeds, as there has to be a consistent schema for the different RSS readers. In providing a consistent schema, extracting information from the content model should be the same regardless of the content domain. This provides a reusable method for extracting information from feeds to produce a personalised end result.

The RSS snippet below, is one item of information which will be chosen for inclusion based on the user model. The structure for RSS is the same regardless of the provider of the RSS, allowing a methodical way of presenting news information.

```
<item>
<title>Williams chases tyre solution</title>
<description>
Sam Michael hopes that Williams' strenuous efforts have
cured their tyre problems.
</description>
<link>http://www.itv-
f1.com/Controller.aspx?PO_ID=33877</link>
<category>News Article</category>
<pubDate>Sun, 28 Aug 05 14:25:35 GMT</pubDate>
</item>
```

Each RSS news article is contained within an item tag. The item tag contains five children, which are needed for an RSS News Reader to parse properly. The title tag is fairly self explanatory in that it presents the user with a very short description of the news article. The description tag expands the title a little. The link tag provides a link to a page containing the full story, the category tag provides an insight into the type of article, and the pubDate tag provides the user with the time the news story went online, so as to determine whether the news is old or new. In general however, news that is more than a couple of days old is normally discarded from the feed as it is old news.

4.4.1.4 Conceptual

The Concept Model is, like all the other models, written in XML. This model provides the narrative with a view to which relationships are of interest to objects, given a specified depth of interest in that object. Taking a low interest in an object, it is inferred that anyone with a low interest in an object, will want to know only about that object, and none of the relating objects. This may not be the case, but it is probably close to what a user would imply by saying they had a low interest in a certain object. Similarly, it can be inferred that a medium interest in an object would correspond to an interest in that object, and a couple of other objects related to it. Again, for a high level of interest in an object, a user would more than likely be interested in a relatively large amount of related objects, and the extra relationships would be more abstract and fine grained than the relationships provided by a medium interest in the object

An Example of the Conceptual Model is provided below and shows how the mapping from a level of interest in a concept corresponds to the types of relationships which are likely to satisfy a user with that type of interest. The conceptual Model is only used in the Strict ontology, as it can provide a good mapping. It is not used in the Loose ontology as the Loose ontology uses a hop count over relationships rather than having an unneeded conceptual model.

```
<interest>
  <concept>Driver</concept>
  <interest_type>
    <type>high</type>
    <relationship>drivesFor</relationship>
    <relationship>hasTeamMate</relationship>
    <relationship>hasCountryofBirth</relationship>
    <relationship>isCompetitorOf</relationship>
    <relationship>racesOnTrack</relationship>
  </interest_type>
  <interest_type>
    <type>medium</type>
    <relationship>drivesFor</relationship>
    <relationship>hasTeamMate</relationship>
  </interest_type>
  <interest_type>
    <type>low</type>
  </interest_type>
</interest>
```

The mapping can be explained as follows. For a Driver that a user might be interested in, if the user has a low interest in a driver, then it is likely that they will only be interested in the driver, and will not be interested in anything related to the driver. Therefore, there are no relationships provided for a low interest in any concept, such as Driver. For a medium interest in a Driver, it is likely, that the user will be interested in the team that the Driver drives for, and the Drivers team mate. This may not be the case, but is a result of careful thought and scoping of interests. Finally, a user with a high interest in a driver, will be interested in all of the relationships provided by the medium level of interest, plus some more, including Drivers country of birth e.g. when the Drivers home Grand prix is coming up, competitors of the Driver e.g. if the Driver is challenging for the Championship information about rivals would be of interest to a user. See Appendix III – Conceptual Model

4.4.2 Ontology Reasoning

The ontology reasoning web service allows for the querying of ontologies. The web service requires the use a query language, and RDQL is the chosen query language. RDQL is similar in ways to the database query language SQL, but the objects they query are quite different. An example query may ask for metadata about the drivers which drive for a particular team. The ontology reasoning web service provides metadata for each instance to the Narrative Model.

The RDQL query for selecting information about the BAR team such as who drives for the team, who is the team principal and who is the team technical director would look something like this.

```
SELECT ?x WHERE (?x ?predicate "BAR")
  (?x so:hasDriver ?ob1)
  (?x so:hasTeamPrincipal ?ob2)
  (?x so:hasTechnicalDirector ?ob3)
  (?ob1 rdfs:label ?ob1Label)
  (?ob2 rdfs:label ?ob2Label)
  (?ob3 rdfs:label ?ob3Label)
USING so FOR <http://www.owl-ontologies.com/unnamed.owl#>
```

Having queried the ontology, and obtained the necessary information, the web service supplies the narrative with the information required. The reasoning over the ontology is done with Jena

4.4.2.1 Jena

Jena [40] provides an open source Ontology reasoner, and can be used with OWL DL. Jena allows the checking of consistency of ontologies, classification of ontologies, and answering a subset of RDQL queries. It is a semantic web framework, and most importantly supports all OWL DL constructs, as OWL DL is the language with which the ontologies were developed.

4.4.3 News Feed Capture and Personalisation

The News Feed Capture and Personalisation service is independent of the news service provided and the terms of interest provided to it. The news service could personalise information from any feed provided to it, and on any list of interests provided to it. The web service receives information using SOAP, and passes information back using the same mechanism.

4.5 Technological Architecture

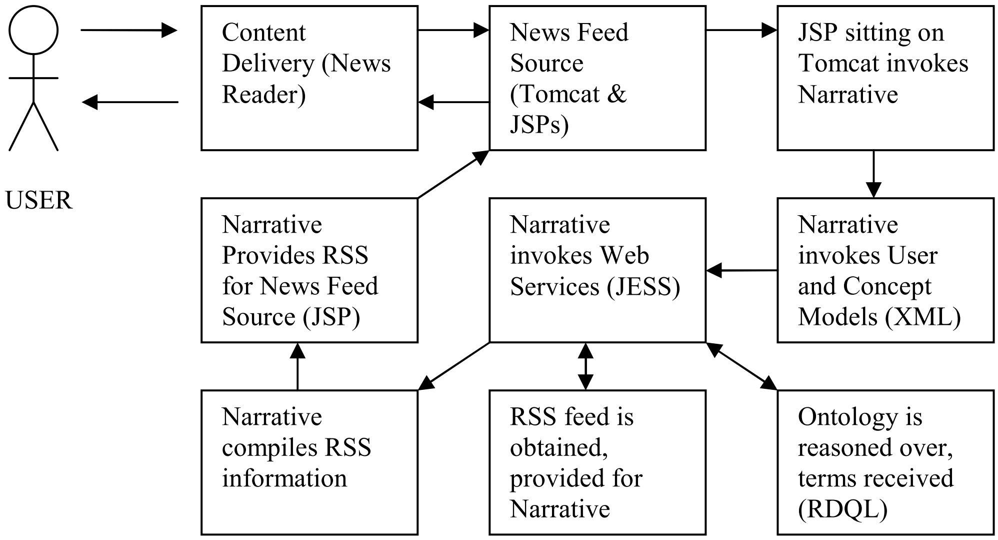

USER
Content Delivery (News Reader)
News Feed Source (Tomcat & JSPs)
JSP sitting on Tomcat invokes Narrative
Narrative Provides RSS for News Feed Source (JSP)
Narrative invokes Web Services (JESS)
Narrative invokes User and Concept Models (XML)
Narrative compiles RSS information
RSS feed is obtained, provided for Narrative
Ontology is reasoned over, terms received (RDQL)

Figure 8 – Technological Architecture

Content delivery is provided by the News Reader client. The client connects to the page providing the personalised news feed, and uses the preformatted tags to provide the information in such a way as it would do with the original feed. The News Feed source is provided by a Java Server Page, which sits on top of the Tomcat Web

Server. The Tomcat Web Server provides an environment for the Adaptive Engine 3 to run on, and allows the JSPs to perform the functions which will provide the personalised feed for the user.

The Narrative model is loaded into the adaptive engine, and is invoked within the JSP. The Narrative itself then invokes both the user model and the conceptual model provided in the adaptive engine. Once the appropriate information has been garnered from both of these this information is passed through JESS Custom Functions to perform operations on the data. The ontology is reasoned over with the Resource Description Query Language, and the feed is obtained and personalised using another Web Service available to the system.

Once the Narrative has received the information, which is in RSS format, it presents it to the Java Server Page which annotates its source with this information, providing a news feed for the News Reader to produce.

4.6 Summary

This chapter introduces the implementation of the Personalised News Service, with respect to the goals of the thesis. In using the AE3 System, the adaptive logic and content model can be completely separate from each other and do not have to rely on each other. The OWL ontologies provide a structured, domain experts view of a domain which does not require mining of information to establish a domain model. The incorporation of Web Services and reusable components allows the system to become more extensible though the addition of services which perform different tasks.

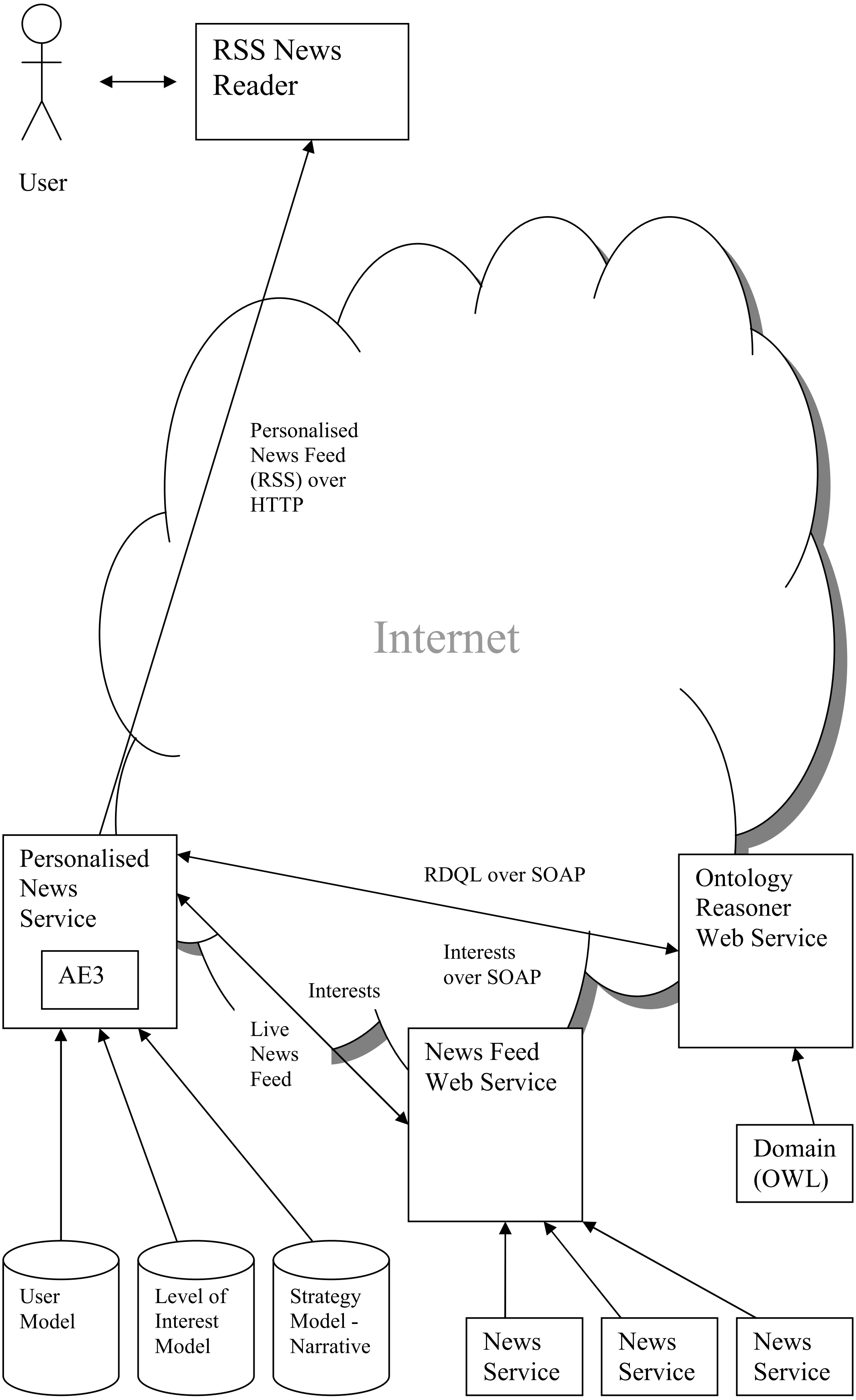


Figure 9 – Framework Implementation

5 Trial and Evaluation

5.1 Introduction

The benefits of the ontology driven approach to domain modelling can be determined through investigating user satisfaction with the service against the backdrop of the relative costs of developing strict and loose ontologies. The evaluation of the ontology driven approach asks if users were happy with the content they received, if they received the content they expected, and did they notice much of a difference between the two ontologies. This satisfaction is paramount to the project, as if the users are not happy with the service provided, then it is likely that they would not use such a service again. This evaluation is carried out by means of a paper based questionnaire, determining aspects of the system the user was happy with, or where the system could be improved. This evaluation is presented in section 5.2 User Satisfaction.

The evaluation of the service oriented approach is one which cannot be measured purely by user satisfaction, but by looking at the different components and comparing their qualities with respect to independence and reusability. This aspect of the evaluation uses the State of the Art as a background against which comparison may be made. In evaluating the service oriented approach one must consider how the project would be feasible using a different approach, those proposed by systems such as AHA! and SeAN. The evaluation of the service oriented architecture is provided in section 5.3.

5.2 User Satisfaction

Users are the most significant stakeholder of any software system, and it is paramount that they are satisfied with what the system has to offer them. This is certainly even more important in terms of adaptive hypermedia systems. In the case of a personalised news service, the evaluation may take into account the news a user was offered, the news a user was not offered, previous experience with news services, and how they thought the system performs overall. This section describes the user trial and evaluation, beginning with an overview of the trial, continuing with figures produced from the evaluation questionnaire, and concluding with key findings from the questionnaire.

When looking at the user trials it is important to note that there are three characteristics in which a system must take into account. These are the level of interest, the domain specificity, and the structure of the domain. The level of interest corresponds to a depth of interest within a particular domain. A user may have a very high, or broad interest in a domain, and this would affect the news they would receive. It is likely that such a user would like detailed articles regarding information within the domain. A user could also have very little interest in the domain of news. Intuitively, it is likely that such a user will only want a general overview of the content provided.

The domain specificity is also something which must be taken into account. If the domain is focused on one particular area, then the likelihood is that there will be users who have a broad interest in the domain. If the domain is not focused however, it is likely that there is not enough interest in the domain, to warrant having a specific domain model. Users will only have an interest in the general information about the domain.

Finally, the structure of the domain is important. The domain model can provide a structured, semantically meaningful view of a domain, or can simply be classified as a taxonomy of inter-related things.

Take for example, a high interest user, in a focused domain. This user will be interested in niche topics, or specific details about the domain. Because the domain model is focused on a particular domain, these niche topics should be available for the user to obtain. Adding to this, a structured domain model, which has rich semantic metadata and meaning within the domain model, the information presented to a user can be even more fine tuned towards their level of interest. On the other side of the coin, if one were to look at a low interest user, using a system with a focused domain, and a semantically rich domain model, they may receive information that is too detailed, or too specific for their particular interests. This can cause user dissatisfaction within a system. There needs to be a balance within the three dimensions, with a system that provides high level information for low interest users, and low level, detailed information for high or broad interest users.

5.2.1 Trial Setup

The trial involved 9 users, using the services for two weeks. The users first filled out a questionnaire which contained various interests, including concepts such as teams and drivers from the domain of Formula 1. The results of this questionnaire were then fed into user models, which modelled the interests supplied by the users. The users were required to setup a news client, which would parse the feed being supplied, and the Feed Reader news client was chosen for users.

The two ontologies were disguised as two models, and the users did not know how the two models were different. Model One referred to the Strict Ontology, which has semantic meaning. Model Two referred to the Loose Ontology, which has little or no knowledge about the domain. It can be inferred that personalisation for model one, means something different to personalisation for model two. This is because it is likely that a user will get the same amount or more information from model two than they will from model one, due to model two having little knowledge about the domain. It is this difference, and users perceived difference between the two models that provides the most interesting results.

5.2.2 Overview

Each of the users was interviewed with the same paper based questionnaire, allowing them to evaluate the personalised news service they had been using. The questionnaire comprised of 12 multiple choice questions, and an open comments section. The questions were divided into sections, but these sections were not made explicit. Through the questionnaire, the evaluation tries to examine the following items –

- Use of News Services prior to the use of the personalised news service
- Comfort in using News Services
- Comparison of the results the user received with the interests they had given in the pre-use interest questionnaire
- Satisfaction with the news provided by the service
- Comparison of the news received with the news expected

- Satisfaction with the way the user was modelled within the system (did it accurately reflect interests)
- Comparison of the two ontologies and in particular, in the results provided by both ontologies
- Additional features which could be added to enhance the users experience of the system

5.2.3 Use of News Services

The goal of determining a user's prior knowledge and comfort with using news services on the web was to see how familiar the users would be with such a system. It is not all that important that the user is very familiar with using news services on the web, but it can give an insight into how many people will have no problem using a system such as the personalised news service. If a user has no experience in dealing with online news services then this may explain the reason for answering questions differently to other users of the system who have experience with news services.

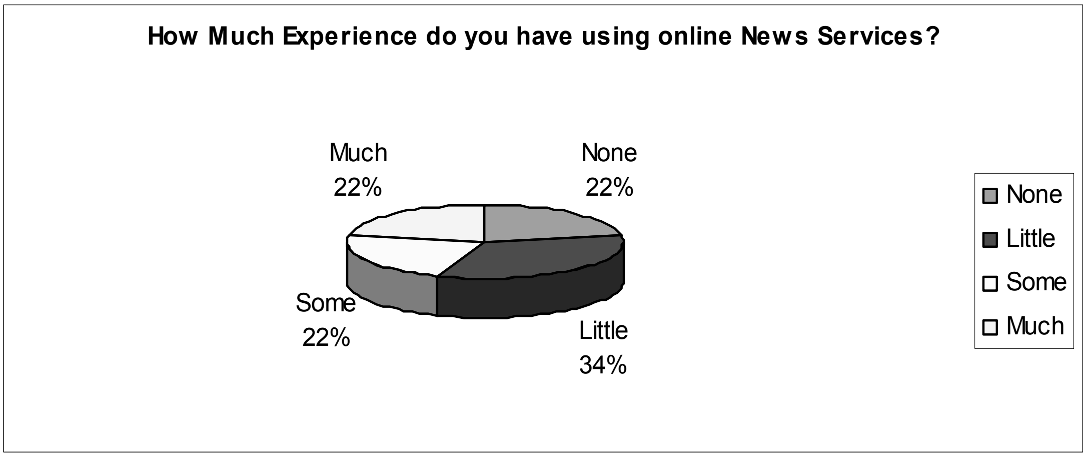


Figure 10 – Users Prior Experience with online News Services

Looking at the above graph, it can be seen that the users' experience with online news services was a little diverse, but spread pretty evenly across the board. This may imply that some users may have a little bit of difficulty in using the system, but it is highly unlikely as they are all comfortable using computing facilities.

5.2.4 Comfort with using News Services on the Web

A user should feel comfortable when using the news service provided to them. By comfortable, it is meant that the users feel at ease using the service. If the user does not feel comfortable then it can reflect badly on the users experience when using the system. In gauging the users comfort when using news services on the web, an insight can be gained into why a user might not be happy with the end results provided by the system.

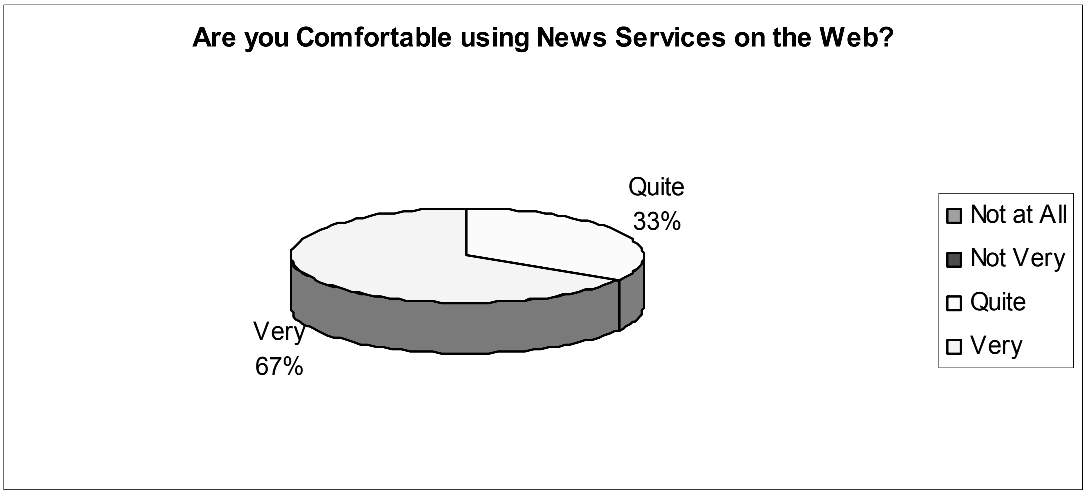


Figure 11 – Comfort using News Services on the Web

Following on from Figure 10, it can be seen in Figure 11 that users in general were more than capable of navigating a News Service online. This should lead on to none of the users having very much technical difficulty throughout the course of the evaluation. The reason the results were so favourable in this case is that all of the users are people who use computing facilities on a daily basis. They are students in the research area of computing, and therefore should be fairly at ease when using a new system such as the one provided.

5.2.5 Comparison of Results with Interests Questionnaire

It is important for any personalised service to reconcile information based on the user model. The user model can be determined in a number of ways, but in this instance it was determined by means of a questionnaire outlining possible interests in the domain model. These interests provided the basis of the user model. This section endeavours to find out if the users interests, as chosen by themselves, accurate reflected the news they received from the personalised news service.

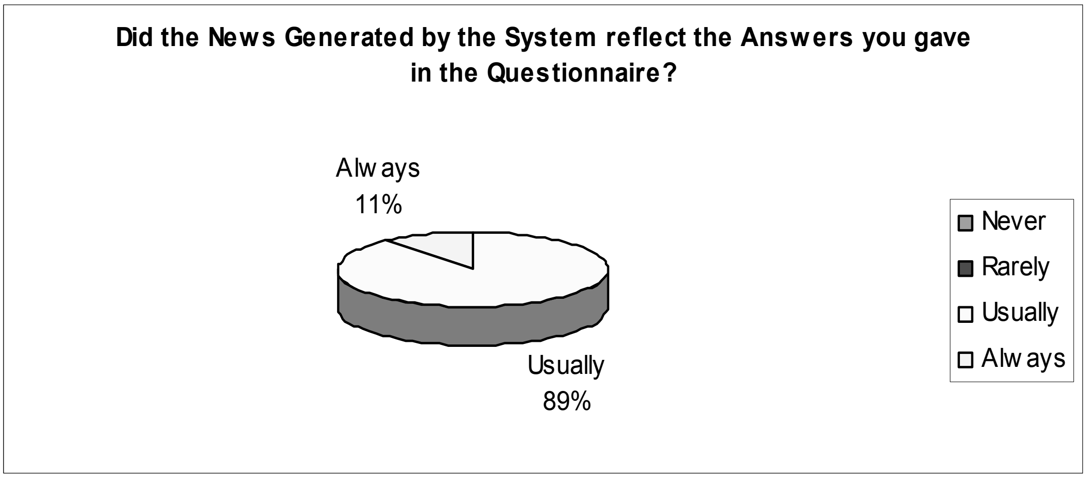


Figure 12 – Did the News generated reflect Questionnaire for Strict Ontology?

It can be seen from Figure 12, that in general, users felt the system accurately reflected the answers given in the first questionnaire to establish a users interests. The strict ontology formed the basis of this model.

It is not very surprising that the users felt the system was accurate, as the domain model had a good structure to it. In providing the structure for the domain, the users, whether they have a broad range of interests or a small number of interest, have news presented to them which is accurate according to the views presented by them in the questionnaire. The Strict Ontology, according to these results, provides a solid, accurate way of modelling a domain. In providing the structured, semantically rich domain ontology to represent the domain, each of the users were happy that their results reflected their answers given in the questionnaire.

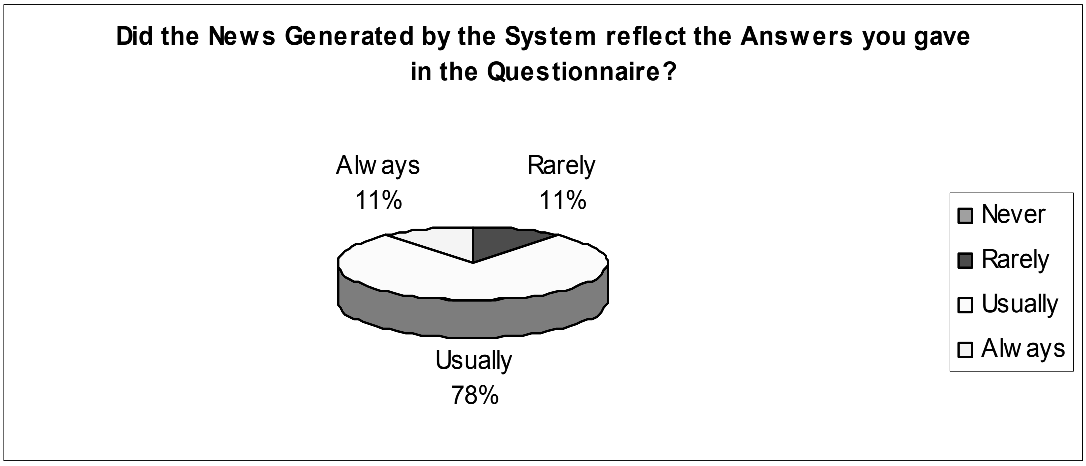


Figure 13 – Did the News generated reflect Questionnaire for Loose Ontology?

Again, in general, most users felt that the news generated and presented to them reflected the answers given to them in the questionnaire, even when the domain was modelled by the Loose ontology. Looking at the above pie chart in Figure 13, it is interesting to note how one user claimed that the service provided by the Loose ontology rarely reflected the answers given in the questionnaire. What is most interesting about this, is that this particular user had little interest in the sport, and had a very small user model, with very few interests. The Loose ontology would have provided the user with information which was of no interest to them. It is also interesting to note how the rest of the users felt that for the loose ontology that the system still accurately reflected the answers given in the questionnaire most of the time.

According to these results, the Loose ontology provides a solid method of modelling a domain, but it only really works when users have broad interests in the domain. The one user who wasn't happy with the reflection of their answers given in the questionnaire on their generated news did not have a very broad interest in the domain.

Concluding from Figure 12 and Figure 13, the Strict Ontology provides users with an accurate reflection of their interests. The Loose ontology provides an accurate reflection if the interest a user has in the domain is broad.

5.2.6 Users satisfaction with the Service

It is important that the user is satisfied with the service that is being provided. It is imperative that hits and correct rejections are kept to a maximum, whereas misses and false positives are kept to a minimum. A hit is where an item of interest to the user is provided to the user. A correct rejection is where an item which is of no interest to a user is not given to the user. Misses consist of items of interest for a user not being presented to a user, and false positives represent items which are presented to the user, but the user has no interest in the article. These four components must come under scrutiny when analysing a personalised news service.

5.2.6.1 User received news they found interesting

In a personalised news service, it is most important, that users receive the news they will find interesting. If they receive news that they find interesting then it is likely that they will be happy with the service being provided to them.

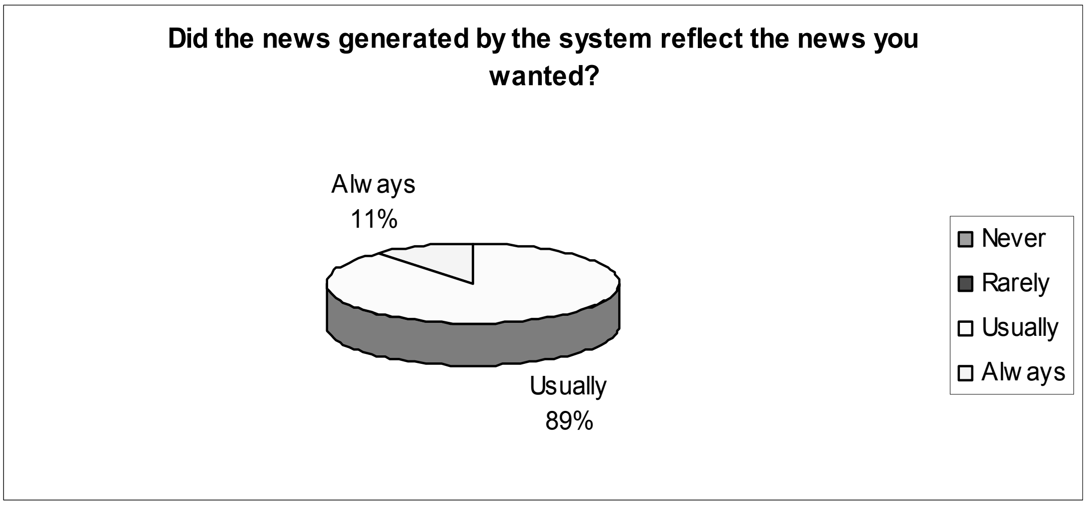


Figure 14 – Did the news generated reflect news you found interesting?

It is interesting to note that the pie chart above, Figure 14 reflects the answers given by users in relation to both ontologies. That is, the same question was asked of users with regard to both models, and the results came back exactly the same. There is no difference in the way any of the users answered, and it can be inferred that in generating news that the user wants, there is no significant difference between a strict and a loose ontology.

5.2.6.2 User received news they did not want

If a user receives news that they don’t want, then this is not much of a problem within the scope of this project. This is because with an RSS feed, the user does not have to look at every item provided, they only look at the ones they would like to read. Receiving news that is not wanted is still a problem however, and should be minimised as much as possible. It is important that a user is not sifting through news to find articles which are of interest.

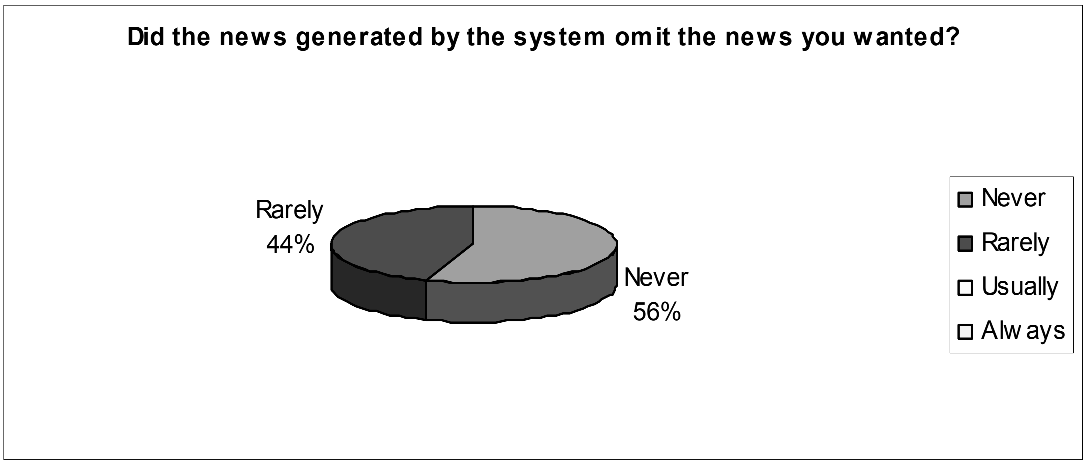


Figure 15 – Did the news generated omit news you wanted?

In much the same way as Figure 14, Figure 15 reflects the answers given by users for both the strict and loose ontologies. The system is generally accurate with the omitting news that it perceives is not of much interest to the user.

5.2.6.1 User did not receive news they wanted

In a commercial personalised news service this is something the user is not likely to see. It is unlikely that the user will be able to browse articles which were not presented to them. This can cause a problem if taking into consideration the advantage the user can have in changing the user model. If a user was to see the articles they weren’t presented with, then they could provide feedback based on providing feedback or information on which articles they would have liked to have seen.

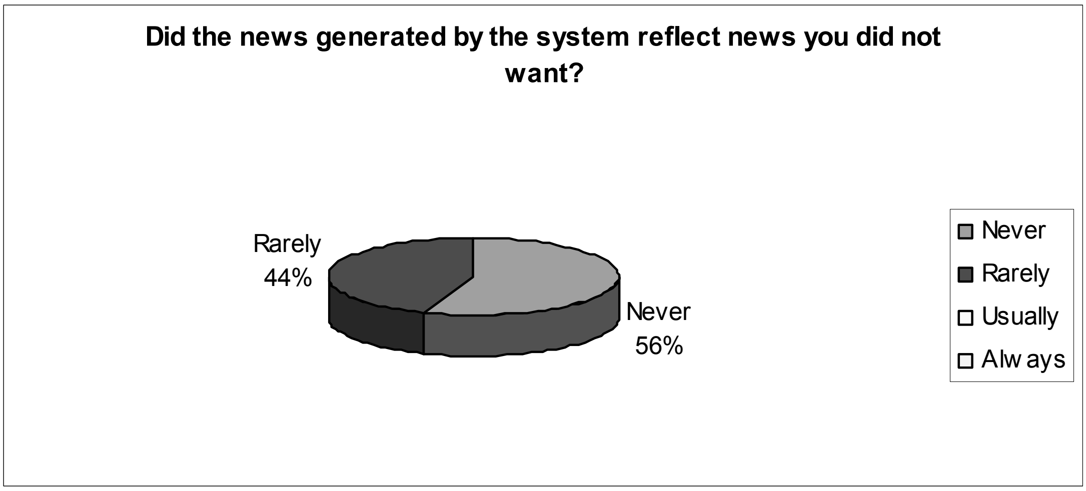


Figure 16 – Did the news generated reflect news you did not want (Strict)?

In Figure 16, it can be seen that in the case of the Strict ontology, news is not generally presented which isn't of interest to a user, and when it is, it is a rare occurrence.

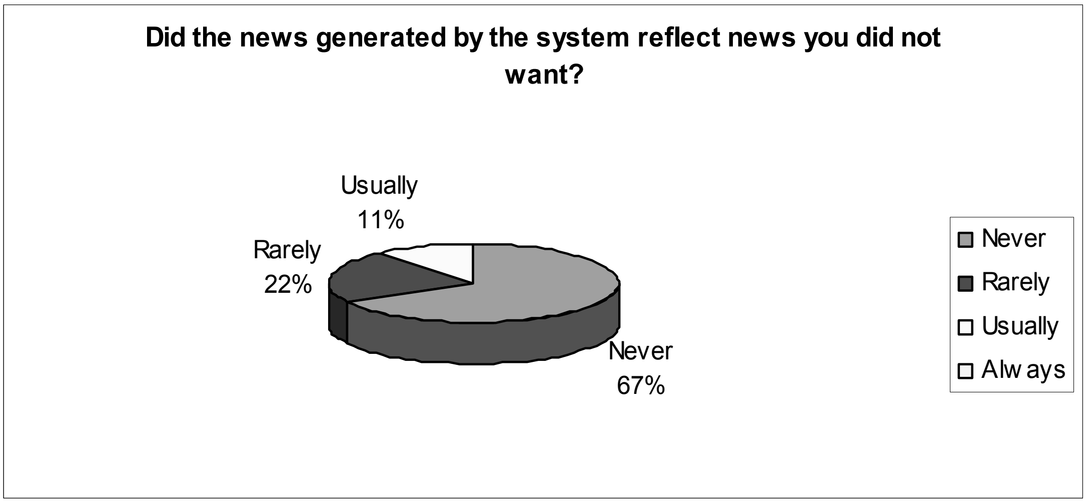


Figure 17 – Did the news generated reflect news you did not want (Loose)?

Figure 17 goes to show that in general, the system did not present items of little interest to a user. In this case, the Loose ontology was in use. There is however, one case where the system presented information to the user where it was of little interest. It should be mentioned, that this user is the same user who had very little interest in the domain, and who thought that the system rarely reflected answers they had given in the first Questionnaire.

5.2.7 Comparison of Results with Expected Results

The comparison of the results received with the results expected is also interesting. Given that the users knew little of what a high interest in an object correlated to, what the users were expecting may be different to what they received. The question asked of the users is trying to reflect the accuracy of the user modelling. If a user receives results which they expect to get, then the accuracy of the user model is pretty correct. If however, the user does not receive the results they were expecting then this can be very much to do with inaccurate user modelling.

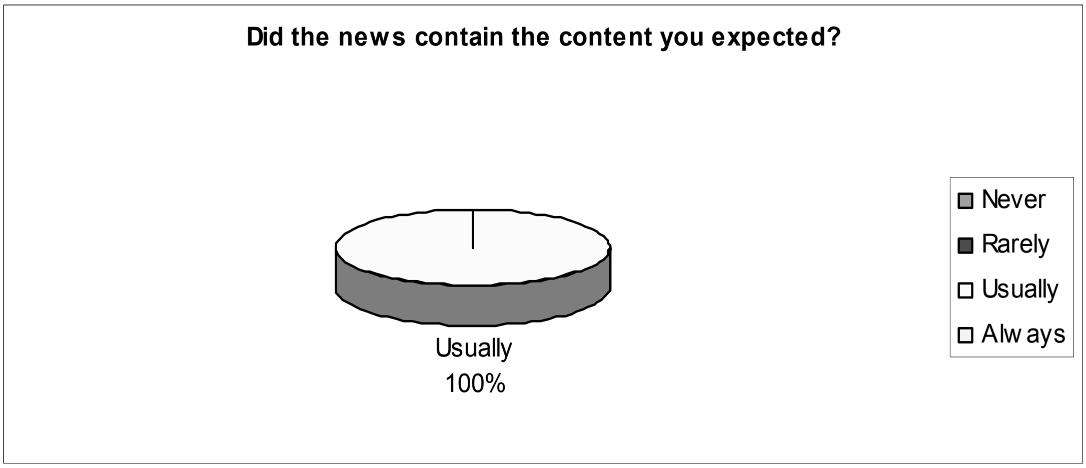


Figure 18 – Did the news generated contain the content you expected?

The users believed that for both models, the news usually contained the content they had expected it to contain. The one user who with a low interest in the domain did mention however that there was a little more than expected with the loose ontology, but in general, the news reflected by the system was the same as the news they were expecting.

5.2.8 Satisfaction with the User Model

The user model is the component of the system which represents the user’s interests in certain topics. It is vital that this is as accurate as possible, when representing a user’s interests. If the user model does not accurately model a user’s interests, then there should be a way of manipulating it so that the user model can change or adapt to the user’s interests.

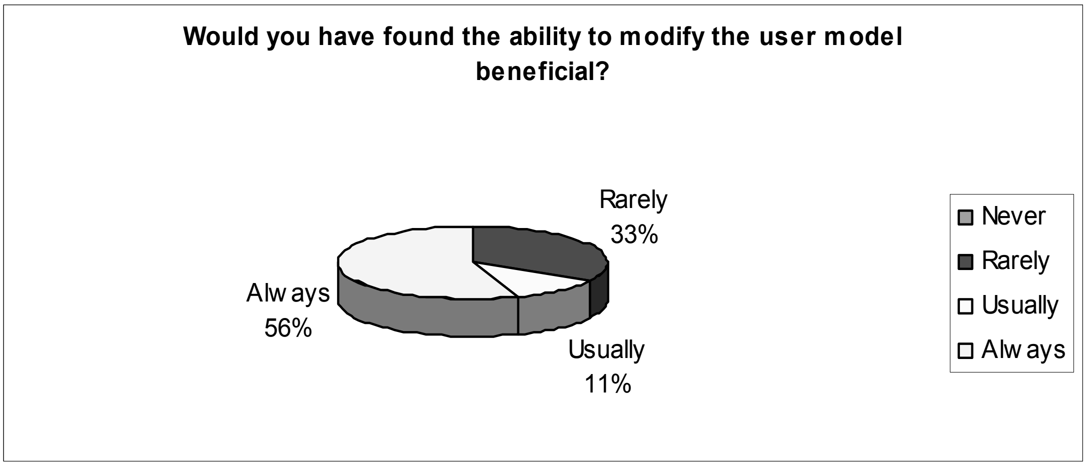


Figure 19 – Would you have liked to change your user model?

It is fascinating to see that although most users were happy with the content they received and their user model, they still would have liked to have the ability to change their user model. This is not entirely surprising, as most users will never usually be entirely happy with the first few runs of a system such as this, but with over half the users admitting they would have liked a lot more control over their user model, it seems that the system has to mature more before users are entirely happy with it.

5.2.9 Comparison of two Ontologies

This is the most interesting of the questions answered by the users. The structure of the two ontologies were the same, and the only difference between them was the semantic meaning being present in one but not in the other. It is interesting to note the perceived differences between the results provided by the two. It is also interesting to note the perceived differences between the two sets of results for different users, i.e. highly interested users who have an interest in most aspects of the domain, and other users who only have a few small interests in concepts in the domain.

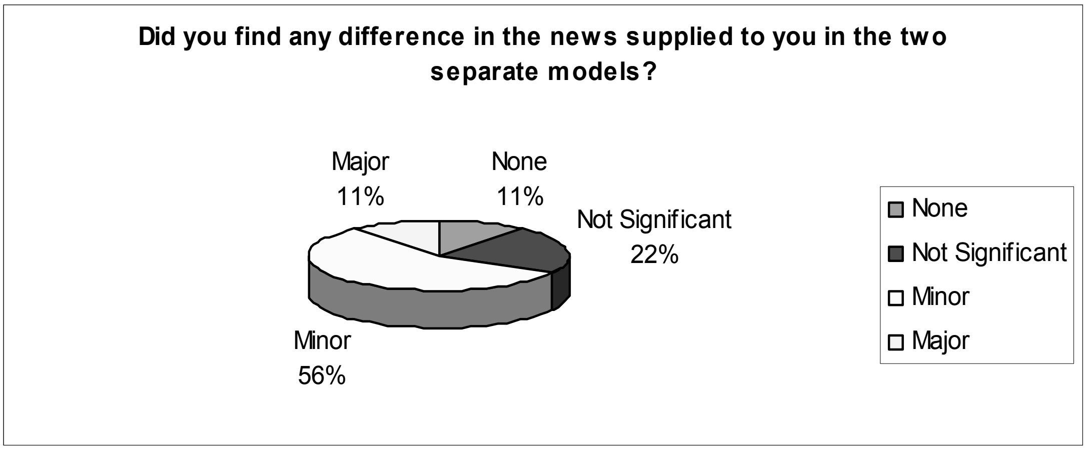


Figure 20 – Users perceived difference between the two ontologies

As can be seen from Figure 20, a majority of users perceived minor differences between the two ontologies. The user who perceived a major difference between the two was the user with the low interest in the domain. The other users, who had relatively broader interests in the domain, saw nothing more than minor differences between the two.

5.2.10 Additional Features which could be added

As with any other personalised news service, it is not fully complete, and it is interesting to note the requests for further features in the system. Unsurprisingly, there is the possibility for extending the personalised news service for different domains, and this is a feature which was requested. One user would have liked a rationale as to why news stories were chosen. If this was put into place then a user, who could control their user model, would be in a much better position to refine their user model. This is because they would see why news is being selected, based on their user model, and would refine their user model based on this information. Another request for the system was to have a more aggressive algorithm for choosing news in the loose ontology. The users did not know which ontology was the loose, and which was the strict, but one user would have liked more news on the loose ontology. This seems odd, as having received news from the strict ontology, and probably received more from the loose ontology, asking for even more news from the loose ontology but not the strict is a little odd. The personalisation applied to news, was not seen as much of a problem, but some users would have liked to have been able to change the level of personalisation of the news. This really calls for a user to be able to modify their user model and their interests.

5.2.11 Key Findings

There are a couple of key findings within the user evaluations of the system. Most significantly, the users who have a broad range of interests within the domain of interest, seem to see little difference between the two ontologies. This is because the wide range of interests translated into roughly the same type of interests for both ontologies. That is to say, with a wide range of interests, there was an interest in most if not all aspects of the domain. The one user who had little interest in the domain, and had a small but focused user model, perceived a bigger difference between the results provided by the two ontologies. This is because the focused set of interests the user had corresponded to different types of interests for both ontologies. For example, a medium interest in a driver would have a given a user a lot more information for the loose ontology than for the strict ontology. This is due to the fact that the loose ontology had little semantic meaning.

Given that the strict ontology provided all users with an accurate reflection of their interests, the notion could be put forward that strict ontologies should be used when modelling domains. The users with a low interest in the domain, would certainly benefit from it. The problem with using strict ontologies is that they take time to develop. They also must be developed by a domain expert, who is willing to put in the time and effort into modelling that domain.

By contrast, the loose ontology provided the broad interest users with an accurate reflection of their interests, but it did not provide an accurate reflection of the interests put forward by users who had only small and focused interests in the domain. The advantage of a loose ontology is that it does not require a domain expert to provide a model of the domain. A loose ontology could be developed by trawling relevant news sites and performing matching techniques to relate objects.

The decision when providing a personalised news service knowing this, is how to model the domain. If the user population is likely to have a large proportion of people who are interested in the domain, then a loose ontology would provide a painless method of model construction yet still provide the information the users are likely to want. However, the drawback is that if users are pretty interested in the domain, then they are likely to want specified information about the domain, and this may not be able to be provided by a loose domain model.

The Strict ontology model provides a happy medium in that it provides by broad interest and low interest users with an accurate reflection of their user model. With a strict domain model, there is also the likelihood that specific information will be provided with better regularity than a loose domain model. The only problem lies with the availability of a domain expert to produce a model of the domain.

It is also interesting to see that within the system, a simple user model reflected accurately the interests of the user. While the simple user model was employed, most of the users were reasonably happy with the content provided to them. This is not to say that the users were entirely happy with their user model, as all expressed some interest in being able to modify and refine the way their interests were modelled. This goes to further show that simple user models can work well in adaptive systems.

5.3 Service Oriented Architecture

The service oriented architecture was chosen for the implementation of the personalised news service. The architecture comprised of three main components, the personalised news service, the ontology reasoning web service, and the news feed capture and personalisation service. The reason for choosing this type of architecture was to try and make sure that the components were as independent and reusable as possible.

5.4.1 Personalised News Service

The Personalised News Service comprises of an independent reusable system which can provide personalised news for users of the system. The narrative model, which contains the adaptivity, is independent of the content it provides. It simply works off parameters provided to it, including the models such as the conceptual and user models. The dynamic query built by the narrative is dependent on the domain over which it reasons, but the query and domain model can be provided as parameters, thus keeping the Personalised News service as independent as possible. The narrative is also independent of the news feed service, and can provide the news feed service with the parameters needed for it to do its job.

The system is extensible, as it is possible to add Web Services to the system which can performs tasks the narrative model would find difficult to do. Web services provide a structured method of manipulating data in a way that the Narrative Model cannot do.

5.4.2 Ontology Reasoning Web Service

The ontology reasoning web service is a web service which not so much wholly independent, but not incredibly dependant on the domain. The ontology reasoning web service knows where the domain model is located, and is also based on whether the query is for the strict or the loose ontology. The service could be made wholly independent by passing the location of the domain model as a parameter, along with the query. The web services are stateless and allow a structured methodology for procuring information by means not available in the adaptive system

The Ontology Reasoning Web Service is reusable within the context of the Personalised News System. The service can provide reasoning over a domain, once the service is passed a query capable of querying the domain, and the location of the domain model. The service is also extensible, and can for instance, provide more incisive querying over a domain model if needs be.

5.4.3 News Feed Capture and Personalisation Web Service

The news feed capture and personalisation web service is a highly reusable and completely independent component. The service is independent of the news service (i.e. RSS Feed) it connects to, and is independent of the list of items which a user might find interesting within those articles. The service would apply equally well from a different perspective regardless of the domain, and the specificity of the domain.

The News feed capture and personalisation service can be reused by any system looking to incorporate a live RSS feed into its usage. The service requires two parameters, the RSS feed it needs to connect to, and the list of terms of interest, thus rendering it completely independent and a highly reusable component. It is also extensible and can provide more than manipulation of an RSS feed. This ability would however have to be plugged into the web service, as it is not currently capable of doing anything more than manipulating a specified RSS feed.

6 Conclusion

6.1 Introduction

This thesis has presented a service oriented approach to ontology driven personalised news services. This approach provides an extensible system, with reusable and independent components, that provides users with personalised news based on information provided by those users with respect to their interests within a particular domain.

This chapter discusses the objectives of the thesis and how these objectives were achieved. It contains a section on the contributions to the state of the art in the area of adaptive hypermedia, and another final section on future work which could be carried out in the area.

6.2 Objectives and Achievements

The research question posed by this thesis was whether there was a tangible difference between the results provided by an adaptive system when provided with two semantically different ontologies. The primary objective of the thesis was to analyse and evaluate this tangible difference, and provide a conclusion on which type of ontology would be better suited to providing information that a user is satisfied with.

The Personalised News Service delivered a satisfactory news service to the users of the system. In the majority of cases, users received the news they found interesting, which is the most important part of any news service. The users were happy that the user models provided an accurate reflection of their interests, and the users generally got the news they were expecting to receive.

The adaptive logic, or narrative model, is completely separate from the news content, which provides an independent method for producing personalised news feeds. Because the content is not tied down to the narrative model, the service can provide news feeds for any domain, as long as there is a model describing that domain. The model, is also independent, and does not necessarily have to be an ontology, but may be constructed differently. The goal of having reusable components and models is implemented in the construction of separate models, which perform a single task, and combine to produce a personalised news service.

The service, does not require, a priori mining of information from the news content. The service requires a domain expert to provide the information to model the domain. The mining of content provides a taxonomy of objects and relationships, but provides no structure or semantics for the domain. In providing a structured view of the domain, there is no need for mining, or trawling of hyperlinked resources to establish a domain model.

The structured view is provided by a domain ontology, providing semantic meaning to concepts within the domain. The ontologies produced, provide a structured view of the domain being represented. In using the ontologies, the system has gained a methodology for describing and querying a domain, in a robust and expressive fashion.

The system provides an extensible methodology, for allowing more adaptive axes to be adapted toward. Adaptive axes such as the user's temporal constraints, contextual axes such as network bandwidth and screen size can all have an effect on how information is provided to a user, and how much information is provided to a user. This has been accounted for in the design by employing many independent and reusable components, rather than having interdependent components which do not easily facilitate the addition of new constraints to the problem domain.

The system provides reusable, independent components, such as the two Web Services in use, the Ontology Reasoner, and the News Feed Capture and Personalisation Web Services. The two components, are independent and are not tied to any of the models which provide adaptation, such as the user model, and the content model. Other reusable and independent components include the user model and the content model. In incorporating a service oriented design, the system lends itself well to additions of new services to the solution domain.

This thesis proposed an approach to personalizing news feeds towards a user's interests. The key points in assessing the success of this work were the trial and evaluation and the innovative technical approach. Through the Trial and Evaluation it was shown that the system produced satisfactory results for the majority of users. This finding underpins the validity of the technical approach and, therefore, gave a baseline from which the two ontology mapping approaches could be assessed. The Personalised News Service utilises generic adaptation mechanisms and a suite of distributed services to achieve the effective personalisation of news based on both strict and loose ontologies. The high levels of user satisfaction indicate that this approach has merit.

6.3 Contribution to State of the Art

Ontologies have been shown in this thesis to provide a structured and semantically rich view of a domain. The tangible differences between the two ontologies modelled, are only really noticed by the users of the system. This thesis has contributed an analysis of the two types of ontology and a reasoning behind using these ontologies when providing a personalised news service.

In the context of users who have a broad interest in the domain, a case must be put forward of the need for a personalised news service. If a user has a broad interest in a domain, then it is likely that they will find most, if not all, articles presented to them interesting. This can prove that there may not be a need for a Personalised News Service for people of broad interests. For users, who have small focused interests, the need for a Personalised News Service is greater. This is because there is a likelihood that there are articles of interest which a focused interest user will not find interesting. This proves that there may be a need for a Personalised News Service for the focused interest type of user.

For a user with a broad interest in a domain, it is likely that a loose ontology would suffice in providing news for that user. The loose ontology will negate the need for a domain expert, and can provide the broad interest users with a reasonably accurate service. For the focused interest users, a strict ontology would be a more accurate domain model for personalising news. This is because with the semantic meaning provided, it would be better able to infer the information a user would like to see presented to them.

The thesis has provided a service oriented solution to the problem of creating a personalised news service, and has provided reusable adaptable components for future work within the area. The three components which provide the Personalised News Service are independent components, allowing the service to be extensible and scalable.

The thesis has also contributed to a specific problem within the domain of ontology querying. RDQL is a query language which is difficult to implement, and work into practice. The narrative model, which was the driving force behind the adaptivity, was required to build a dynamic RDQL query. This is because the narrative was adapting to information provided by the user model and the level of interest model. This dynamic query was built for both the strict ontology, and the loose ontology, and the thesis provided a methodology for producing dynamic RDQL queries, which could then be used to query the user model.

6.4 Future Work

6.4.1 Different Sporting Domains

It would be interesting to see the results provided by ontologies which model different sporting domains. The domain chosen was one which is small in nature, compared to other sports such as football, which would require a colossal task to model. With so many teams, players and managers, and the associated political figures such as chief executives and agents in the news, football would certainly be an interesting domain in which to test the difference between a strict and loose ontology.

6.4.2 More Diverse Domains

Looking at more diverse domains is another area of work which could be looked at in the future. Areas such as sport can be relatively easy to model, whereas areas such as politics and entertainment tend to be less easy to model. Determining the difference between a strict ontologies view and loose ontologies view of politics would be an interesting experiment, especially given the diversity of political parties and political views within different countries and communities.

6.4.3 Ontologies

Future work in the area of ontologies would include trying to find a mid point between a strict and a loose ontology for the domains being modelled which suits users. This may not be very feasible, as the primary way to fine tune how a personalised service provides information is to refine the user model rather than changing any other part of the system.

6.4.4 Adaptation Mechanisms

There are other Adaptation mechanisms which must be taken into account when looking at future work in the area. Mechanisms such as temporal or location based adaptation could be looked at. Temporal adaptation would be useful for users who wish to have the most up to date view of the news which they are receiving. Older news may not be of much use to them, and only recent news may be of interest to them. In cases such as this, a temporal axis should be put in place. Also, a location based axis could be put in place. A user's location should have a minimal effect on the news a user is able to receive. In cases such as this, it is important to be able to adapt to a location based contextual axis.

6.4.5 User Model Control

Looking back at the trial and evaluation chapter, it can be seen in section 5.2.8 that the users would have liked more control over the user model. The user models were hand built from the information provided in the interests questionnaire, and could only be adapted through access of the files. One future area of work would be to look at the possibility of creating an adaptive user model. A user model which would be beneficial would be a model the user could control, and refine, so they could receive the news which best fits their interests.

Bibliography

[1] Extensible Markup Language (XML) 1.0 (Third Edition),
http://www.w3.org/TR/2004/REC-xml-20040204/

[2] What is RSS Feedreader? | Feedreader - Totally Free RSS / ATOM Newsreader / Aggregator
http://www.feedreader.com/

[3] Java Technology,
http://java.sun.com/

[4] Paul Cotter, Barry Smyth, *PTV: Intelligent Personalized TV Guides*, in Proceedings of the 17th National Conference on Artificial Intelligence, AAAI 2000, Austin, Texas, 2000, pp 957-964

[5] Joseph Konstan, Bradley Miller, David Maltz, Jonathan Herlocker, Lee Gordon, John Riedl, *Applying Collaborative Filtering to UseNet News*, in Communications of the ACM March 1997/Vol. 40, No. 3, pp 77-87

[6] Manolis Vozalis, Konstantinos Margaritis, *Unison-CF: A Multiple-Component, Adaptive Collaborative Filtering System*, in Adaptive Hypermedia 2004, LNCS 3137, pp 255-264

[7] Aditya Sunderam, *Automated Personalization of Internet News*, in Adaptive Hypermedia 2002, LNCS 2347, pp 348-357

[8] Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more,
http://www.amazon.com

[9] Patrick Baudisch, Lars Brueckner, *TV Scout: Lowering the Entry Barrier to Personalized TV Program Recommendation*, in Adaptive Hypermedia 2002, LNCS 2347, pp 58-68

[10] Tim Berners-Lee, James Hendler, Ora Lassila, *The Semantic Web*, in Scientific American, May 17, 2001

[11] Jacco van Ossenbruggen, Lynda Hardman, Lloyd Rutledge, *Hypermedia and the Semantic Web: A Research Agenda*, in Journal of Digital Information, Volume 3 Issue 1, May 17, 2002

[12] J Arjona, R Corchuelo, A Ruiz, M Toro, *Automatic Extraction of Semantically-Meaningful Information from the Web*, in Adaptive Hypermedia 2002, LNCS 2347, pp 24-35

[13] Trent Apted, Judy Kay, Andrew Lum, *Supporting Metadata Creation with an Ontology Built from an Extensible Dictionary*, in Adaptive Hypermedia 2004, LNCS 3137, pp 4-13

[14] Yannis Kalfoglou, John Domingue, Enrico Motta, Maria Vargas-Vera, Simon Buckingham Shum, *myPlanet: An Ontology-Driven Web-Based personalised news service*, in Proceedings IJCAI 2001 Workshop on Ontologies and Information Sharing, Seattle USA, pp 140-148

[15] AnHai Doan, Jayant Madhavan, Pedro Domingos, Alon Halevy, *Learning to Map Between Ontologies on the Semantic Web*, in WWW2002, May 7-11, 2002, Honolulu, Hawaii, USA, pp 662-673

[16] Marcus Specht, Milos Cravcik, Leonid Pesin, Roland Klemke, *Authoring Adaptive Educational Hypermedia in WINDS*, in Proceedings of ABIS2001, Dortmund, Germany, October, 2001, pp 1-8

[17] Gerhard Weber, Hans-Christian Kuhl, Stephan Weibelzahl, *Developing Adaptive Internet Based Courses with the Authoring System NetCoach*, in Adaptive Hypermedia 2001, LNCS 2266, pp 226-238

[18] Peter Brusilovsky, *Adaptive Hypermedia*, in User Modeling and User-Adapted Interaction 11: pp 87-110, 2001

[19] Peter Brusilovsky, *Adaptive Navigation Support: From Adaptive Hypermedia to the Adaptive Web and Beyond*, in PsychNology Journal, 2004 Volume 2, Number 1, pp 7-23

[20] Paul De Bra, Ad Aerts, Bart Berden, Barend de Lange, Brendan Rousseau, Tomi Santic, David Smits, Natalia Stash, *AHA! The Adaptive Hypermedia Architecture*, in Proceedings of the ACM Hypertext Conference, Nottingham, UK, pp 81-84

[21] Alexandra Christea, Lora Aroyo, *Adaptive Authoring of Adaptive Educational Hypermedia*, in Adaptive Hypermedia 2002, LNCS 2347, pp 122-132

[22] Licia Calvi, Alexandra Christea, *Towards Generic Adaptive Systems: Analysis of a Case Study*, in Adaptive Hypermedia 2002, LNCS 2347, pp 79-89

[23] Owen Conlan, Vincent Wade, *Evaluation of APeLS – An Adaptive eLearning Service based on the Multi-model, Metadata-driven Approach*, in Adaptive Hypermedia 2004, LNCS 3137, pp 291-295

[24] Owen Conlan, Vincent Wade, Catherine Bruen, Mark Gargan, *Multi-Model, Metadata Driven Approach to Adaptive Hypermedia Services for Personalized eLearning*, in Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 100-111

[25] RDF Vocabulary Description Language 1.0: RDF Schema,
http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[26] OWL Web Ontology Language Reference,
http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[27] Jatha - LISP Library in Java,
http://jatha.sourceforge.net

[28] LispWorks,
http://www.lispworks.com/products/lisp-overview.html

[29] Algernon-J Rule-Based Programming,
http://algernon-j.sourceforge.net/

[30] Bernard Merialdo, Kyung Tak Lee, Dario Luparello, Jeremie Roudaire, *Automatic Construction of Personalised TV News Programs*, in ACM Multimedia '99, October 99, Orlando, Florida, USA, pp 323-331

[31] Sami Jokela, Marko Turnpeinen, Teppo Kurki, Eerika Savia, Reijo Sulonen, *The Role of Structured Content in a Personalised News Service*, in Proceedings of the 34th Hawaii International Conference on System Sciences 2001, pp 1-10

[32] Daniel Billsus, Michael Pazzani, *User Modelling for Adaptive News Access*, in User Modeling and User Adapted Interaction 10: pp 147-180, 2000

[33] Liliana Ardissono, Luca Console, Iliara Torre, *An Adaptive System for the Personalised Access to News*, in AI Communications 14: pp 129-147, 2001

[34] Ah-Hwee Tan, Christine Teo, *Learning User Profiles for Personalised Information Dissemination*, in Proceedings of International Joint Conference on Neural Network 1998, pp 183-188

[35] J Boticario, E Gaudioso, F Hernandez, O Santos, A Rodriguez, C Barrera, *Current problems in eLearning and the aDeNu Approach*, in European Association of Distance Teaching Universities Annual Conference, Bologna, 2003

[36] Marcus Specht, Milos Kravcik, Roland Klemke, Leonid Pesin, Rüdiger Hüttenhain, *Personalized eLearning and eCoaching in WINDS*, in Proceedings of Workshop on Integrating Technical and Training Documentation, ITS 2002, San Sebastian, Spain

[37] Peter Brusilovsky, Sergey Sosnovsky, Michael Yudelson, *Adaptive Hypermedia Services for eLearning*, in Proceedings of Workshop on Applying Adaptive Hypermedia Techniques to Service Oriented Environments at the Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2004), Eindhoven, The Netherlands

[38] Peter Brusilovsky, Elmar Schwarz, Gerhard Weber, *ELM-ART: An Intelligent Tutoring System on World Wide Web*, in Third International Conference on Intelligent Tutoring Systems, ITS-96, Montreal, 1996

[39] The Protégé Ontology Editor and Knowledge Acquisition System,
http://protege.stanford.edu

[40] Jena Semantic Web Framework,
http://jena.sourceforge.net/

[41] Ian O'Keeffe, Owen Conlan, Declan Dagger, Vincent Wade, *Dynamically Adapting Multimedia Content using Strategically-driven Generic Adaptive Services*, in ACM Multimedia 2005, Singapore, November (Submitted)

[42] Declan Dagger, Vincent P. Wade, Owen Conlan, *Developing Adaptive Pedagogy with the Adaptive Course Construction Toolkit (ACCT)*, in the workshop proceedings of the AH2004 conference (online), Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2004) Proceedings, Eindhoven, The Netherlands (2004)

[43] itv.com/f1 News and Features,
http://www.itv-f1.com/rss/latest/

[44] itv.com/f1
http://www.itv-f1.com/

[45] DAML+OIL (March 2001) Reference Description
http://www.w3.org/TR/daml+oil-reference

Appendices

Appendix I – JSP Pages

```
<%@ page import="ie.tcd.cs.kdeg.ae.AdaptiveEngine" %>

<%

  AdaptiveEngine myAE;
  myAE = new AdaptiveEngine (false);
  myAE.createNewEngine("myae");


myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/n
arrative", "narrativeCollection");

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/c
ourses", "coursesCollection");

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/l
earners", "learnerCollection");

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/l
os", "learningObjectCollection");

  //These are for the User Models and the Concept Models

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/m
yUsers", "myUserCollection");

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/m
yConcepts", "myConceptCollection");

  String username = request.getParameter( "user" );

  myAE.loadModelIntoManager("myConceptCollection", "myae", "CombinationConceptModel",
"completeConceptModel");
  myAE.loadModelIntoManager("narrativeCollection", "myae", "ModelOne", "narrative");
  //userModel (Final Variable) name is the name defined in the narrative
  //User2 is the name of the files who user model we are extracting info from
  myAE.loadModelIntoManager("myUserCollection", "myae", username, "userModel");


  myAE.addModelToEngine("myae", "narrative");
  myAE.addModelToEngine("myae", "userModel");
  myAE.addModelToEngine("myae", "completeConceptModel");

  myAE.setNarrative("myae", "narrative");

  myAE.setPersistenceState("myae", true);

  myAE.startEngine("myae");

  //myAE.createNewModel("myae", "nModel");

  String text = "";
  //getModelContent, getNodeText give null
  //getPrettyModelContent sometimes returns null, other times returns an error
  text = text + myAE.getPrettyModelContent("myae", "rss");

  text = text.replaceAll("&gt;", ">");
  text = text.replaceAll("&lt;", "<");
%>

               <%= text %>

<%
  myAE.exit();
%>
<%@ page import="ie.tcd.cs.kdeg.ae.AdaptiveEngine" %>

<%

  AdaptiveEngine myAE;
  myAE = new AdaptiveEngine (false);
  myAE.createNewEngine("myae");


myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/n
arrative", "narrativeCollection");

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/c
ourses", "coursesCollection");

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/l
earners", "learnerCollection");

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/l
os", "learningObjectCollection");

  //These are for the User Models and the Concept Models

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/m
yUsers", "myUserCollection");

myAE.createRepositoryConnection("xmldb:exist://localhost:8080/exist/xmlrpc/db/ae/sql/m
yConcepts", "myConceptCollection");

  String username = request.getParameter( "user" );

  myAE.loadModelIntoManager("myConceptCollection", "myae", "CombinationConceptModel",
"completeConceptModel");
  myAE.loadModelIntoManager("narrativeCollection", "myae", "ModelTwo", "narrative");
  //userModel (Final Variable) name is the name defined in the narrative
  //User2 is the name of the files who user model we are extracting info from
  myAE.loadModelIntoManager("myUserCollection", "myae", username, "userModel");



  myAE.addModelToEngine("myae", "narrative");
  myAE.addModelToEngine("myae", "userModel");
  myAE.addModelToEngine("myae", "completeConceptModel");

  myAE.setNarrative("myae", "narrative");

  myAE.setPersistenceState("myae", true);

  myAE.startEngine("myae");

  //myAE.createNewModel("myae", "nModel");

  String text = "";
  //getModelContent, getNodeText give null
  //getPrettyModelContent sometimes returns null, other times returns an error
  text = text + myAE.getPrettyModelContent("myae", "rss");

  text = text.replaceAll("&gt;", ">");
  text = text.replaceAll("&lt;", "<");
%>

               <%= text %>

<%
  myAE.exit();
%>
```

Appendix II – Narrative Models

```
<?xml version="1.0" encoding="UTF-8"?>
<narrative>
    <id>ModelOne</id>
    <type>JESS</type>
    <code>;; Function which takes a single Multifield value and turns it into a string
(deffunction get-multifield-value (?field)
    (bind ?varBind (implode$ ?field))
    (return (sub-string 2 (- (str-length ?varBind) 1) ?varBind))
)

(deffunction output-query2 (?uModel ?cModel)

        (bind ?nameInterestVarInMethod
        (xpath-query-model ?uModel "/user/interest/name_of_interest")
        )

        ;;This will iterate through the Name_Of_Interest Stuff, and build some dynamic
XPath
        ;;Queries based on that
        (foreach ?bindingVariable ?nameInterestVarInMethod
                ;;This below line has a direct correlation with the (cd modelname)
thing
                ;;at the end of the for loop
                ;;(update-model "rss" "myQuery")

                ;;(update-model "rss" "interestName" ?bindingVariable)

         (bind ?firstXPathQuery (str-cat "//interest[name_of_interest='"
?bindingVariable "']/depth_of_interest"))
                (bind ?secondXPathQuery (str-cat "//interest[name_of_interest='"
?bindingVariable "']/concept_of_interest"))

                (bind ?depthInterest (xpath-query-model ?uModel ?firstXPathQuery))
                (bind ?conceptInterest (xpath-query-model ?uModel ?secondXPathQuery))

                (bind ?depthInterest
                (get-multifield-value ?depthInterest)
                )

                (bind ?conceptInterest
                (get-multifield-value ?conceptInterest)
                )

                ;;(update-model "rss" "interestConcept" ?conceptInterest)
                ;;(update-model "rss" "interestDepth" ?depthInterest)

                (bind ?thirdXPathQuery (str-cat "//interest[concept='" ?conceptInterest
"']/interest_type[type='" ?depthInterest "']/relationship"))

                (bind ?relationshipsOfInterest (xpath-query-model ?cModel
?thirdXPathQuery))

                ;;This is where I will build the query dynamically
                (bind ?strictQuery (str-cat "SELECT ?x WHERE (?x ?predicate "
?bindingVariable ")"))
                ;;;;(bind ?looseQuery (str-cat "SELECT ?x WHERE (?x ?predicate "
?bindingVariable ")"))

                (if (eq (str-compare ?depthInterest "low") 0)
                then
                        (bind ?strictQuery (str-cat ?strictQuery " (?x rdfs:label
?ob1Label)"))
                        ;;;;(bind ?looseQuery (str-cat ?looseQuery " (?x rdfs:label
?ob1Label)"))
                else
                        (bind ?firstOb 1)

                        (foreach ?relationship ?relationshipsOfInterest
                        ;;(update-model "rss" "relationship" ?relationship)
                        (bind ?strictQuery (str-cat ?strictQuery " (?x so:"
?relationship " ?ob" ?firstOb ")"))
                        (bind ?firstOb (+ ?firstOb 1))
                        )

                        (bind ?secondOb 1)

                        (foreach ?relationship ?relationshipsOfInterest
                        (bind ?strictQuery (str-cat ?strictQuery " (?ob" ?secondOb "
rdfs:label ?ob" ?secondOb "Label)"))
                        (bind ?secondOb (+ ?secondOb 1))
                        )

                        ;;;;(if (eq (str-compare ?depthInterest "medium") 0)
                        ;;;;then
                                ;;;;(bind ?looseQuery (str-cat ?looseQuery " (?x
lo:isRelatedTo ?ob) (?ob rdfs:label ?obLabel)"))
                        ;;;;)
                        ;;;;(if (eq (str-compare ?depthInterest "high") 0)
                        ;;;;then
                                ;;;;(bind ?looseQuery (str-cat ?looseQuery " (?x
lo:isRelatedTo ?ob1) (?ob1 lo:isRelatedTo ?ob2) (?ob1 lo:isRelatedTo ?ob1Label) (?ob2
rdfs:label ?ob2Label)"))
                        ;;;;)
                )

                (bind ?strictQuery (str-cat ?strictQuery " USING so FOR
&lt;http://www.owl-ontologies.com/unnamed.owl#&gt;"))
                ;;;;(bind ?looseQuery (str-cat ?looseQuery " USING lo FOR
&lt;http://www.owl-ontologies.com/unnamed.owl#&gt;"))

                ;;(update-model "rss" "strictQueryUnfolding" ?strictQuery)

                ;;;;(update-model "rss" "looseQueryUnfolding" ?looseQuery)

                (bind ?strictResults
                (call-web-service  "OntologyReasonerWebService.jws"
"http://sweep.cs.tcd.ie:8080/ShanesReasoner/" "returnTerms" "strict" ?strictQuery)
                )

                (bind ?overallStrictResults (str-cat ?overallStrictResults
?strictResults " "))

                ;;(update-model "rss" "webserviceresult" ?strictResults)

                ;;;;(bind ?looseResults
                ;;;;(call-web-service  "OntologyReasonerWebService.jws"
"http://sweep.cs.tcd.ie:8080/ShanesReasoner/" "returnTerms" "loose" ?looseQuery)
                ;;;;)

                ;;;;;(bind ?overallLooseResults (str-cat ?overallLooseResults
?looseResults " "))

                ;;;;(update-model "rss" "webserviceresult" ?looseResults)

                ;;(cd "rss" "..")
        )


)

;;---------------------------------------------------------------------------------

(create-model "rss")

;;---------------------------------------------------------------------------------

(bind ?overallStrictResults " ")

;;;;(bind ?overallLooseResults " ")



(output-query2 "userModel" "completeConceptModel")

(bind ?myStrictResults
(call-web-service "ReturnPersonalisedFeedWebService.jws"
"http://sweep.cs.tcd.ie:8080/personalisedNewsService/" "personaliseFeed"
"http://www.itv-f1.com/rss/latest/" ?overallStrictResults)
)
```

```
(update-model "rss" "channel" ?myStrictResults)


(store-model "myUserCollection" "rss" "myPNSModelOneResult")</code>
</narrative>





<?xml version="1.0" encoding="UTF-8"?>
<narrative>
    <id>ModelTwo</id>
    <type>JESS</type>
    <code>;; Function which takes a single Multifield value and turns it into a string
(deffunction get-multifield-value (?field)
    (bind ?varBind (implode$ ?field))
    (return (sub-string 2 (- (str-length ?varBind) 1) ?varBind))
)

(deffunction output-query2 (?uModel ?cModel)

        (bind ?nameInterestVarInMethod
        (xpath-query-model ?uModel "/user/interest/name_of_interest")
        )

        ;;This will iterate through the Name_Of_Interest Stuff, and build some dynamic
XPath
        ;;Queries based on that
        (foreach ?bindingVariable ?nameInterestVarInMethod
               ;;This below line has a direct correlation with the (cd modelname)
thing
               ;;at the end of the for loop
               ;;(update-model "rss" "myQuery")

               ;;(update-model "rss" "interestName" ?bindingVariable)

         (bind ?firstXPathQuery (str-cat "//interest[name_of_interest='"
?bindingVariable "']/depth_of_interest"))
               (bind ?secondXPathQuery (str-cat "//interest[name_of_interest='"
?bindingVariable "']/concept_of_interest"))

               (bind ?depthInterest (xpath-query-model ?uModel ?firstXPathQuery))
               (bind ?conceptInterest (xpath-query-model ?uModel ?secondXPathQuery))

               (bind ?depthInterest
               (get-multifield-value ?depthInterest)
               )

               (bind ?conceptInterest
               (get-multifield-value ?conceptInterest)
               )

               ;;(update-model "rss" "interestConcept" ?conceptInterest)
               ;;(update-model "rss" "interestDepth" ?depthInterest)

               (bind ?thirdXPathQuery (str-cat "//interest[concept='" ?conceptInterest
"']/interest_type[type='" ?depthInterest "']/relationship"))

               (bind ?relationshipsOfInterest (xpath-query-model ?cModel
?thirdXPathQuery))

               ;;This is where I will build the query dynamically
               ;;;;(bind ?strictQuery (str-cat "SELECT ?x WHERE (?x ?predicate "
?bindingVariable ")"))
               (bind ?looseQuery (str-cat "SELECT ?x WHERE (?x ?predicate "
?bindingVariable ")"))

               (if (eq (str-compare ?depthInterest "low") 0)
               then
                       ;;;;(bind ?strictQuery (str-cat ?strictQuery " (?x rdfs:label
?ob1Label)"))
                       (bind ?looseQuery (str-cat ?looseQuery " (?x rdfs:label
?ob1Label)"))
               else
```

```
                        ;;;;(bind ?firstOb 1)

                        ;;;;(foreach ?relationship ?relationshipsOfInterest
                        ;;;;;;(update-model "rss" "relationship" ?relationship)
                        ;;;;;;(bind ?strictQuery (str-cat ?strictQuery " (?x so:"
?relationship " ?ob" ?firstOb ")"))
                        ;;;;(bind ?firstOb (+ ?firstOb 1))
                        ;;;;)

                        ;;;;(bind ?secondOb 1)

                        ;;;;(foreach ?relationship ?relationshipsOfInterest
                        ;;;;(bind ?strictQuery (str-cat ?strictQuery " (?ob" ?secondOb "
rdfs:label ?ob" ?secondOb "Label)"))
                        ;;;;(bind ?secondOb (+ ?secondOb 1))
                        ;;;;)

;;                      (if (eq (str-compare ?depthInterest "medium") 0)
;;                      then
                                (bind ?looseQuery (str-cat ?looseQuery " (?x
lo:isRelatedTo ?ob) (?ob rdfs:label ?obLabel)"))
;;                      )
;;                      (if (eq (str-compare ?depthInterest "high") 0)
;;                      then
;;                              (bind ?looseQuery (str-cat ?looseQuery " (?x
lo:isRelatedTo ?ob1) (?ob1 lo:isRelatedTo ?ob2) (?ob1 rdfs:label ?ob1Label) (?ob2
rdfs:label ?ob2Label)"))
;;                      )
                )

                ;;;;(bind ?strictQuery (str-cat ?strictQuery " USING so FOR
&lt;http://www.owl-ontologies.com/unnamed.owl#&gt;"))
                (bind ?looseQuery (str-cat ?looseQuery " USING lo FOR
&lt;http://www.owl-ontologies.com/unnamed.owl#&gt;"))

                ;;;;(update-model "rss" "strictQueryUnfolding" ?strictQuery)

                ;;(update-model "rss" "looseQueryUnfolding" ?looseQuery)

                ;;;;(bind ?strictResults
                ;;;;(call-web-service  "OntologyReasonerWebService.jws"
"http://sweep.cs.tcd.ie:8080/ShanesReasoner/" "returnTerms" "strict" ?strictQuery)
                ;;;;)

                ;;;;(bind ?overallStrictResults (str-cat ?overallStrictResults
?strictResults " "))

                ;;;;(update-model "rss" "webserviceresult" ?strictResults)

                (bind ?looseResults
                (call-web-service  "OntologyReasonerWebService.jws"
"http://sweep.cs.tcd.ie:8080/ShanesReasoner/" "returnTerms" "loose" ?looseQuery)
                )

                (bind ?overallLooseResults (str-cat ?overallLooseResults ?looseResults
" "))

                ;;(update-model "rss" "webserviceresult" ?looseResults)

                ;;(cd "rss" "..")
        )


)

;;-------------------------------------------------------------------------------

(create-model "rss")

;;-------------------------------------------------------------------------------

;;;;(bind ?overallStrictResults " ")

(bind ?overallLooseResults " ")
```

```
(output-query2 "userModel" "completeConceptModel")

;;(update-model "rss" "narf" ?overallLooseResults)

(bind ?myLooseResults
(call-web-service "ReturnPersonalisedFeedWebService.jws"
"http://sweep.cs.tcd.ie:8080/personalisedNewsService/" "personaliseFeed"
"http://www.itv-f1.com/rss/latest/" ?overallLooseResults)
)


(update-model "rss" "channel" ?myLooseResults)


(store-model "myUserCollection" "rss" "myPNSModelTwoResult")</code>
</narrative>
```

Appendix III – Conceptual Model (Level of Interest Model)

```
<combined_models>
       <interest>
         <concept>Chassis</concept>

      <interest_type>
       <type>high</type>
          <relationship>chassisBuiltFor</relationship>
         </interest_type>

      <interest_type>
       <type>medium</type>
          <relationship>chassisBuiltFor</relationship>
         </interest_type>

      <interest_type>
       <type>low</type>
         </interest_type>
       </interest>



       <interest>
         <concept>Country</concept>

      <interest_type>
       <type>high</type>
          <relationship>
          </relationship>
          <relationship>
          </relationship>
         </interest_type>

      <interest_type>
       <type>medium</type>
          <relationship>
          </relationship>
          <relationship>
          </relationship>
         </interest_type>

      <interest_type>
       <type>low</type>
          <relationship>
          </relationship>
          <relationship>
          </relationship>
         </interest_type>
       </interest>



       <interest>
         <concept>Driver</concept>
```

```
<interest_type>
 <type>high</type>
     <relationship>drivesFor</relationship>
     <relationship>hasTeamMate</relationship>
     <relationship>hasCountryofBirth</relationship>
     <relationship>isCompetitorOf</relationship>
     <relationship>racesOnTrack</relationship>
    </interest_type>

<interest_type>
 <type>medium</type>
     <relationship>drivesFor</relationship>
     <relationship>hasTeamMate</relationship>
    </interest_type>

<interest_type>
 <type>low</type>
    </interest_type>
  </interest>



  <interest>
    <concept>Engine</concept>

<interest_type>
 <type>high</type>
     <relationship>engineBuiltFor</relationship>
    </interest_type>

<interest_type>
 <type>medium</type>
     <relationship>engineBuiltFor</relationship>
    </interest_type>

<interest_type>
 <type>low</type>
    </interest_type>
  </interest>



  <interest>
    <concept>RaceTrack</concept>

<interest_type>
 <type>high</type>
     <relationship>racedOnBy</relationship>
    </interest_type>

<interest_type>
 <type>medium</type>
     <relationship>racedOnBy</relationship>
    </interest_type>

<interest_type>
 <type>low</type>
    </interest_type>
  </interest>



  <interest>
    <concept>Team</concept>

<interest_type>
 <type>high</type>
     <relationship>hasDriver</relationship>
     <relationship>hasTeamPrincipal</relationship>
     <relationship>hasTechnicalDirector</relationship>
     <relationship>isCompetitorOf</relationship>
     <relationship>usesChassis</relationship>
     <relationship>usesEngine</relationship>
     <relationship>useTyre</relationship>
    </interest_type>
```

```
      <interest_type>
       <type>medium</type>
           <relationship>hasDriver</relationship>
           <relationship>hasTeamPrincipal</relationship>
           <relationship>hasTeamTechnicalDirector</relationship>
          </interest_type>

      <interest_type>
       <type>low</type>
          </interest_type>
        </interest>



        <interest>
          <concept>TeamPrincipal</concept>

      <interest_type>
       <type>high</type>
           <relationship>isTeamPrincipalOf</relationship>
          </interest_type>

      <interest_type>
       <type>medium</type>
           <relationship>isTeamPrincipalOf</relationship>
          </interest_type>

      <interest_type>
       <type>low</type>
          </interest_type>
        </interest>



        <interest>
          <concept>TeamTechnicalDirector</concept>

      <interest_type>
       <type>high</type>
           <relationship>isTeamTechnicalDirectorOf</relationship>
          </interest_type>

      <interest_type>
       <type>medium</type>
           <relationship>isTeamTechnicalDirectorOf</relationship>
          </interest_type>

      <interest_type>
       <type>low</type>
          </interest_type>
        </interest>



        <interest>
          <concept>Tyre</concept>

      <interest_type>
       <type>high</type>
           <relationship>tyreBuiltFor</relationship>
          </interest_type>

      <interest_type>
       <type>medium</type>
           <relationship>tyreBuiltFor</relationship>
          </interest_type>

      <interest_type>
       <type>low</type>
          </interest_type>
        </interest>
</combined_models>
```

Appendix IV – User Models

```
<user>
  <interest>
    <name_of_interest></name_of_interest>
    <depth_of_interest></depth_of_interest>
    <concept_of_interest></concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>
</user>
```

```
<user>

  <interest>
    <name_of_interest>BAR</name_of_interest>
    <depth_of_interest>low</depth_of_interest>
    <concept_of_interest>Team</concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>
  <interest>
    <name_of_interest>Ferrari</name_of_interest>
    <depth_of_interest>low</depth_of_interest>
    <concept_of_interest>Team</concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>
  <interest>
    <name_of_interest>Williams</name_of_interest>
    <depth_of_interest>low</depth_of_interest>
    <concept_of_interest>Team</concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>

  <interest>
    <name_of_interest>MichaelSchumacher</name_of_interest>
    <depth_of_interest>low</depth_of_interest>
    <concept_of_interest>Driver</concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>

  <interest>
    <name_of_interest>FernandoAlonso</name_of_interest>
    <depth_of_interest>medium</depth_of_interest>
    <concept_of_interest>Driver</concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>
  <interest>
    <name_of_interest>JensonButton</name_of_interest>
    <depth_of_interest>medium</depth_of_interest>
    <concept_of_interest>Driver</concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>
  <interest>
    <name_of_interest>JuanPabloMontoya</name_of_interest>
    <depth_of_interest>medium</depth_of_interest>
    <concept_of_interest>Driver</concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>

  <interest>
    <name_of_interest>FrankWilliams</name_of_interest>
    <depth_of_interest>medium</depth_of_interest>
    <concept_of_interest>TeamPrincipal</concept_of_interest>
    <source_of_interest></source_of_interest>
  </interest>

</user>
```

Appendix V – Interest Questionnaire

LEVEL OF INTEREST →	None	Low	Medium	High
TEAMS				
BAR	[ ]	[ ]	[ ]	[ ]
Ferrari	[ ]	[ ]	[ ]	[ ]
Jordan	[ ]	[ ]	[ ]	[ ]
McLaren	[ ]	[ ]	[ ]	[ ]
Minardi	[ ]	[ ]	[ ]	[ ]
Red Bull	[ ]	[ ]	[ ]	[ ]
Renault	[ ]	[ ]	[ ]	[ ]
Sauber	[ ]	[ ]	[ ]	[ ]
Toyota	[ ]	[ ]	[ ]	[ ]
Williams	[ ]	[ ]	[ ]	[ ]
DRIVERS				
Christijan Albers	[ ]	[ ]	[ ]	[ ]
Fernando Alonso	[ ]	[ ]	[ ]	[ ]
Rubens Barrichello	[ ]	[ ]	[ ]	[ ]
Jenson Button	[ ]	[ ]	[ ]	[ ]
David Coulthard	[ ]	[ ]	[ ]	[ ]
Giancarlo Fisichella	[ ]	[ ]	[ ]	[ ]
Patrick Freisacher	[ ]	[ ]	[ ]	[ ]
Nick Heidfeld	[ ]	[ ]	[ ]	[ ]
Narain Karthikeyan	[ ]	[ ]	[ ]	[ ]
Christian Klein	[ ]	[ ]	[ ]	[ ]
Felipe Massa	[ ]	[ ]	[ ]	[ ]
Tiago Monteiro	[ ]	[ ]	[ ]	[ ]
Juan Pablo Montoya	[ ]	[ ]	[ ]	[ ]
Kimi Raikonnen	[ ]	[ ]	[ ]	[ ]
Takuma Sato	[ ]	[ ]	[ ]	[ ]
Michael Schumacher	[ ]	[ ]	[ ]	[ ]
Ralf Schumacher	[ ]	[ ]	[ ]	[ ]
Jarno Trulli	[ ]	[ ]	[ ]	[ ]
Jacques Villeneuve	[ ]	[ ]	[ ]	[ ]
Mark Webber	[ ]	[ ]	[ ]	[ ]
TEAM PRINCIPALS				
Flavio Briatore	[ ]	[ ]	[ ]	[ ]
Ron Dennis	[ ]	[ ]	[ ]	[ ]
Luca Di Montezemolo	[ ]	[ ]	[ ]	[ ]
Eddie Jordan	[ ]	[ ]	[ ]	[ ]
David Richards	[ ]	[ ]	[ ]	[ ]
Peter Sauber	[ ]	[ ]	[ ]	[ ]
Paul Stoddart	[ ]	[ ]	[ ]	[ ]
Tsutomu Tomita	[ ]	[ ]	[ ]	[ ]
Frank Williams	[ ]	[ ]	[ ]	[ ]

TEAM TECHNICAL DIRECTORS

Bob Bell	[ ]	[ ]	[ ]	[ ]
Ross Brawn	[ ]	[ ]	[ ]	[ ]
Mike Gascoyne	[ ]	[ ]	[ ]	[ ]
Patrick Head	[ ]	[ ]	[ ]	[ ]
Adrian Newey	[ ]	[ ]	[ ]	[ ]
Willi Rampf	[ ]	[ ]	[ ]	[ ]
Gabriele Tredozi	[ ]	[ ]	[ ]	[ ]
Geoffrey Willis	[ ]	[ ]	[ ]	[ ]

TYRE

Bridgestone	[ ]	[ ]	[ ]	[ ]
Michelin	[ ]	[ ]	[ ]	[ ]

ENGINE

BMW P845	[ ]	[ ]	[ ]	[ ]
Cosworth TJ 2005	[ ]	[ ]	[ ]	[ ]
Ferrari V10	[ ]	[ ]	[ ]	[ ]
Honda RA 005E	[ ]	[ ]	[ ]	[ ]
Mercedes Benz FO 110R	[ ]	[ ]	[ ]	[ ]
Petronas 05A	[ ]	[ ]	[ ]	[ ]
RS 25	[ ]	[ ]	[ ]	[ ]
RVX-05	[ ]	[ ]	[ ]	[ ]
Toyota	[ ]	[ ]	[ ]	[ ]

CHASSIS

C24	[ ]	[ ]	[ ]	[ ]
EJ15	[ ]	[ ]	[ ]	[ ]
F2005	[ ]	[ ]	[ ]	[ ]
FW27	[ ]	[ ]	[ ]	[ ]
MP4-20	[ ]	[ ]	[ ]	[ ]
007	[ ]	[ ]	[ ]	[ ]
PS05	[ ]	[ ]	[ ]	[ ]
R25	[ ]	[ ]	[ ]	[ ]
RB1	[ ]	[ ]	[ ]	[ ]
TF105	[ ]	[ ]	[ ]	[ ]

Appendix VI – Evaluation Questionnaire

Q1 How much experience do you have using online News Services?

□ None □ Little □ Some □ Much

Comments ____

Q2 Are you comfortable using News Services on the Web?

□ Not at all □ Not very □ Quite □ Very

Comments ____

Q3 Did the news generated by the system reflect the answers you gave in the news questionnaire?

□ Never □ Rarely □ Usually □ Always

Comments ____

Q4 Did the news generated by the system reflect the news you wanted? (Hit)

□ Never □ Rarely □ Usually □ Always

Comments ____

Q5 Did the news generated by the system omit the news you wanted? (Miss)

□ Never □ Rarely □ Usually □ Always

Comments ____

Q6 Did the news generated by the system reflect news you did not want? (False Positive)

□ Never □ Rarely □ Usually □ Always

Comments ____

Q7 Did the news generated by the system omit the news you did not want? (Correct Rejection)

□ Never □ Rarely □ Usually □ Always

Comments ____

Q8 Did the news contain the content you expected?

□ Never □ Rarely □ Usually □ Always

Comments ____

Q9 Would you have found the ability to modify the user model (your interests) beneficial?

□ Never □ Rarely □ Usually □ Always

Comments ____

Q10 Did you find any difference between the news supplied to you in the two separate models?

□ None □ Not Significant □ Minor □ Major

Comments ____

Q11 Are there any additional features you would like to see in the Personalised News Service?

□ Yes □ No

Comments ____

Q12 Did you experience any technical difficulties with the course?

□ Yes □ No

Comments ____

Q13 Please use the space below to add any additional comments.

____

____

____

____

____

____

____

____

____

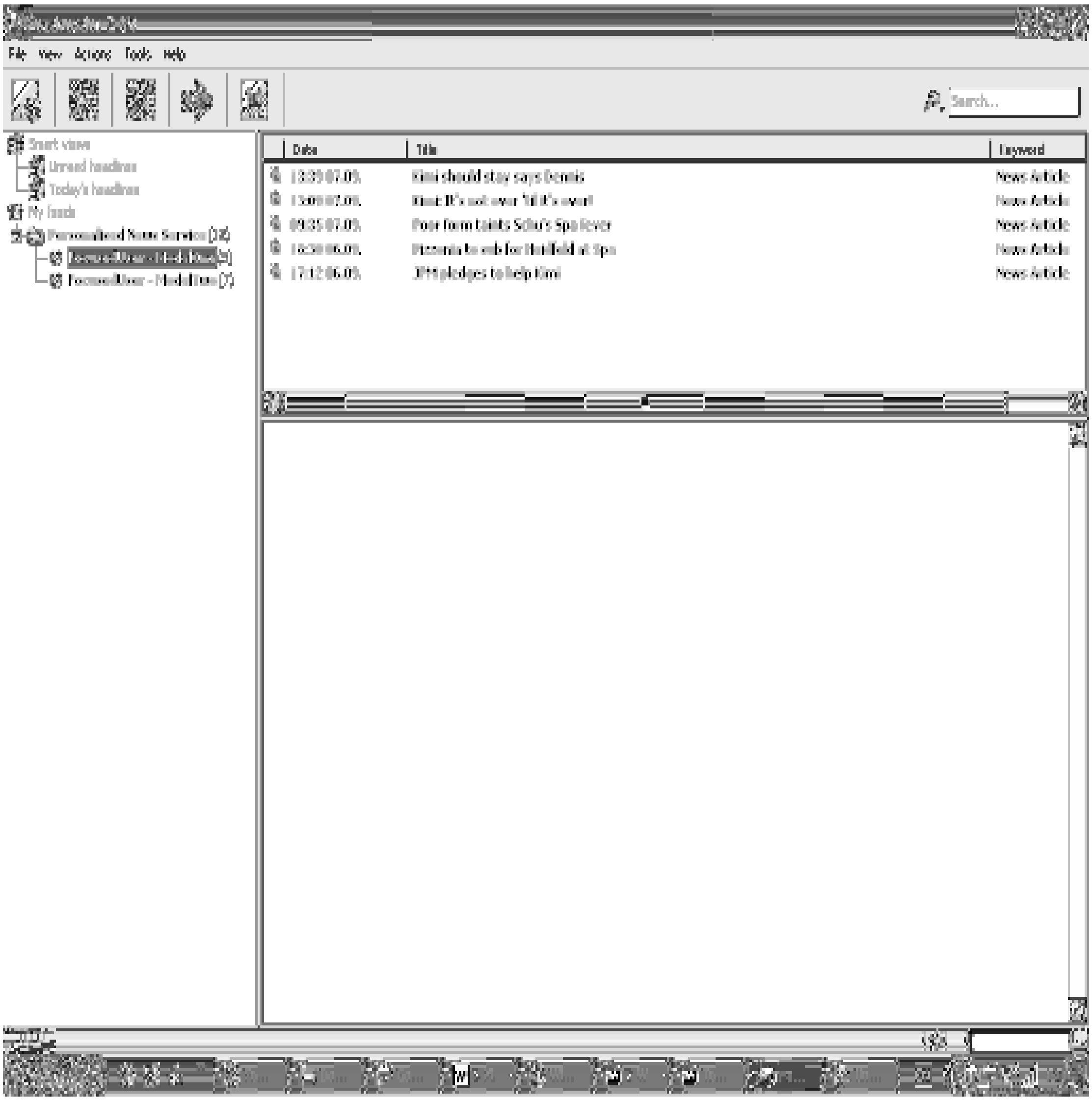
____

____

____

Thank you for answering the questionnaire.

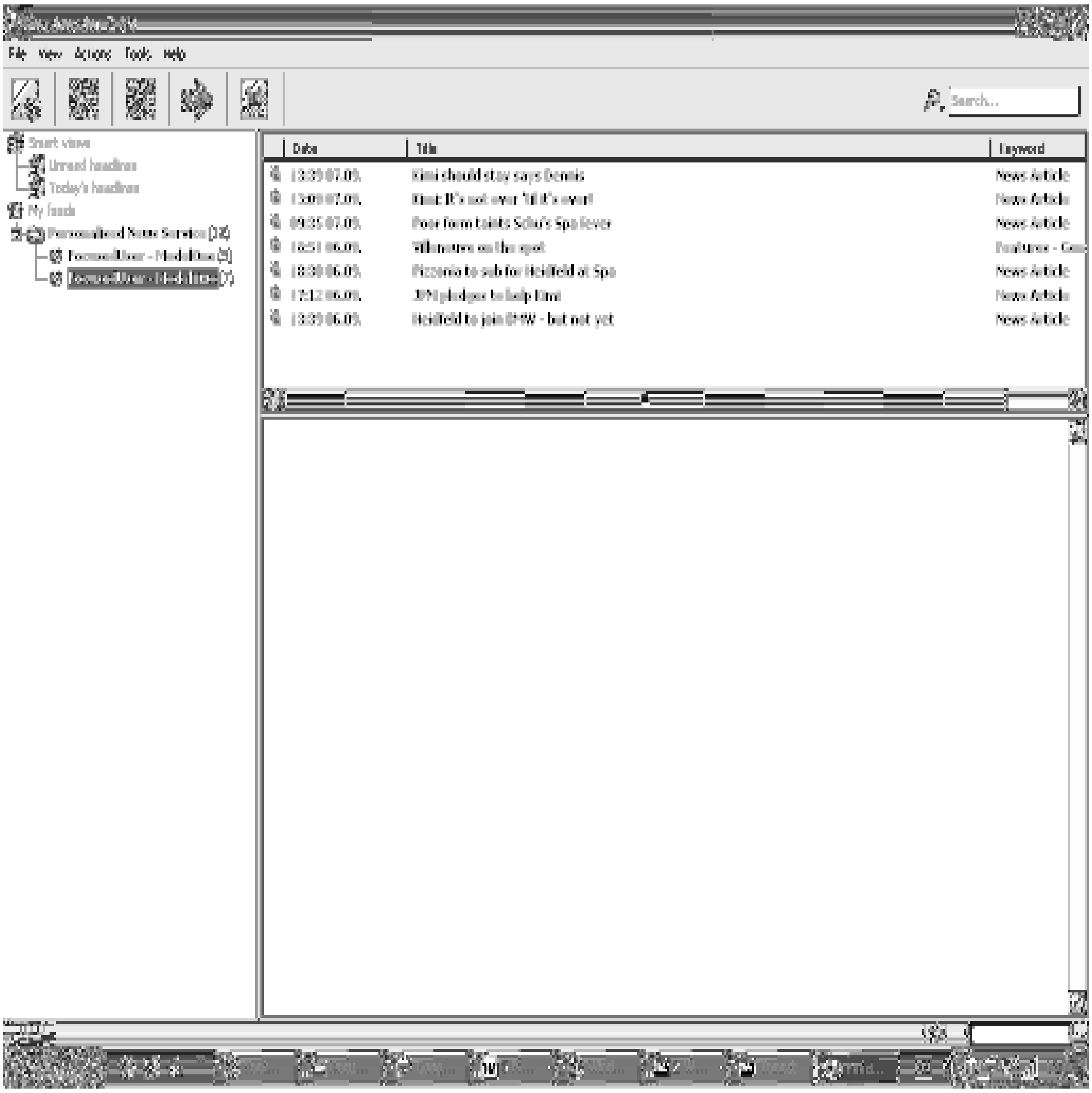
Appendix VII –Screen Shots of Feed Reader using the PNS



Date	Title	Keyword
13:39 07.09	Kimi should stay says Dennis	News Article
13:09 07.09	Kimi: It's not over 'til it's over!	News Article
09:35 07.09	Poor form taints Schu's Spa fever	News Article
16:30 06.09	Pizzonia to sub for Heidfeld at Spa	News Article
17:12 06.09	JPM pledges to help Kimi	News Article

Screen Shot: Model One of the Personalised News Service

This is a Screen Shot of the news feeds being provided to a user by the Personalised News Service. Focused User – Model One refers to the user who had seen a difference in the news they were getting between the two ontologies. Model One represents the Strict Ontology, and Model two represents the Loose Ontology. As can be seen here, there is a difference in the number of feeds received between the two ontologies. Two feeds appear in the loose ontology that do not appear in the strict ontology. These two feeds will be seen below in the Screen Shot for Model Two


File View Actions Tools Help
Search...
Smart views
Unread headlines
Today's headlines
My feeds
Personalised News Service (12)
PersonalUser - ModelOne (5)
PersonalUser - ModelTwo (7)

Date	Title	Keyword
13:39 07.06	Kimi should stay says Dennis	News Article
13:09 07.06	Kimi: It's not over 'till it's over!	News Article
09:35 07.06	Poor form taints Schu's Spa fever	News Article
16:51 06.06	Villeneuve on the spot	Features - Com
18:30 06.06	Pizzonia to sub for Heidfeld at Spa	News Article
17:12 06.06	JPM pledges to help Kimi	News Article
13:39 06.06	Heidfeld to join BMW - but not yet	News Article

Screen Shot: Model Two of the Personalised News Service

As can be seen, the two headlines, “Heidfeld to join BMW” and “Villeneuve on the Spot” are news feeds which do not occur in the Strict Ontology. This can go some way to showing the difference between the results provided by the two different ontologies.