Mechanisms for Context-Informed Adaptive Hypermedia

Alexander O'Connor BA, BAI

A dissertation submitted to the University of Dublin, Trinity College

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

September 2004

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Alexander O'Connor

Dated: September 2, 2004

Mechanisms for Context-Informed Adaptive Hypermedia

Approved by
Supervising Committee:

_____

_____

Permission to Lend and/or Copy

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

Alexander O'Connor

Dated: September 2, 2004

Acknowledgements

This Project would not have been possible without the guidance and assistance rendered by Vincent Wade, whose supervision provided a vital framework in the completion of the project. This project is based in software developed by Owen Conlan, without which there would have been no basis for investigation. At many of the most critical points in the course of the work, insightful comments and suggestions of a technical nature were offered by Ian O'Keeffe. The author wishes to extend his deep thanks to all of those named, and the many others who offered thoughts and suggestions during design and development.

Alexander O'Connor

University of Dublin, Trinity College

September 2004

Abstract

The Cross-fertilisation potential of the work being undertaken by both the Adaptive Hypermedia and Context-Aware communities is considerable. In particular, the benefits to Adaptive Hypermedia constitute a method for handling contextual influences without the need to extend deep models to accommodate axes of marginal interest or high complexity. The benefit to the field of Context-awareness lies in the opportunity to examine application areas where Context is not the only concern within the logic of the system, it is rather a set of influences on advanced, complex semantic structures.

This project is concerned with an initial investigation of the mechanisms for creating Context-Informed Adaptive Hypermedia, specifically in the area of influencing an educational Adaptive Hypermedia . A definition of context is provided, along with the design, implementation and evaluation of an example architecture.

The project was composed of three main phases: initially, a detailed survey of Adaptive Hypermedia and Context-Aware Systems was undertaken, this was followed by the outline of an architecture for Context-Informed Adaptive Hypermedia. Detailed design of the mechanisms for applying Contextual Axes to Adaptive Hypermedia was supplemented with the Prototyping and testing of these mechanisms with an extant eLearning Adaptive Hypermedia System.

A Paper Presentation on this project was given to the Workshop on Advanced Context Modelling Managing and Reasoning at the Sixth International Conference on Ubiquitous Computing under the tile *'Context-Informed Adaptive Hypermedia'* with authors Alexander O'Connor, Owen Conlan and Vincent Wade.

Contents

List of Figures

Chapter 1

Introduction

This chapter explains the motivation of the project, the general and specific questions which inspired this research. This is followed by an examination of the objectives of theu project, general and specific. Finally, a technical guide provides the reader with insight into the structure of this document.

1.1 Motivation

Adaptive Hypermedia[2] eLearning systems constitute the evolutionary advance from Intelligent Tutoring Systems and Hypertext. Adaptive Systems take the concept of a navigatable, digital educational course, and supplement it with deep models of the learner and course content. The *Adaptive Engine* of such a system contains business logic which makes use of these deep models to create a tailored course.

There is considerable variation in the forms which these models may take, and as Adaptive Hypermedia Technology evolves, additional factors supplement even more powerful rule sets, the factors or *axes* of adaptivitiy increase in number and complexity. This expansion of the capability of Adaptive Systems has the potential to yield considerable advantage, but there is also a considerable cost in terms of complexity and size of resultant systems.

Context-aware systems, on the other hand, are concerned with capturing, managing

and interpreting a wide variety of heterogeneous data about users and their environment from a number of sources, such as sensors and usage information. In general, this field has provided many useful techniques, but the example applications demonstrated are concerned with relatively simple tasks and processes. These applications are arguablr not sufficiently complex to demonstrate the true potential of the mechanisms provided.

The two categories of systems outlined above appear to show an overlap, where complex models may be expressed and measured using context-aware techniques, and the interaction between contextual information and deep model-based knowledge in the Adaptive Hypermedia System.

This project is motivated by the large amount of work being undertaken separately in both the Context-Aware and Adaptive Hypermedia Fields, and the potential for cross-polinisation.

1.2 Objectives

In general, the objective of this project is to examine Contextual Information and its potential for Adaptive Hypermedia. *Context-informed* Adaptive Hypermedia has the advantage of being able to include information from external sources, provided as the *Context* of the system.

Specifically, the objectives defined for this project are:

1. To conduct a survey of Context Aware Systems, their models and inputs. This includes recording the mechanisms used to apply measured data to the logic of the system, and an assessment of the characteristics of the different types of information handled by the system. This Survey is combined with a Survey of Educational Adaptive Hypermedia systems in order to determine a definition for context in Adaptive Hypermedia along with a list of descriptive attributes for effective use of Context. These attributes form part of the functional overlap between Adaptive Hypermedia and Context Aware systems.

2. To research and design mechanisms for collaboration between Adaptive Hypermedia and Context Awareness, based on requirements discovered in the surveys described, and others developed through the design process.

3. To prototype a set of collaborative mechanisms as designed above and test them with an example Educational Adaptive Hypermedia System.

1.3 Technical Approach

1.3.1 Document Outline

This document is structured around the objectives outlined. Initially, a survey of Adaptive Hypermedia and Context Aware Systems is presented, with analysis of the capabilities and attributes of these systems. This analysis gives rise to a number of suggested properties of context, along with a possible definition for context in collaboration with Adaptive Hypermedia. Chapter 3 is concerned with the Design of an integrated Context Information System for Adaptive Hypermedia. The thrust of this chapter is related to defining the guiding principles of the design, and presenting the challenges that these principles create. Major design decisions, and some possible alternate routes, are outlined. Chapter 4 contains an account of the third Objective of the project. This includes the relationship between the design and the concrete examples, a presentation of these examples, and possible alternatives. The evaluation of the system provided in Chapter 5 is derived first from the guiding principles enumerated, along with other technical and practical factors. Finally, the Conclusions of Chapter 6 embody an assessment of this project, as related to its objectives, along with suggestions for future work and overall impressions.

Chapter 2

Survey of Systems and Definition of Context

This chapter is intended to provide a survey of Context-aware Systems, a definition of Context and a survey of Adaptive Hypermedia Systems. This is supplemented with an analysis of the degree of functional and conceptual overlap between the categories of system, and the potential for the creation of a contextual element to an Adaptive System. In view of this aim, a definition of context is provided, to assist in determining the boundaries of each system.

2.1 Review of Context-Aware systems: Defining Context

In reviewing context-aware systems, the majority of them can be characterised as 'location-aware' applications. Most take a relatively straightforward view of context being largely contained as location, and attach a relatively simple business logic, such as messaging. Three examples of systems are provided, they follow a more general discussion on context, as seen in a number of publications over time.

2.1.1 General Views of Context

Context, in itself, is a difficult concept to characterise. Fundamentally, it appears related to the application domain in which the context-aware system is located. However, certain general principles may be drawn from current experience. In particular, there are three ways in which context can be viewed, as outlined below.

2.1.2 3 Major Questions

Three questions are defined, to provide an insight into context and its uses. These are intended to illustrate that contextual information, uninterpreterd, is of relatively little use. Contextual information provides a system with descriptive information, and is a means to a richer view of the world for that system.

2.1.2.1 What is 'Context'?

In the field of pervasive and ubiquitous computing, the definition of context is a well-worn path. In general, it can be said that these applications derive most if not all of their adaptivity and customisation from their axes of context. Nonetheless, these definitions provide a useful basis for deciding *what* context is.

Perhaps one of the more commonly cited definitions for context is[14]:

> 'Any information that can be used to characterise an entity, where an entity can be a person, place or physical or computational object.'

This definition covers an extremely wide scope, perhaps too great to be useful. Another[5] definition, provided as part of a survey of context-aware mobile systems, makes a useful distinction of *relevance* within the body of entity properties defined above:

> Context is the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user.

Again, this is a very wide definition, which encompasses a great many attributes, entities and relationships under a single heading. In fact, it can be said that the definitions delineated above provide a reasonably clear explanation of the concept of context, but they provide little assistance in recognising contextual properties.

In seeking to understand what context **is**, it is useful to examine what context **is not**:

1. **Context is not just Location**: Contextual Information can not be considered as simply location, nor can it be considered along any single axis of attributes or events. Context must be considered as a data set of inter-related dimensions.

2. **Context is not simply sensor data**: Context-awareness does not simply mean that a program or system simply accesses an additional sensor or data source in the process of execution. Context data has semantic significance beyond direct measurement.

3. **Context is a Semantic 'View'**: Drawing from the two preceeding points, it is clear that context-awareness amounts to the combination of a variety of axes of information, which may have undergone processing before they reach the client system.

The first aspect discussed was known relatively early[25], and has been repeated on a number of occasions(eg:[26]). Indeed, Schmidt *et. al.* draw an even more useful definition of context[26] as

> 'Interrelated conditions in which something exists or occurs'

This resembles the standard definition, however the authors expand on what these conditions constitute to include an aspect of 'level'. The example provided is most illustrative, and demonstrates that context can range from a direct measurement of temperature to an abstract concept such as 'In my Office' or 'At Home'.

It appears that context, as a concept, exists in a general form. Efforts to enumerate the axes of context do not provide appear to provide sufficient information to include

context in any way other than a 'hard-coded' or pre-determined and static form. This approach risks losing a large proportion of the potential of context-awareness.

On the other hand, the more general definitions still provide no method for identifying contextual attributes, they particularly fail to distinguish context from other forms of input data, which may or may not fall into the contextual 'situation'.

With regard to Adaptive Systems, which already contain a situational model of sorts, the requirement for a clearly defined boundary becomes even more important. Since Model-driven adaptive systems employ 'interesting entity relationships', the general definitions above are not sufficient. Nor, indeed are the enumerated models, as the axes of context and their implications alter substantially with each model/narrative composition.

It can be seen that context may well constitute the remainder of the data available for adaption, that which the system is capable of providing, but which the model employed does not make primary use of. In this scenario, the designation of context for a particular axis can be seen as being given on a case-by-case basis.

2.1.3 Where can Context be Used?

Despite the difficulties of locating a specific definition for context, it is still possible to examine the possible uses for context and context-like attributes.

Schilit *et. al.* define four primary categories for applying contextual information:

- **Proximate Selection** defines a method for determining the elements of choice most relevent to a user's location. For example, the system might prioritise using a printer that is in a location closest to the user. It can be seen that the concept of 'Proximity' can be employed with metrics other than distance by location.

- **Automatic Contextual Reconfiguration** proposes that applications alter their composition based on contextual boundaries. This might include loading or un-loading modules, creating network connections or power management related features.

- **Contextual Information and Commands** This set of functions gather many of their parameters from context. The semantic significance of these parameters might include keywords such as “Here”, “Now” or “This device”. The power of these commands lies in providing the user transparency with regard to the interface, while offering powerful functionality.

- **Context-triggered Events** In some sense, the reverse of the previous item, these are actions within programs which occur independently of user interaction, based on some set of contextual conditions. The authors suggest that these events are suitable perhaps for implementation in a rule-based model.

These categories broadly cover not only contextual information, but also other model-driven adaptivitiy mechanisms. Where context drives the system entirely, this is sufficient. However, when working in concert with other models, it is necessary to describe context in an alternate fashion.

In this case, context can be described as a set of attributes which combine together, and with the adaptivity models, to refine the selection or presentation of an already-relevant model.

2.1.4 Where can Context be Gathered?

The infrastructure described in [14] is now a well-known example of the method for gathering and processing contextual data as employed in ubiquitous/pervasive applications.

The authors define the main design considerations in a context-aware system. These include:

1. **Generalisation:** Requiring applications to directly handle contextual data creates excessive requirements for developers. In addition, a direct strategy engenders a system which is limited in respect of expandability and dynamicity.

2. **Separation of Concerns:** Axes of context and other input measures may become confused if there is no discrete separation of concerns between contextual data and other input.

3. **Onus of measurement:** Dependant on applications, it may be desirable to employ either a polling mechanism, or some form of event based context update. Hybrid solutions may also be applicable.

It is evident from these concerns that an abstracted context layer is the only practicable method for gathering and processing contextual data. This requirement is reinforced by the possibility that raw context data may require significant processing before it is employable by any system.

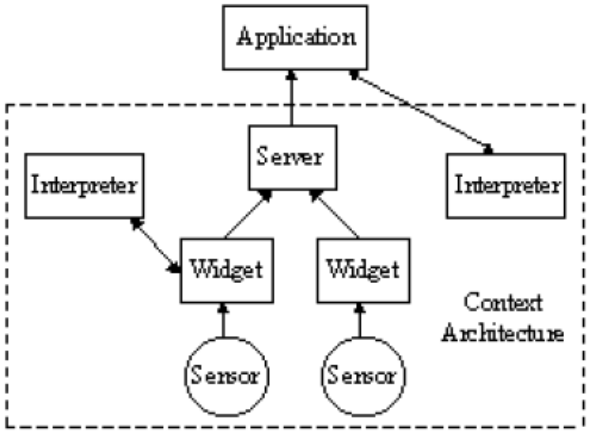

Fig. 2.1: Data flow between application and context infrastructure. **Source:** [14]

The authors describe an abstract **Context Widget**, which is connected to a sensor. This widget provides a number of useful features in the base class, including the ability to provide data either by responding to polling or by indicating changes with events. The widget also provides the very useful feature of potentially storing historical contextual state information via a relational database. Context Widgets can be subclassed to permit specific features such as, for example, more intelligent caching.

Contextual data is treated and processed via **Interpreters**. These objects can interface either with context widgets directly; alternatively they can be accessed by

applications seeking perhaps to treat contextual information combined with other parameters.

Context Servers permit designers to aggregate contextual information in a reusable fashion. In addition, the aggregation paradigm permits the use of heterogeneous data gathering (via Interpreters), as well as reusability and a reduced service discovery requirement[1].

The benefits of this model are clear, primarily the advantage is one of light-weight integration and simplicity of use. However, the authors also acknowledge a number of limitations of the system:

1. **Quantisation:** The system described has a no built-in method for dealing with continuous data, such as GPS location information. Handling the quantisation of this data is most important where the context correlation load is high, or where small changes to this data are unimportant.

2. **Inconsistency:** Perhaps the most difficult feature of context management is the issue of inconsistent or erroneous data. The primary guards against this are to provide soft failure mechanisms in the abstraction, caching to permit the retrieval of the last valid state and the aggregation of heterogeneous sensors for redundancy.

3. **Service Discovery:** Only limited discovery mechanisms are provided, and details of the network address of the various entities is required. Superior transparency is vital to a more portable system.

With regard to adaptive systems, the gathering of context raises a number of important issues and indeed highlights some of the potential weaknesses which are less prevalent in context-only systems.

In general, the point of interest for contextual data is at adaptive content generalisation. It is here that contextual data will be combined with other model information to select content. This favours a polling model of context gathering.

[1]The application must only find Server(s), not each individual widget and interpreter.

However, it is also true to say that contextual decisions polled at generation may become stale over time. It is therefore desirable for a mechanism to exist to permit the update of selection based on change in context. This favours an event-based model.

Combined, these concerns outline the requirement for a hybrid method of context measurement.

It can also be said that adaptation is a computationally costly process, and considerable inconvenience may result if the engine is forced to re-evalutate its selection excessively. This is particularly true where the resultant adaptation does not significantly differ from the previous iteration.

From these concerns, we can infer that context in an adaptive system should not include information that is critical to the adaptation. This concern is derived from the fact that a contextual abstraction is not a completely modelled entity, but rather a measurement of an abstraction.

Furthermore, context is better suited to those entities which are either large or highly dynamic, as they provide the Adaptive Engine a means to ignore unimportant changes while retaining an interest in the dimension measured.

Architecturally, these arguments combine to indicate that, architecturally, context should be considered as an abstract layer, with a variety of points of contact within the adaptive engine.

2.1.5 ActiveCampus

2.1.5.1 Overview

The ActiveCampus suite consists primarily of two tools[22]:

1. **Graffitti:** a web-based location aware bulletin board system. Users are able to 'tag' locations with information, to be retrieved by others who are nearby.

2. **Messaging:** provides instant messaging between users, permitting them to send messages in realtime.

2.1.5.2 What is Context?

In [21], Griswold *et. al.* indicate that they employ the definition of context outlined in [14], with a number of extensions. This results in the definition of context as a set of entities, selected and processed, in order to provide supplemental information about a task transparently[2].

The main feature of this definition is that it encompasses more or less the entire body of state information for a user (from 'buddy icon' and 'user status' to 'current location'[21]).

2.1.5.3 Where can Context be used?

Contextual data is employed in this system to provide all state information. The system employes **Proximate Selection** in Messaging by listing users which are nearby, and in Grafitti by displaying only nearby tags. There is little or no support for **Automatic Contextual Reconfiguration**. **Contextual Information and Commands** do exist in the form of the Grafitti tagging process, which depends on current location. **Context-triggered Events** take the form of alerts when other users enter the proximity, or upon user status change as well as upon message receipt by the Messaging system.

2.1.5.4 Where can Context be Gathered?

The ActiveCampus Architecture takes the form of a 5-layer model. The main paradigms involved include a highly centralised client-server relationship, with context forming one of two functions external to the server (the other being user-end data representation).

The contextual model is highly object orientated. Sensor data is marshalled into 'Entity Objects', which are typed to assist in determining entity relationships. Furthermore, there is a process of 'Service Introspection' in order to assist in discovery. This provides a convenient pluggability and expandability. However, the developers experi-

[2]This definition resembles the goal of the Aura Project: 'Distraction-Free Computing'[9]

enced 'Entity Bloat' when they attempted to expand or alter services. This arose from their failure to distinguish entities from their representations.

An indexing model was therefore implemented, and this also permitted the use of index-keyed caching of information.

2.1.5.5 Analysis and Comments

The ActiveCampus model appears to merge the conflicting goals of separation of concerns with integration and performance quite successfully. However, the definition of context provided has no substantive outer bound. By their definition, all state data is, to a greater or lesser extent, contextual. In this author's opinion, this is one of the key reasons that the 'Entity Bloat' problem was felt so keenly, as non-contextual data (ie: internal user model data) was confused with contextual information.

It is also likely that the authors' acknowledged decision not to tackle the issue of radically different display mechanisms may engender difficulties for certain axes of context.

2.1.6 Context-Aware User Authentication

2.1.6.1 Overview

In a pervasive computing environment such as a hospital, there is considerable inconvenience in attempting to maintain current 'username/password' login methods, particularly for the large number of embedded devices. The goal of this system[1] is to provide secure access to devices in pervasive computational environments.

2.1.6.2 What is Context?

Primarily, context is expressed as location, which is used to verify the process of logging in using a Radio Frequency Idendifier (RFID) smart card read on a particular device. However, simple location is, in this case, used as a more useful concept of 'proximity'. The difference is that proximity has semantic importance to the system, and is

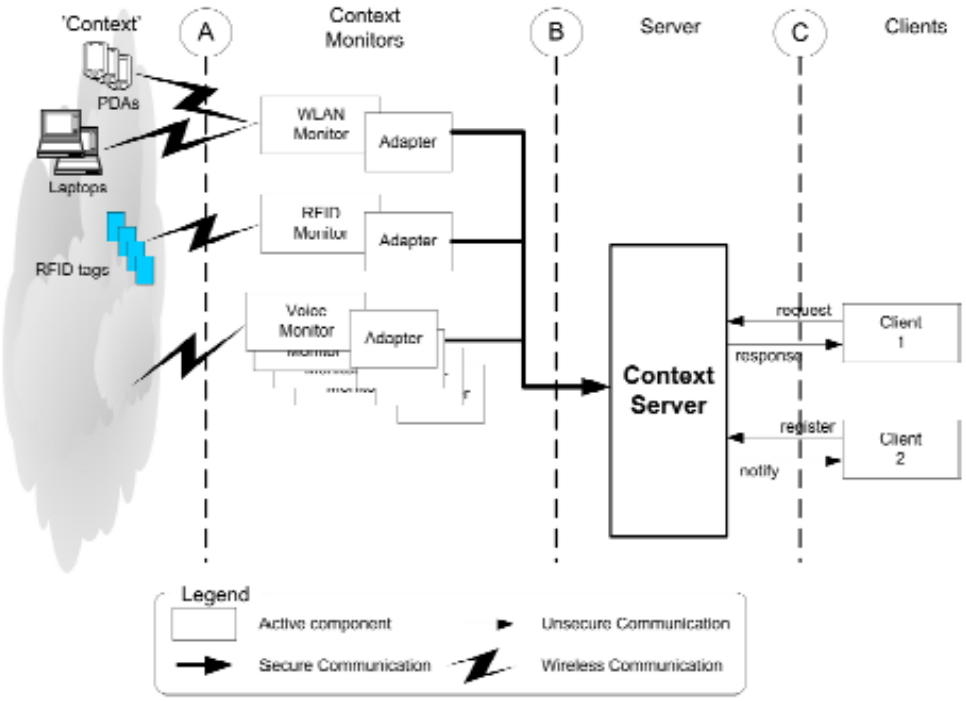

Fig. 2.2: Location-aware login architecture **Source:** [1]

interpreted from location information. Direct measurement of location is performed in a number of ways, along with the use of digital schedules and diaries to produce a confidence value of the location of the user.

2.1.6.3 Where can Context be Used?

Contextual proximity is used to verify the location of the user within a probability. This data is then used in the decision whether or not to grant access to the user for a particular device.

2.1.6.4 Where can Context be Gathered?

Context Monitors are sensors employing a variety of techniques including RFID, WaveLAN and voice detection to determine the location of a user. This data, in addition to schedule information, is analysed in the Context Server, which sends a probability to

the authentication system.

2.1.6.5 Analysis and Comments

The proximity concept as outlined above yields insight into the relationship between context and more complex systems. The two components of context, Monitors and Server, create an interpreted response by correlating numerous factors unknown to the business logic of authentication. In other words, the contextual factors are encapsulated as a probability, which is the shared medium between the systems. This is a simple but highly expressive medium for communication of contextual factors.

2.1.7 MOBIlearn

2.1.7.1 Overview

The MOBIlearn project is a 'generic mobile learning architecture', where context-awareness is employed to select content based on the learner's goals, situation and resources[24].

2.1.7.2 What is Context?

The MOBIlearn authors do not characterise context in direct terms. Instead, context is considered as a

> set of changing relationships that may be shaped by the history of those relationships

In order to provide a somewhat more concrete model, the following hierarchy is defined (*cf. fig:2.3*).

- **Context** is the overall situation over a range of values (including time, space, etc...)
- **Context State** Is the instantaneous (with respect to the above values) set of entities relevent to the user's task.

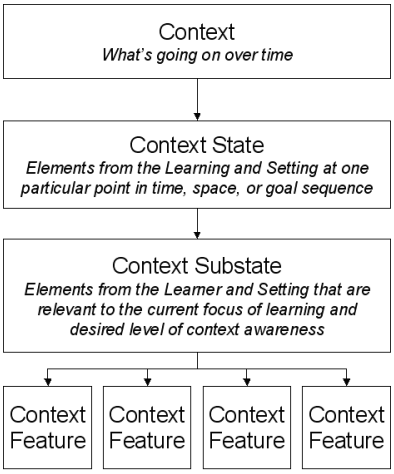

Fig. 2.3: The MOBIlearn context hierarchy. **Source:** [24]

- **Context Substate** is the set of entities that are contextually relevent to the user focus.
- **Context Features** are the individual entities within the Substate set.

Contextual selection depends on constructing the Substate and analyzing the Features therein.

2.1.7.3 Where can Context be used?

In this system, the context substate is employed to generate a meta-data model of the current situation. This data is then matched to meta-data tagged content for retrieval.

In terms of Schilit's[25] criteria, the context usage is therefore purely on the **Proximate Selection** dimension, with a metric based on meta-data matching.

2.1.7.4 Where can Context be Gathered?

A variety of specific sensor solutions are discussed. In general it can be said that expandable and abstract properties of this system are embodied by assigning a Context Feature to a sensor (*c.f.* Context Widget in [14]).

2.1.7.5 Analysis and Comments

Once again, this context model indicates an intentionally broad set of what context may be. However, the reference to a 'Learner' model (which can be compared to a position in an educational narrative) provides a very useful insight, despite forming part of the context implicitly.

This distinction, though partial, of Learner and Setting, points to a key property of context: that, though inter-linked, it is separate and distinct from model data.

Furthermore, this system differs from the other examples because it demonstrates a complex logic: that of a Mobile Learning Application. However, this project views its models as portions of a larger 'contextual' view. This method is perhaps derived from the mobile nature of the application domain, but it does not differentiate deep adaptive models from other contextual influences. In addition, context is seen to cover all levels, from the low-level of sensor information to conceptual implications, all in one system.

2.2 Review of Adaptive Hypermedia Systems

This section of the document attempts to review a number of Adaptive Hypermedia systems in the educational sphere. The analysis is provided based first on an overview, followed by a discussion of the models contained within the system. The method whereby these models are combined is then described. This is followed by an examination of the mechanics of presenting the resultant adapted content. Some comment and analysis is provided, with a particular interest in the contextual possibilities of each system.

2.2.1 AHA!

The AHA! Project is the educational adaptive hypermedia project of the Department of Computer Science of the University of Eindhoven. The lead researcher is Professor

Paul De Bra.

2.2.1.1 Overview

Originally created as a conditional link and content hiding system, the AHA! model has evolved through three major versions to an xhtml compliant[3] hypermedia adaptation system.

2.2.1.2 Learner Model

The AHA! 'User Model' can be considered as a vector of *concepts* with *attributes*. These concepts are typed either Boolean, String or integer[13]. The act of accessing content will alter this vector, generally to increase a *knowledge* attribute.

2.2.1.3 Content Model

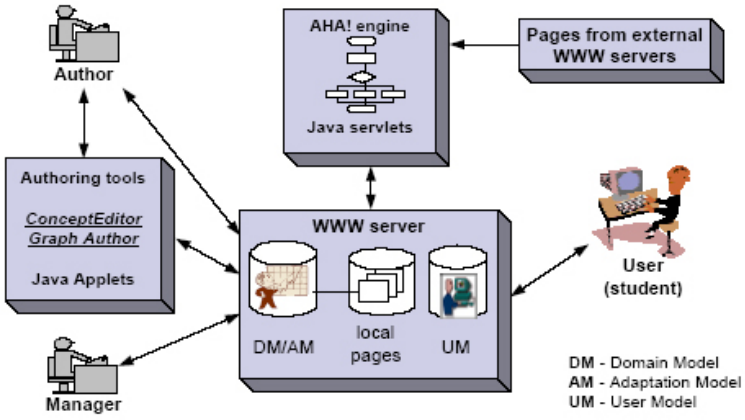

Fig. 2.4: AHA! Architecture **Source: [13]**

Discrete portions of content are represented as *concepts*. These can be linked to any number of *pages*, *objects* or *fragments*. Each concept within the AHA! system approximates a single constituent unit of the Domain. Examples given include a style of painting or a topic to be studied[13].

[3]When the correct tags are employed

The concepts and attributes contained by the Domain Model are reflected in the User Model. This is an example of the **Overlay Model**:

> For each domain model concept, an individual student's knowledge model stores some value which is an estimation of the knowledge level of this concept.[3]

2.2.1.4 Adaptation Mechanism

The adaptation mechanism of the AHA! model is related to the concept overlay. Upon visiting a page, a number of concept-linked attributes within the user model are altered according to embedded values within the concept ruleset. The two most common associated attributes are the *visited* and *knowledge* attributes.

Each page is composed of fragments and objects which can, themselves, contain fragments which require adaptation. This creates a mechanism for recursively creating complex pages[4].

AHA! pages can require certain attributes before a page is accessible, therefore embedded fragments can be used to enforce requirements in pages.

2.2.1.5 Presentation

AHA! exposes two primary mechanisms of adaptive presentation(*cf. fig.2.5*). The first is through link hiding/annotating, the second is via fragment inclusion/selection.

In both processes, as discussed above, the system evaluates the *desirability* of a page, relative to the requirement set of the page and the user's knowledge vector, particularly the *visited* attribute.

The link annotation process colours link text based on three categories: good, neutral and bad. In fact, 'bad' annotated links may be concealed entirely from the user.

[4]This also permits the incautious author to create infinite loops within the system. One mechanism for preventing this is an arbitrary resolution depth.

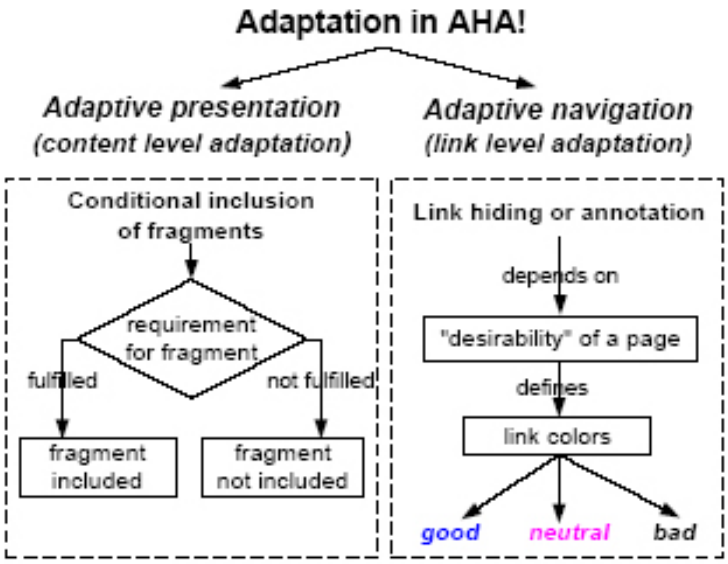

Fig. 2.5: Presentation of Adaptation in AHA! **Source:** [13]

Object inclusion acts on a similar basis. However, in this case the candidate content is a selection of fragments (or objects), the most appropriate of which is embedded in the page.

2.2.1.6 Content Authoring

Given the low-level expression of the AHA! system, authoring tools are vital to the creation of useful content. Two principle tools are employed[13]:

- **The Concept Editor**: Employed to annotate content with an attribute set.
- **The Graph Author**: Permits the high level creation of concepts, pre-requisite sets and knowledge propagation.

2.2.1.7 Analysis and Comment

In providing co-located rules and content, the AHA! system provides a very low-level control for the user. However, this model does not encourage reusability, nor does it

readily accommodate expansion and alteration. A more modular method for content navigation control may be useful.

The AHA! system provides significant control over its adaptation mechanisms. It is possible for the author to control the stability of the system at an object level. There are four levels of control: **always adapted**(default setting), **always stable**, **session stable**, **expression stable**.

It would, perhaps, be useful to empower the user to make a similar choice with regard to the reconfiguration rate of the system, thus placing change control at the desire of the user, instead of enforcing potentially considerable dynamicity.

The authors of the AHA! system indicate that an 'important idea' would be to examine expanding beyond 'user behaviour' into the area of context 'such as device and network characteristics'[13].

Integrating context into the AHA! system is assisted by the adaptable nature of the attribute vector. However, the combined rules and content would compound the difficulty of expressing context in a 'pluggable' manner, creating instead new axes of adaptation.

2.2.2 InterBook

Originally intended to break the paradigm of simply transferring day-to-day course content to the web (with an associated tangle of links), this system was designed to assist in creating a responsive and adaptable distance learning scheme based on web technologies.

This system is based on the metaphor of an 'electronic textbook'. This provides a hierarchical, ordered structure to the content to be presented.

2.2.2.1 Learner Model

Once again, the overlay model is employed to represent the knowledge state of a user. However, in addition to the overlay model as presented in AHA!, this system permits

the definition of 'learning goals', sets of attributes it is intended the user acquire over the course of the system's operation.

In addition to page visits, the Learner Model in this system takes account of factors such as quiz results and problem solving[3].

2.2.2.2 Content Model

The Interbook Domain Model is based around a set of concepts, 'elementary pieces of knowledge of the domain'[5][3]. This concept space is arranged hierarchically within a *glossary*.

Each concept constitutes a node within the glossary, and each node constitutes a hypermedia document. Concepts are arranged within *books*, which are arranged within *bookshelfs*, consisting of several books on the same topic.

Each book contains a *spectrum*, consisting of the pre-requisite data for the book, along with its content. Thus, a book can be considered as a function with inputs (the pre-requisites) and outputs in the form of an expression of the knowledge gained.

2.2.2.3 Adaptation Mechanism

Once again, the overlay model is employed to map the Learner model to concept requirements. Content selection is based on the sequence of learning goals[6] and their constituents.

Two primary adaptation schemes are presented, which take some account of learning style. The first is to provide **Direct Guidance**, where the system rates and arranges nodes within the glossary. The overlaid model permits the system to rate links as *ready to be learned*, *not ready*(pre-requisites not met) and *known*. In addition, the system can track whether or not a node has been *visited*.

The second learning model is to present **Prerequisite-based Help**. This model can perhaps be most useful in assisting a student with difficult or ill-understood topics.

[5]similar to the AHA! definition

[6]each goal consisting simply of a pre-determined set of concepts to be learned

In this mode, the system lists and ranks the pre-requisite links for a topic, presenting them for the student's perusal. A topic rank is based on the number of relevant concepts addressed. This model lends support for a back propagated mode of learning.

2.2.2.4 Presentation

The Interbook interface is arranged around two main windows:

- **The Glossary Page**: The glossary relationships are designed to resemble the semantic organisation of the concepts and this provides the primary navigation method.
- **The Textbook Window**: Consisting of three frames the Navigation Bar, the Concept Bar and the Text Window.

In the process of Direct Guidance, the Concept Bar is employed to list both the pre-requisite and resultant concepts of a particular topic, as presented in the Text Window. In addition, the Navigation Bar contains various standard buttons employed to navigate the text. This includes the portion of the glossary related to the topic, with links annotated for rating via colour (red for not ready, green for ready, white for topics which yield no new information. A checkmark icon indicates the visited status of the topic.

2.2.2.5 Content Authoring

Content authoring in the Interbook system is performed in a number of stages:

1. The content for a concept is written and marked up in Microsoft Word, based on the style mechanism[7] provided.
2. This annotation is augmented with conceptual relationships, which are generated via the use of pre-defined blocks[8] that provide information on pre-requisites and resultant knowledge.

[7] eg the document Title marked in the Title style
[8] These are hidden from view in the Word document.

3. These Rich Text Format files are converted to HTML, and interpreted automatically to create hypertext nodes and LISP rulesets.

2.2.2.6 Analysis and Comment

Though the rule specification system of Interbook is basic, this system has a number very desirable features with regard to user interface. In particular, the explicit listing of pre-requisites and requirements, along with a simple and direct visual representation of the adaptation state.

There is no specific mention of user context within the Interbook system. In particular, the fact that rulesets are generated purely on the basis of input and output concepts makes the introduction of non-conceptual factors in adaptation difficult without major revision.

However, the interface within the system does provide a useful clue for a more 'intelligent' adaptivity: the effect of any contextual input on the system must *empower* the user's choices, rather than simply removing control. This is particularly true in deep cognitive tasks such as learning.

2.2.2.7 'AHA! meets Interbook'

From inception, the AHA! system was considered an 'assembly language'[12] for adaptive Hypermedia. In the course of development, it was decided that the AHA presentation system should be generalised to produce new functional possibilities. This includes, for example, the presentation of an Interbook Electronic Textbook as an AHA! course.

The main feature of this process was to alter the method of presentation of the content representation scheme, to permit type classing of conceptual elements in an arbitrary fashion. In addition, normal integrated entity relationship information was exported to separate xml documents. The product of this research was a more generalised AHA! system, with an example Interbook to AHA! compiler.

This provides a significant assistance to the introduction of context to the AHA!

model. By providing separate rule descriptions, it would be feasable to introduce contextual rule sets to the descriptions, via perhaps an additional compiler.

This method points to perhaps the most important feature of context, that it does not form core membership of the adaptation model, but instead provides supplemental information to be employed once content has been selected.

2.2.3 KnowledgeTree

The KnowledgeTree Architecture[4] is designed to combine just in time content provision and re-usability with adaptive eLearning techniques.

2.2.3.1 Overview

Knowledge Tree is a distributed eLearning architecture, composed of one *Portal* per Learner, connected to a network of *Activity Servers* and a *Student Model Server*.

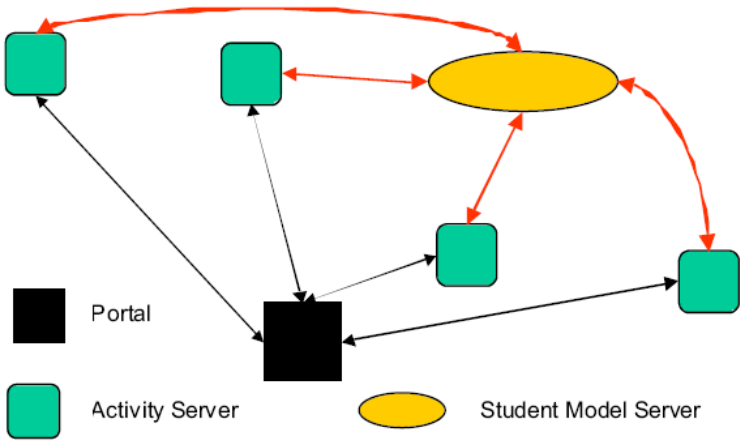

Fig. 2.6: Knowledge Tree Distributed Architecture **Source:** [4]

2.2.3.2 Learner Model

The Learner model includes student performance and learning history over a number of portals. Each learner model is stored in a **Student Model Server**, which can

either be located on the local Learner's machine or centrally, for example in the case of a University.

2.2.3.3 Content Model

Content is either backed by metadata annotations, or can be based on fuzzy text searching. The content is grouped in **Activity Servers**, which provide static or dynamic content as required.

The Teacher makes use of the **Portal** to designate learning goals for a particular section or complete course. This is then automatically adapted.

2.2.3.4 Adaptation Mechanism

Adaptation takes place in the Portal, based on the goals laid down by the course author and from runtime adaptive information about the content, the user's preferences and the user's history from the Model Server.

2.2.3.5 Content Authoring

Content can either be metadata annotated in advance, or sourced by fuzzy text searching. A simple URI (Universal Resource Indicator) based protocol is used to exchange resource locations.

2.2.3.6 Analysis and Comment

This system employs a wide variety of advanced techniques to give reuse to the learning objects. However, it does not address the use of 'external' factors which would normally be described as contextual. In some ways, this system is quite highly centralised, the Learning Portal holds a large portion of the functionality, and there is scope to further distribute this architecture with the inclusion of remote contextual reasoning during runtime adaptation.

2.2.4 APeLS

This system can be summarised as the application of the highly separated models of Intelligent Tutoring Systems to the field of Adaptive Hypermedia. The project is under way at the Knowledge and Data Enginerring group of the Department of Computer Science, Trinity College under the leadership of Vincent Wade.

2.2.4.1 Overview

The APeLS system can perhaps best be characterised by the fact that it does not employ the same model format as the two systems mentioned previously. Instead, this system separates Content, Learner Model and adaptation ruleset (entitled 'Narrative Model'[8]).

2.2.4.2 Learner Model

The Learner model in APeLS consists of a profile of discrete assumptions about a Learner. Of primary importance to the system are the *Prior Knowledge* and *Learning Goals* of the Learner. These factors are populated both from within the system, and mainly via the use of pre-determined online tests. The Learner Model is designed as an extensible framework, which is accessed via a vocabulary of *concepts*.

2.2.4.3 Content Model

The content Model in APeLS is composed of Learning Objects, arranged in 'pagelets', these conceptual units are annotated with an extended version of the IEEE LOM metadata format[8]. This provides a re-usable set of content, with arbitrary granularity and an extensible metadata annotation.

Annotations to pagelets include such features as *competencies taught*, *competencies required*, *learning style*

2.2.4.4 Narrative Model

In separating content from information flow, the APeLS system employs the Narrative Model to express those charactersitics of information to be learned not addressed directly through the content data. In particular, this permits the Learner model's expression of preferred learning style to be accommodated without necessarily requiring new or altered content. Instead, content expressed in a style-neutral form can be re-ordered and presented in a more suitable format.

A particular Narrative will not refer to specific pagelets. Instead, *candidate content groups* of pagelets, each referring to the same Learning Object. This permits the introduction of runtime variables into content selection.

2.2.4.5 Adaptation Mechanism

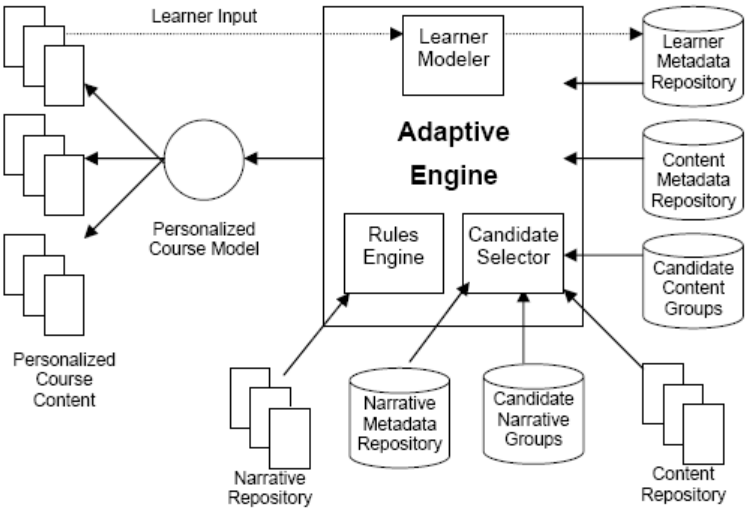

Fig. 2.7: Architecture of APeLS **Source:** [8]

The APeLS adaptation model consists of the correlation of the narrative model with the candidate content groups. Specific selection is made based on the state of the learner at the point of adaptation.

There are two primary selections to be made. The first is to select the narrative

most suited to the User, and the second is to select content based on that narrative.

Throughout the system, there is a correlative vocabulary of terms to relate entities within each model, particularly in regard of the Learner, Narrative and Content Models.

Certain external factors may be brought to bear on the selection of content/narrative. For example, content presentation and selection will be affected when the student is interested in a revision course rather than a long course of instruction [8].

2.2.4.6 Content Authoring

An additional substantial bonus of this architecture is that different authors can create the detailed content, based on their knowledge of the domain and narratives, based on their knowledge of paedagogical and information theory.

The standards-formed basis for expressing much of the metadata within the system also assists in interoperation and importation of content[8].

2.2.4.7 Analysis and Comment

The separation of the Narrative Model from the Content in the APeLs system provides a very useful, dynamic method for the introduction of Contextual Semantics. In particular, this is true due to the fact that the Narrative makes reference only indirectly to content, permitting an 'external factor' such as the capacity of the client terminal[11].

The complexity and possibly considerable load of model integration also opens the opportunity to 'lighten the load' on the Adaptive Engine by removing the need to model directly non-core concerns. This retains the separation of concerns within different models[9] while providing significant enrichment to the axes of adaptivity of the system.

[9]For example, it cannot be truly stated that a Learner has *Bandwidth* , but rather that their client system does. It is therefore desirable to avoid modellig this within the domain of the Learner Model, but to relegate it to a contextual framework.

2.3 Analysis

The separate domains of Context-Aware and Adaptive Hypermedia Systems appear to contain an overlap. In comparing their properties below, this overlap will become visible, as a clear need to supplement conventional adaptive models with context.

2.3.1 Properties of Context-Aware Systems

Context-aware models can be viewed as two primary components: the context-gathering and interpretation module, and the business logic module. Context-gathering and interpretation is concerned with measuring contextual information (such as location, but also time, terminal type and many other factors) and providing it in a relevant form to the business logic module, which communicate with an agreed degree of shared knowledge. The state of the overall system is composed of input from both modules, the initiative for change coming from both the context and the business information.

2.3.2 Properties of Adaptive Hypermedia Systems

Adaptive Hypermedia systems are centered around direct, deep modelling of entities such as the Learner and their requirements, and the Content and its attributes. These models are correlated on the basis of potentially complex learning style routines, with many factors in each decision point. The addition of further factors for consieration requires further explicit modelling, and integration into the overall ruleset.

2.3.3 Definition for Context-Informed Adaptive Hypermedia

Based on these properties, it can be seen that the area of potential for context lies in the provision of additional factors to the adaptation which are not central to the domain process[10], but which provide value-added service to the overall system. The goal of the integration is to permit a variable number of extra factors to be accounted,

[10]Otherwise, they would need to be included in the core models

without the need to model them explicitly, or integrate them into carefully-defined models. Therefore, there must be a relatively simple method for influence of context to be permitted into the various aspects of the system, while remaining sufficiently expressive. In addition, there may be some more important factors to the adaptation which are simply impractical or undesirable to model directly in-system.

2.4 Conclusions

In summary, Context-informed Adaptive Hypermedia benefits from the simple, expressive sharing of dynamic or extraneous data to permit value-added improvements to an adaptation. Context is:

> The axes of knowledge which, though not defined *a-priori*, provide useful additional input to the core models of adaptation.

These systems often attempt to provide complete 'vertical-market' coverage, by modelling a specific domain completely, within their own idiom. By combining the properties of the 'deep' adaptive models and the 'dynamic' contextual models, it is intended that a more hoizontal, broader system be developed.

Chapter 3

Designing Context-Informed Adaptive Hypermedia

This chapter relates the factors affecting the design of the mechanisms for Context-Informed Adaptive Hypermedia. The overall design is of a separated model: where Contextual axes are managed by a separate element of the system to the Adaptive Engine, which is concerned with it s core models. There were a number of factors which governed the process of design, and these are enumerated and outlined. Specific Design choices are then Examined, with a discussion of some alternate possibilities, and the reasons for not choosing them. While the design presented is aimed at with the APeLS Adaptive Engine, the analysis portion of this chapter will include a discussion of some of the over-riding principles, which would lead to a generalised system.

3.1 Guiding Principles for Design

The design of Context-Informed Adaptive Hypermedia was undertaken with the APeLS[8] architecture in mind. However, the basis for the design, its fundamental characteristics, are applicable to many other domain and systems.

Certain guiding principles were identified in the process of the Survey(*cf. 2*). Others were defined during the process of analysis, where the behaviour of the system

was considered in detail. The resultant principles of design can be viewed under the following headings:

1. **Autonomy**: The degree to which the Context Interpreter and the Adaptive Engine can make decisions and change the state of information on their own initiative. This is desirable as contextual parameters are likely to change over time in a potentially unpredictable pattern, while the process of adaption requires the ability to make considerable changes based on its axes. This principle is concerned with minimising interference during changes effected by each system.

2. **Simplicity**: Fundamentally, the addition of contextual sensitivity to the system is aimed to reduce complexity and increase the power of the system by value-added means. It is therefore required that the contextual mechanisms not create a significant overhead in the specification of narratives, or the design of the course.

3. **Expressiveness**: Exchanges between the Contextual Elements and the Adaptive Engine must be representative of the information being exchanged. The degree of expressiveness of any dialogue must be sufficient to permit a useful transfer of information for a range of contextual influences, from a small alteration to a concept list to a key decision in the path of the narrative.

4. **Shared Knowledge**: It is vital that both portions of the system operate on agreed terms. There must be the minimum risk of a miscommunication between the Adaptive Engine and Context, that would produce an error. For example, in reference to the term 'golf club' both systems must be in agreement that they refer to the instruiment of play, not the organisation. However, this shared knowledge should also not place excessive restriction on the Context Module in particular, where a more complete model of an area might require more detailed knowledge.

5. **Obfuscation**: The Adaptive Engine should ideally be unaware of the method by which the Contextual information is gathered, and the contextual decisions are made. Similarly, the Context Elements of the system should not attempt to

supplant the deep models of the Adaptive Engine. This is a key principle of the separated model of design: it is vital that one side not attempt to 'second guess' the other.

6. **Encapsulation**: The encapsulation of a large variety of different information as a functional interface to the Adaptive Engine is dependent on obfuscation, and on transparency. There should be identical methods for exchange, no matter the contextual capabilities of the Context Elements.

7. **Range**: One of the reasons to preclude an axis from core adaptation is that it evolves an extremely large range of values, the specific variation of which has little importance. For example, location has a large variance, which for most educational courses the specific value is irrelevant. However, the inference of, for example, proximity to a particular entity is a useful application of location.

8. **Frequency**: The process of adaptation is a complex one, and rapid changes in input which would not engender significantly useful alterations to the result should be minimised. Contextual Elements should be capaple of converting rapid changing (and, combined with the previous item highly variable) inputs to substantive changes for the Adaptive Engine where necessary and ignoring changes where the input change is marginal.

9. **Importance**: The involvement of Context as a 'peripheral' set of information should not be taken to mean trivial information. The Context Element may supply core and may add key points to the overall state of the system. However, without these additions, it must still be possible for the Adaptive Engine to create a useful result. The contextual information adds to this default result, enriching the adaptive process.

10. **User Empowerment**: In principle, there is no reason why contextual axes could not be employed to supply all the model parameters for an adaptation. This **Zero-Knowledge Adaptivity** suggests that the models within the Adaptive

Engine are effectively empty, and that all information is supplied by the Adaptive Engine. While, in theory, this is something of a breach of the concept of context as peripheral entities, it is a practical and perhaps even useful method for performing adaptation where the Adaptive Engine is critically short of information.

However, while this and other forms of adaption are valuable, it is important that the Learner remain involved and empowered. In particular, the educational domain is dependent on considerable attention and concentration from the user, depending on the requirements of the task. Bloom's Taxonomy of Learning defines a number of categories of skill and types of learning[6]. In particular, *Cognitive Learning* (the main category of learning supplied by the Adaptive Engines, divided into six types of learning, some of which are passive and some of which are active. In the case of higher cognitive tasks, greater active involvement is required. While this is primarily reflected in the course as managed by the Adaptive Engine, it is also desirable for the Context Interpreter to respond to the active input of the Learner[1].

3.2 Design Challenges

In addition to the principles outlined above, there were a number of functional and non-functional requirements for the design of the Context Elements.

3.2.1 Web Architecture

The standards-driven approach visible throughout the design of APeLS is a strong encouragement to continuing a cohesive architecture. In addition, the wide potential of the separated architecture necessitates transparent location and interaction with Context Elements. These requirements lead to the implementation of the system using web-services and xml-oriented transport.

[1] A Similar requirement can be observed[7] in the theory of Andragogy[23], where adults in education prefer to take significant control of their learning

3.2.2 Applying Context over the Entire System

The principle of Encapsulation and Obfuscation, as well as the drive to simplicity, encourages the design to give maximum access to the Contextual Elements. In the case of APeLS, the main models are the **Learner Model**, the **Content Model** and the **Narrative**. These models each contain different aspects of the adaptation process, all of which may be subject to contextual alteration.

3.2.2.1 Accommodating Narrative Style

Narratives in APeLS are specified in Jess[2][20], a Java-based implementation of a LISP-style Rule Chaining System. Narratives can therefore either be structured in many ways, including as procedural or as a rule-chain. It is important, for simplicity, that narrative-related contextual mechanisms support a wide variety of styles.

3.3 Overall Design

This section presents an overall architecture for Context-Informed Adaptive Hypermedia. Later chapters will explain in more detail the Prototyping and Implementation of this design, as well as its evaluation.

3.3.1 Context Interpreter

The Context Interpreter greatly simplifies the architecture and use of Context by encapsulating all Contextual Elements, and managing shared knowledge and expressiveness by translating information from Context into terms usable by the Adaptive Engine directly. The Context Interpreter is the endpoint for the Adaptive Engine for all Contextual Mechanisms. It is important to note that, under the principles outlined above the Context Interpreter is not itself an adaptive Engine, but rather an aggregator and broker of contextual information.

[2]Jess is a trademark of Sandia National Laboratories

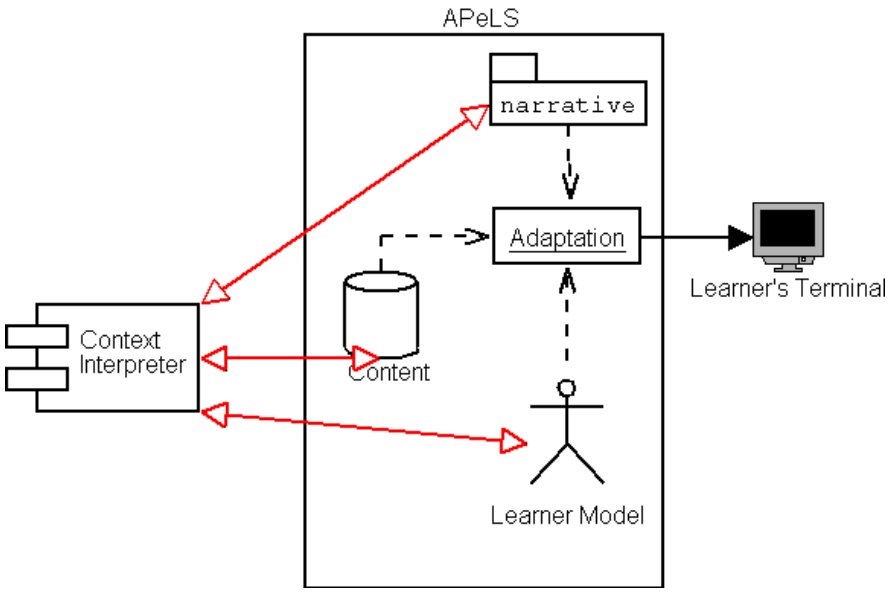

Fig. 3.1: The Influence of the Context Interpreter on the Adaptive Engine

3.3.2 Contextual Mechanisms

One of the primary goals of this project is to define a set of Mechanics for the exchange of contextual information with the Adaptive Engine. The mechanisms outlined below reflect the design phase of this objective.

3.3.2.1 User Model Enrichment

This form of contextual input is initiated by the Context Interpreter as and when changes to the context arise. In principle, the User Model can be considered as the set of knowledge open to the Adaptive Engine for reasoning. The intention of this form of update is to permit Context to enrich the relevant user details by reasoning over data not accessible to the core adaptivity. An example of this form of input might be for the Context Interpreter to discover that the user has undertaken a project involved in a particular subject. This information might have bearing on the learning goals of the user, and so this fact (expressed in terms relevant to the adaptive engine) may be taken into account by the adaptive engine in selecting content such as examples. In

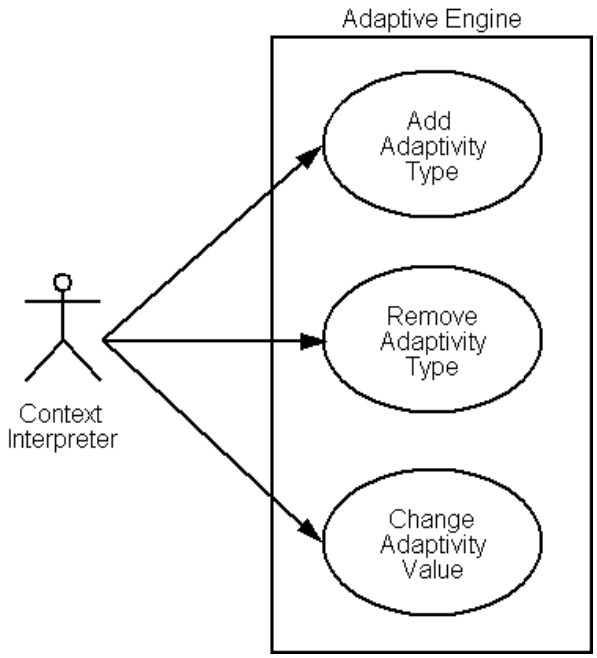

Fig. 3.2: Use Cases for the User Update Mechanism

this example, the user model is enriched with the Learning Context of the User, that is, the process of making connections between topics for users.

3.3.2.2 Narrative Decision Enhancement

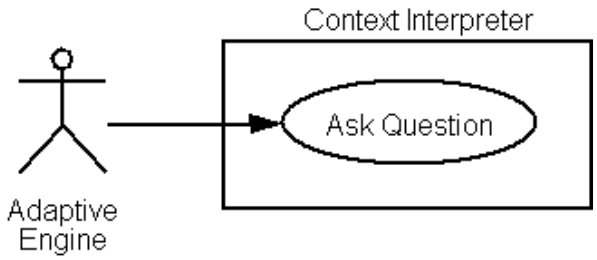

Fig. 3.3: Use Cases for the Narrative Decision Enhancement Mechanism

This form of context pertains to enriching the decision process of the adaptive engine. This decision process can be considered as the navigation of the concept space

as described by the narrative. This can be used to supplement choices during adaptation with concerns not available to the Adaptive Engine. This mechanism can be considered as a 'handover' or 'hook' for black-box contextual input, and permits the narrative to include the option of improved, context-enriched information, in a way which is transparent to the description of the narrative. In addition, this manipulation of the concept space (e.g.: learning goals) and the content space (e.g.: candidate group selection) can be controlled in some fashion, at a minimum by permitting the author of the narrative to decide when the hooks are activated. In view of the open nature of narratives, support must exist for a variety of choices, such as either/or choices and aggregated choices, which may reflect on several items in the narrative.

3.3.2.3 Narrative Decision Enhancement

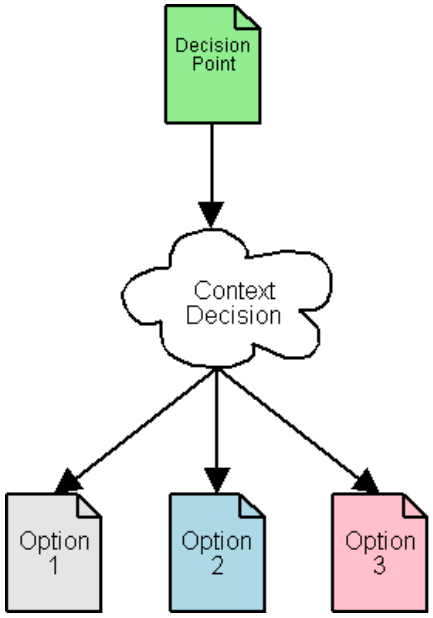

Fig. 3.4: A 'Broken' Narrative, where a decision must be made to continue.

This mechanism permits the creation of 'broken' narratives, where a contextual choice is required to decide the path of the narrative. These do not breach the extrinsic nature of context, as they are intended for use where a random selection would be

identical from the perspective of the Adaptive Engine.

3.3.2.4 Candidate Group Manipulation

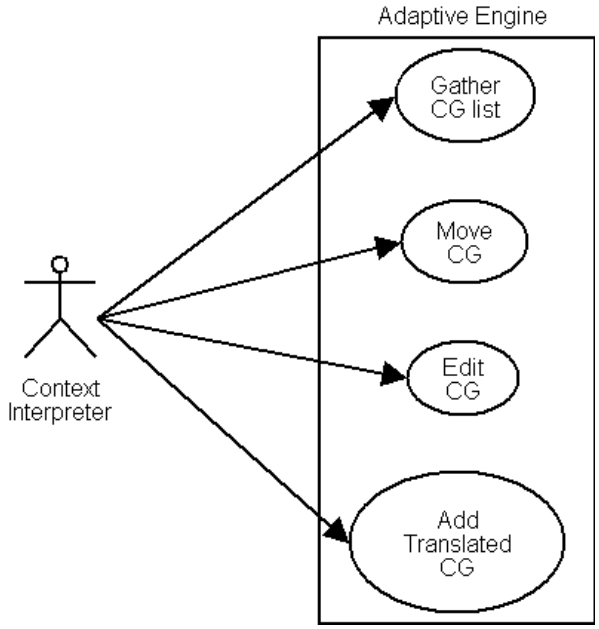

Fig. 3.5: Use Cases for the Candidate Group Manipulation Mechanism

Candidate Groups support the principle of Candidacy[10], whereby they are elements of the course which do not refer directly to content, but rather to a set of similar pagelets of content. These Candidate groups are themselves grouped with concepts to form portions of the course, and the manipulation of this list yields a high-gain method for introducing unknown context to the Adaptive Process.

3.3.2.5 Sub-Concept Content Selection

Similarly, it may be useful to support the candidacy process by permitting the list of pagelets derived from a Candidate Group to be manipulated by the Context Interpreter. Conceptually, this is extremely similar to Candidate Group Manipulation, from the viewpoint of exchange mechanics.

3.3.3 User Interaction

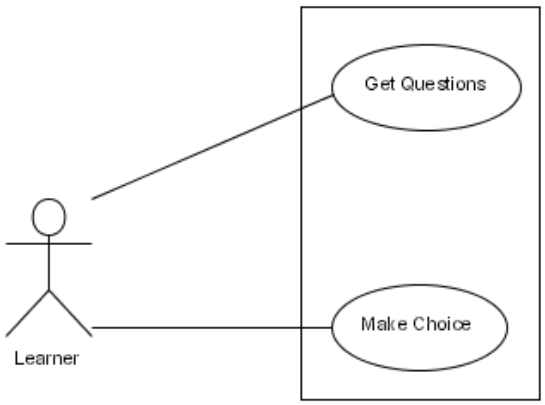

Fig. 3.6: Use Cases for the User Empowerment

The process of contextual influence of adaptation could benefit greatly from user support in the form of feedback and correction of decisions. Through this means, a user preference profile can be developed. This mechanism takes the form of the presentation of questions, sourced from the Context Interpreter and displayed on adapted pages.

3.4 Design Decisions

There were a great many choices to be made in the process of design. Some of the most important ideas are outlined below, with reasons for the choice made.

3.4.1 Direct Manipulation

One possibility was to view the system as a number of co-operating agents, altering the state of a conceptual information space. This could perhaps be viewed as a system where the Context Interpreter makes direct changes to the stored metadata of the Adaptive engine, as required.

The advantage to this model is that there is unrestricted access from the Context Interpreter to the Adaptive Engine. There is no need to define specific mechanisms,

all changes can be made as required by the decision logic of the Context Interpreter.

However, this is a fundamental breach of separation, and has a number of disadvantages relating to this. First and foremost, the timing of such alterations may risk being haphazard. With no co-ordinating entity, concurrent access to the database may result in partial updates, unless great care is taken. Secondly, the process of directly manipulating the database blurs significantly the line of separation between the Adaptive Engine and the Context Interpreter. As they both manipulate the structures as they see fit, their functionality merges, and the system becomes one, large Context-aware Adaptive Engine. This is not as simple, nor does it permit Encapsulation or Obfuscation, finally the separation advantages with regards to Importance, Amplitude and Frequency are potentially lost.

3.4.2 Query Language

A more complex query language, with greater shared knowledge, would permit each system to interrogate the other in detail, and permit gains in complex contextual and adaptive scenarios. However, on the other hand, such queries are by their very nature defined in advance, and requiring of very considerable degree of agreed knowledge. This risks the simplicity of the system, as it also potentially affects the expressiveness of the system, by reducing the Context Interpreter to a sensor interaction layer. It is preferable to make use of agreed conceptual and sub-conceptual terms in order to exchange data, as this further eliminates the need for the Adaptive Engine to expend significant resources to marshall and integrate contextual inputs.

3.4.3 Total Automation

In keeping with the principle of User Empowerment, it is important to permit the system to notify the user of their actions, and permit the learner to alter decisions made by the system. While it is not desirable or practicable to let the learner alter every decision, a web-oriented explanation of selection and presentation of options is

helpful in maintaining interest.

3.4.4 Conclusions

This chapter has presented the Design of Context-Informed Adaptive Hypermedia. The separated architecture, composed of an Adaptive Engine and a Context Interpreter was defined, along with the mechanisms by which the Adaptive Engine and Context Interpreter may exchange information, as well as the principles on which these exchanges take place. The general principle of a Context Interpreter is apllicable to many other domains, where a system can include extra input axes without prior specification through permitting a Context Interpreter to access the models involved. The use of the 'vernacular' by the Context Interpreter in exchanges permits a highly pluggable model, where translation between different domain systems might occur in the realm of contextual enrichment.

Chapter 4

Implementation

This chapter covers the third Objective of the project, the prototyping and testing of the mechanisms outlined previously for applying contextual axes to the Adaptive Engine.

The initial sections of this chapter are concerned with a description of the general implementation, followed by a presentation of the creation of test examples for different applications of the mechanisms to courses.

4.1 Implementation Description

The objective of this implementation was to examine the process of enabling the input of the Context Interpreter to be accounted for by an existing Adaptive Engine. The system in question was the APeLS system outlined previously.

4.1.1 Overview

The mechanisms described in the Design chapter of this document had to be applied to the architecture of APeLS. The APeLS system is based on Apache Tomcat[16], where it is composed of a set of Java Server Pages, which provide a framework to receive adaptive content from the Engine, written as a java library. The content and user

metadata is sourced from an XML Database[1]; this data is composed based on rules specified in the LISP-like language of Jess. The creation of this system was therefore divided into three main subdivisions: the Context Interpreter, the User Module and the Narrative Module.

In order to provide and open, standards-compliant transport method, it was decided that the Contextual mechanisms should be implemented as methods for a Context Interpreter Web Service. SOAP[15] was chosen as the protocol, for remote invocation of a Java-based service. The Apache Axis[17] package was employed for automatic marshalling and transport.

Each mechanism was mapped to a method exposed in the web service, the specific processing was done by a class implementing the appropriate interface[2].

The other advantage to this implementation method, apart from its simplicity and standards-compliance, was that it permitted the use of JSPs on the Context Interpreter side, to provide for user feedback confirmation.

4.1.2 Context Interpreter

A simple framework for test scenarios was needed in order to permit queries to be evaluated. The structure of the Interpreter was arranged to provide easy alteration in the event of revised test cases.

The principle class of the system selects and instantiates specific implementations of the abstract Interpreter Interface. This interface reflects the exposed methods of the overall service.

All methods in the system have a space to provide the endpoint, the URI of the target Context Interpretation Webservice. In the event of none being specified, a default value is chosen.

[1] Apache Xindice[19]

[2] See the appendix for UML information

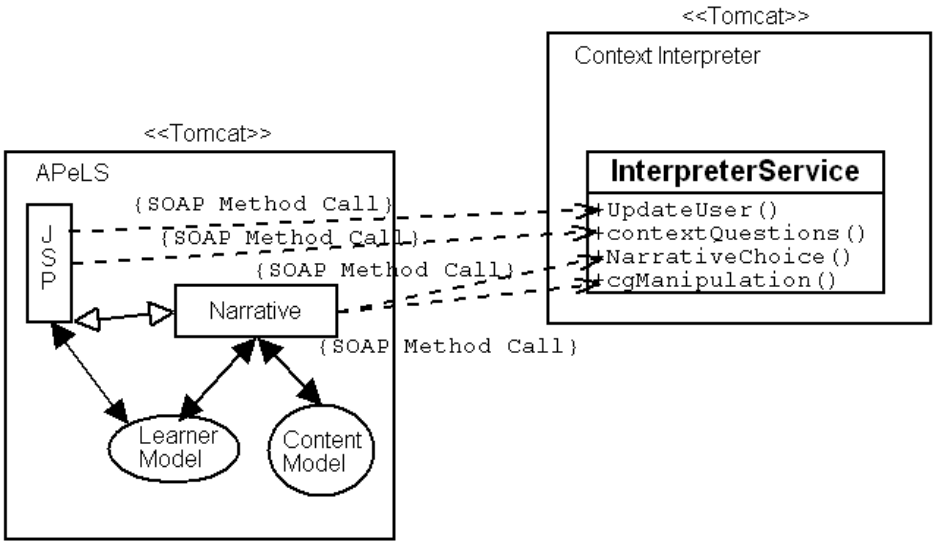

Fig. 4.1: Implementation architecture. Mechanisms are mapped to methods in the Interpreter Web Service. The User Module is called from JSPs directly, while the Narrative Module is called from Jess, during adaptation. These modules marshall, unmarshall and transport queries and responses.

4.1.3 User Module

Written as a Java Class, this Module provides the interface between the JSP controlled sections of Adaptivity and Context. In particular, this Module is responsible for marshalling and exchanging User Model Learner Objects with the Context Interpreter. Invocation of the Adaptive Engine is controlled in JSPs, and so it is desirable to be able to manipulate the User model before the Narrative is executed. This is reflected as the **updateUser** method of the InterpreterService Object.

The code to update a User therefore depends on passing an intitial User Model to the Context Interpreter, which processes and updates the adaptivity matrix of the User. This is expressed in the Object as a Hashtable of Vectors, each associated with an adaptivity type. For example, a Learner might have several *competencies.learned* to reflect their previous learning experience.

JSP file

```
1 //Initialise the interface, with default endpoint.
2 UserSOAPInterface soapIf =
3    new SOAPInterface("http://target.server/InterpreterService.jws")
4 //call the Context Interpreter
5 //learnerObject is the Learner Model before Context is applied
6 learnerObject =
7    soapIf.updateUser(learnerObject, "identifier","narrative", null);
```

This module also provides methods for retrieving the XML document expressing the Context Interpreter's Feedback. The **contextQuestions** method of the InterpreterService returns a matrix of questions as raw xml, which can then be transformed by the system. It is left to the Adaptive Engine to perform any transforms to create the resultant xhtml.

4.1.4 Narrative Module

Most narratives in APeLS are implemented as a function for adding nodes to a document tree, which forms the resultant course, by comparing certain attributes of the Learner to certain attributes of the Candidate Group in question. The Narrative is also responsible for specifying candidate selection from candidate groups. The Narrative Module is therefore concerned with providing functionality for Jess instructions, to permit access to the Context Interpreter.

Three Mechanisms, User Update, Narrative Choice and Content Group Manipulation are implemented within the **call-context** Jess UserFunction. This is a wrapper object, provided by the Jess libraries to permit the embedding of Java code within Jess. These functions process the ValueVector of parameters from the system to select the correct call to the Web service and pass the converted data. It should be noted that the preferred usage for the User Update mechanism is with the use of the JSP based interface. In the example code below, which rule is asserted is determined by

the value of the first string returned from the list passed to the endpoint.

course.clp

```
(assert-string
 (nth$ 1
  (query-context "narrativeChoice"
    (create$ "(userKnowsjava)" "(userKnowsperl)")
"http://localhost:8080/axis/InterpreterService.jws" "narrative")
   )))
```

The results of all the Narrative Module function calls are String-typed and valid for use througout Jess. In particular, Lists are returned for all but the UserUpdate function, which does not return information as the Narrative does not directly interface with that model.

Examples of usage for some of the other mechanisms are provided later in the chapter.

4.2 Main Challenges

There were a number of significant challenges in the implementation of the system. Of prime importance was handling the apparently variable initiative between the Context Interpreter and the Adaptive Engine. During design, it was deemed important that the Context Interpreter be able to make changes to the models in response to changes in Context.

However, as implemented, the system is dependent on the Adaptive Engine to call the Context Interpreter Web Service. This is consistent with the Web Architecture, in responding to the User input for updates and changes. Responsive changes can in fact be implemented relatively simply: each adaptation includes its Contextual features and so all changes are taken into account with each new update. More detail on this factor will be provided during the evaluation of the system in the next chapter.

4.2.1 Data Types

There is a great range of data employed in the process of adaptation, from the features of the Learner and the Attributes of Content to the Concept Space manipulated during the progress of the Narrative.

4.2.1.1 Transporting Information

Both Jess and SOAP place constraints on what types of data can be used for their functions. In particular, it is desirable that lists[3] be employed, where possible in data.

SOAP requires that data take the form of Java primitives, or arrays thereof, so that they may be marshalled into exchangeable formats. In the interests of portability, it was decided to be preferable for the Table of Vectors in the Learner Object to be converted to a 2-Dimensional Array. This required the addition of two methods to the Learner Class, one to retrieve and one to set the adaptivity table.

4.2.2 Candidacy

In principle, the Narrative is concerned with the manipulation and arrangement of *Concepts* and their associated Content Groups. The manipulation of the sub-conceptual content model is not a prime role. However, in practice, it is desirable to use the power of Jess to select content.

This presented a challenge to the design of the context mechanisms regarding the desirability of separate methods for manipulating content groups, and content group membership. In the end, it was found that, from the perspective of transport and marshalling, the difference was functionally *nil*. Both can be considered as simply lists of entities to be manipulated and returned by the Context Interpreter.

[3] or multi-fields

4.3 Experiments

A number of usage scenarios were outlined in order to test the interaction of the mechanisms with the Adaptive Engine. The scenarios are all based around two adaptive courses, one concerning SQL, the other concerning an introductory mathematics course.

The use of two courses provided some useful benefits, as they each treat the data and engine in a distinct fashion. The Mathematics course, for example, rebuilds the course each time the user has viewed a topic. On the other hand, the SQL course has a more developed sense of candidacy, but is less frequently rebuilt.

The Scenarios are divided into three categories, Low-, Mid- and High-Context. This delineation is based on the *importance* of the contextual decision, ie. the input it has on the state of the adaptive models. In all cases, the Adaptive Engine can produce a course without the decision, or with a trivial (random) choice and from the perspective of the core models, all requirements are fulfilled.

In order to demonstrate the capabilities of the mechanisms, certain scenarios are presented in different forms, with different implementations. This provides an insight into the mechanisms, and their effectiveness.

4.3.1 Low-Context Scenario

Terminal Adaptation[11] is a key feature in mLearning applications such as [24]. However, in the APeLS architecture, it is a value-added service and not central to the learning process, which is primarily envisioned as happening on Desktop-style PCs. This makes the scenario and ideal example of a low-importance contextual requirement. The choice of one piece of content (or one candidate group) over another because of terminal capabilities is one that the Adaptive Engine is oblivious to.

4.3.1.1 Terminal Adaptation by Content Selection

This scenario employs the Content Group Manipulation mechanism to pass a list of candidate groups for each section of the course to the the Context Interpreter. In each

case, the Context Interpreter returned the most appropriate group, depending on the terminal properties of the Learner.

This scenario was implemented in the Mathematics Course, by means of the Content Group Update mechanism. This course has single-member candidate groups, sharing the names of their content.

In each case where there is a choice of candidates, the list is simply passed to the Context Interpreter, which selects one, based on the context of the User. This is then associated with a concept and compared with the Learner Model as normal (the **add-unit** function)

base.clp

```
1 (dom "add-to-parent" "section")
2 (dom "add-to-parent" "name" "Numbers, Functions and Graphs")
3 (add-unit "unit_nfg1"
4         (nth$ 1
5         (query-context "cgManipulation"
6         (create$ "nfg1.html" "pda_nfg1.html")
7 "http://localhost:8080/axis/InterpreterService.jws" "aeapp")))
```

The author of the narrative can therefore select which parts of the course can be selected for terminal adaptation, and which cannot. This is done by deciding whether or not to call the relevant function with a list of choices, or simply to add the unit.

4.3.1.2 Terminal Adaption by User Model Enrichment

This implementation of the Low-Context scenario makes use of the User Update mechanism to enrich the Learner model with Terminal information. An additional adaptivity type called *terminal.info* is added to the Learner model. The narrative then takes account of this attribute in selecting pagelets from the content group, the candidate groups having been selected based on (core) educational attributes of the learner.

The function for adding concepts to the Document tree of the SQL course is outlined below:

course model.clp

```
(deffunction add-subsection ( ?sectionname ?require ?list )
;; Declare varible to count page numbers in section
;; variable is set to zero initially
(bind ?pageCounter 0)
;; if learner contains a match to the contents of ?require
;;        then add section to DOM
;;        else do nothing
(if (eq (search-learner ?require "competencies.required") "TRUE" )
then
        (dom "add-to-parent" "subsection")
        (dom "add-to-parent" "name" ?sectionname)
        (dom "add-to-parent" "id" ?sectionname)
        ;gather the terminal type
        (if (eq (search-learner "desktop" "terminal.info") "TRUE")
        then
        ;desktop content is first in candidate list
        (bind ?contentindex 1)
        else
        ;pda content is second
        (bind ?contentindex 2)
)
(foreach ?var ?list
  ;get the content numbered by the index above
        (bind ?module
         (nth$ ?contentindex
          (get-content-list "subcollection3" ?var
          "//childrengroup/children")
         )
```

```
29          )
30          ;add the content
31          (dom "add-to-parent" "card")
32          (dom "add-to-parent" "cg" ?module)
33          (dom "set-parent" "..")
34  )
35  (dom "set-parent" "..")
36  else
37  (return 1)))
```

This function is adapted from the original SQL Course function to add Context-Awareness. In line 14, the system queries the user model for the terminal adaptivity of the User. This is contained in the *terminal.info* adaptivity category of the User, which is added by the Context Interpreter when the course is rebuilt.

This adaptivity category is added to the system in the JSP responsible for building the course:

test.jsp

```
1   learnerObject = Run.getLearner( learnerName );
2   // update the learnerObject with the context information,
3   // this is added to the competency matrix
4   // created above.
5   UserSOAPInterface soapIf =
6   new         UserSOAPInterface
7   ("http://localhost:8080/axis/InterpreterService.jws");
8   //call the CI
9   // "sql2.1" is used where the updated user model includes
10  // preferred programming language
11  // "sql3.1" is used where terminal adaptation is employed
12  learnerObject =
13  soapIf.updateUser(learnerObject, learnerObject.getIdentifier(),
```

```
14  "sql3.1", null);
15  // Store learner object as session attribute
16  session.setAttribute( "learner", learnerObject );
```

After constructing the Learner Model and storing it within the XML database, the JSP adds the additional, temporary contextual adaptivity to the Learner model. The loss of this temporary attribute in between builds is not a problem, as the candidate selection only occurs on rebuild, when the Context Interpreter will be called once more.

4.3.2 Mid-Context Scenario

This is an example of a 'broken' narrative; one where the flow of content requires a decision to continue. In this case, the choice of one language over another is the same (in terms of the Adaptive Models), and the Context Interpreter is simply required to make 'a' decision. This scenario also includes an implementation of the context feedback mechanism, in the form of the storage of the language choice, and presenting the user with the option to change it.

This scenario is implemented with the use of the Narrative Choice mechanism, which provides the Context Interpreter with two possible answers to choose from, 'db.applications.programming.java' or 'db.applications.programming.perl'. Each concept is associated with a set of Candidate Groups, and reuse is demonstrated by the fact that the cgprog1-000 candidate group can be used in both choices. Each of these choices fulfills the same educational role: to introduce database programming. However, from the user's perspective[4] the choice of languages is potentially highly important.

4.3.2.1 Language Selection by Narrative Choice

The Narrative Author defines a decision point by calling the Narrative choice user function with a list of options. The Context Interpreter receives this list and returns a list of its own, with one or more of the options contained in the question. A variable is

[4]but not that of the Adaptive Engine

used to store the resultant choice from the context-query. This is then used to select between Perl and Java.

course model.clp

```
;select perl or java
(bind ?testquestion
(query-context "narrativeChoice"
(create$ "db.applications.programming.java"
"db.applications.programming.perl")
"http://localhost:8080/axis/InterpreterService.jws" "sql"))
;print the response
(printout t ?testquestion)
;extract the first element
(if (eq (nth$ 1 ?testquestion) "db.applications.programming.java")
then
(add-subsection
"Programming with SQL in Java" "db.applications.programming"
(create$ cgprog1-000 cgjava1-000 cgjava1-001 cgjava1-002))
else
(if (eq (nth$ 1 ?testquestion) "db.applications.programming.perl")
        then
        (add-subsection "Programming with SQL in Perl"
        "db.applications.programming"
        (create$ cgprog1-000 cgperl1-000 cgperl1-001 cgperl1-002))))
```

4.3.2.2 Language Selection by User Model Enrichment

In the User Update model, the concept space is expanded to include both the **db.applications.programming.java** and **db.applications.programming.perl** concepts, each associated with their particular content groups. The adaptation is then performed as normal. However, the Learner

Model Concept Space has also been expanded, by enrichment through the User Update mechanism, to include one of the options.

4.3.2.3 Feedback on Content

In both experiments, the decision of the Context Server is recorded in a profile for the user in an XML database. This is then cached for the next use. In addition, a question is appended to the page display of the course, with the option to alter the decision made by the Context Interpreter. This decision is then recorded. The raw XML for a question is as follows:

Example Question (XML)

```
<question>
        <returnaddress>http://path.to.CI/answer.jsp</returnaddress>
        <query>You have Selected: db.applications.programming.java,
         options are:
         </query>
        <choices>
                <choice if="db.applications.programming.java">
                        Java
                </choice>
                <choice id="db.applications.programming.perl">
                        Perl
                </choice>
        </choices>
</question>
```

The **contextQuestions** method of the User Module is employed to gather the raw xml of the question, which it then translates based on a local stylesheet into xhtml, and appends to the page. A JSP implemented on the Context Interpreter is contacted via an xhtml form to update the status of the parameter in question. This update is performed via an XUpdate[18] to the Context Interpreter's database.

4.3.3 High-Context Scenario

One particular advantage of Context is the concept of the Learner's 'Learning Context', their history of learning through other adaptive courses, or through other means. This may mean that they, in fact, have a history of learning which should ideally be expressed in the Learner Model.

Traditionally, this has been acheived through the use of JSP-based tests, with questions and answers associated with tests. This is an extremely powerful method for determining the requirements and aptitude of the Learner, and it is not the intention of this scenario to replace that technique. Instead, it is intended that this scenario represent a situation where a test has *already* been performed, perhaps in another specific course, and that profile is translated by the Contxt Interpreter to the current Adaptive Engine.

4.3.3.1 Profile Discovery for Unknown User by User Model Update

The Adaptive Engine stores profiles of Learners who have used a particular course. In the event that a new Learner accesses the Maths course, the Context-Informed Adaptive Engine can query the Context Interpreter to discover if it has any information on the new Learner. This is performed by means of the User Update mechanism, which is used to pass the known information (perhaps as little as their username) to the Context Interpreter as a Learner Model. In the event that the Context Interpreter can determine information about the Learner, it returns a new Learner Model, with more descriptive attributes, perhaps discovered from other courses undertaken by the Learner. These new attributes are then employed to present a more relevant course to the Learner.

The portion of code responsible for locating the User profile is outlined below. In the event that **newLearner** is **null**, the Context Interpreter was not able to locate a profile.

login.jsp

```
1 String identifier = request.getParameter("learner");
```

```
Learner learner = null;
//if The Learner is *not* found, a NullPointerException occurs
try
{
        learner = Run.getLearner(identifier);
}
//if the learner object is not located, request one from the CI
catch(NullPointerException e)
{
        UserSOAPInterface soapIf =
        new UserSOAPInterface
        ("http://localhost:8080/axis/InterpreterService.jws");
        Learner newLearner = new Learner();
        //call the CI
        newLearner =
soapIf.updateUser(newLearner, identifier, "aeapp", null);
}
```

4.3.4 Additional Applications

There are a great many other ways of implementing these scenarios, as well as a broad range of other applications for context-information. For example it would have been possible to implement a more complex descriptor for content based on desired terminal. Rules-based narratives might have been designed, with the assertions directly or indirectly made via different methods.

As regards other domains, the selection of different media based on bandwith, for example text rather than video or animation on low bandwidth connections, or the selection of content based on licensing requirements could all be relatively easily modelled with these mechanisms.

4.4 Conclusions

This chapter has presented the Prototyping and Testing of Context-Informed Adaptive Hypermedia. While the responses from the Context Interpreter were pre-determined, the process of experimental testing provided significant insight into the many ways in which the different mechanisms can be employed.

In particular, it can be seen that the choice of mechanism reflects directly upon the burden of knowledge for the Adaptive Models —

- In the case of the **User Update** mechanism, the Adaptive Engine must model whatever adaptivities result from the contextual update for their effect to be seen. In particular, this can be seen in the requirement to represent the class of terminal within the narrative, in the terminal emulation experiment(*cf.4.3.1.1*).

- The **Narrative Choice** function permits the system to reason on the results of a choice[5] and supports complex decision processes. However, to be useful it also requires that the narrative be capable of handling whatever response returns usefully.

- **Candidate (Group) Manipulation** places no requirement on the system for models, it simply acts on the results of a decision, without necessarily being aware a decision has been made[6].

It is likely that any desired feature could probably be implemented with combinations of the others, in some form or another. The next chapter is concerned with analysing the effectiveness of this design and implementation *vis-a-vis* the guiding requirements outlined in the previous chapter.

[5]but not on the reasons *for* that choice

[6]In the sense that if the narrative passes all groups and does not descriminate the results, it has no burden to know if, or whether, there was a change

Chapter 5

Evaluation

The evaluation of this project will be based primarily on determining whether its guiding principles are reflected in both the design and implementation of the system. These principles, laid down in the initial phases of design, permit a quantitative evaluation of the performance of the resultant implementation.

5.1 Evaluation with Regard to Guiding Principles

Each of the principles outlined reflects either a property of Context Information, either in itself or in its applications. The determination of how these properties are realised within the system provides significant insight into the suitability of the architecture as designed, and additionally indicates future paths for development.

5.1.1 Autonomy

As implemented, there is no direct co-ordination between the systems as to the changes made in the state of the Adaptive Engine and the Context Interpreter. Each set of models can be viewed as being separated and autonomous. The Context Interpreter is free to alter its internal view at will, and synchronisation occurs whrn the Adaptive Engine runs the Narrative.

This lack of co-ordination does create one potential issue, which relates to the inability of the Context Interpreter, as written, to force the Adaptive Engine to rebuild its course. This can be resolved by arranging to poll the context server, in some fashion, perhaps via the ContextQuestions fed back to the system. In the event of a rebuild being recommended, the Context Interpreter can alert the Learner, or perform the task automatically.

It is likely that the preferred method for timing synchronisations will depend heavily on the application domain. Therefore, the system presented maximises autonomy, while permitting co-ordination to be implemented.

5.1.2 Simplicity

There is a considerable range of complexity open to Narrative authors using the contextual mechanisms. In principle, they can create a context-informed course simply by passing each candidate group list to the Context Interpreter, and adding the remainder. Otherwise a similar effect could be created by devising a fully expressive concept space where the Learner model is completed by the Context Interpreter.

Great care has been taken to provide seamless integration of narrative functions, particularly in Jess. The mechanisms act very much like the methods provided in the core Adaptive Engine, provided for searching the models supplied. This method of implementation also provides for a variety of styles without alteration to the mechanism.

There is a requirement for considerably more content to be provided in the case of a responsive, adaptive system. However, this requirement is not significantly greater than that imposed by providing similar support without context. In fact, contextual factors may reduce the requirement, by aiding in recognising opportunities for reuse through candidacy.

5.1.3 Expressiveness

There are two important factors for consideration in the principle of Expressiveness. The first is to evaluate the expressiveness of the functions provided. As discussed previously, there are many ways to use the functions and methods of the implementation to acheive the same goal. In addition, each mechanism can be used in many ways to accomplish many tasks. They are differentiated by the level of support required on the Adaptive Engine's part, and the model ownership of the representational logic[1].

The Second factor governs the expressiveness of the payloads themselves. It is evident from the course of the implementation that a number of extra possible categories of data are being expressed. The identifier of the user, and one for the narrative, provide the Context Interpreter with metadata to identify the respository of expressive content: the concept list.

The Addition of a sub-conceptual payload to the system significantly widens the scope of possible interactions between Context Interpreter and Adaptive Engine. This is an example of the use of and exchange vocabulary on the part of the Narrative also, and can also be represented by the use of aggregate or generalised concepts for the purposes of more complex reasoning.

5.1.4 Shared Knowledge

The degree of shared knowledge has been increased from the original design with the possibilities outlined such as alternate vocabularies and sub-conceptual listings. In the design of the Context Interpreter itself, it will be vital that these alternate possibilities are correctly recognised.

The matter of the identifier is also relevant. Identifiers are guaranteed unique within a particular instance of a course, however, there is a distinct likelihood of collision between courses and instances of courses. This is an example of the requirement for a full Context Interpreter to be able to accurately aggregate content from heterogeneous

[1] for example, representing the results of a contextual query as a variable within the Narrative, or as a category of adaptivity within the Learner Model.

sources, including different courses and different instances of the same course. Such a requirement may prompt the development of more complex transport metadata; rather than a simple identifier/narrativeName pair, a more descriptive complete characterisation of the context query may be required. In any case, the content or scheme of the descriptor is unimportant, from the perspective of the Transport mechanism unless the type and number of parameters change.

Early investigations suggest that Ontology-style schemes may provide a mechanism for describing the interchange between the internal Contextual Terms, and the language of the Adaptive Engine in question. Further, these ontologies may yield other advantages, such as the possibility of performing profile translation of the sort demonstrated in the experiments automatically or near-automatically. It is important to note that one key ontology within the Context Interpreter will be that of the course it interfaces with.

5.1.5 Obfuscation

Based on the architecture presented, there is little merit in creating an Adaptive Engine which directly accesses sensors and other contextual sources. The merit of the architecture presented is that the contextual concerns are largely prepared for direct use by the Adaptive Engine.

The obfuscation of raw contextual data means that Adaptive Engines cannot determine the reasons for a certain decision, merely that a decision has been made. Similarly, the reduction to concept lists of the question (with, possibly, descriptive metadata of some sort) prevents the Context Interpreter from over-stepping the boundary between the systems. This is a useful feature, as it provides a clear frontier between the systems, and reduces the chance for contention to invalidate the compound model.

Designing a scalable, transparent model for providing context is not a trivial undertaking. However, the architecture described does point to the fact that a relatively small number of key mechanisms have considerable potential to express the results of complex contextual reasoning.

5.1.6 Encapsulation

Encapsulation is synergistic with the Obfuscation principle. In addition to providing a boundary between the responsibilities and influences of the systems, the use of an agreed, limited vocabulary permits the Context Interface to gather its data from a vast array of different sources, and marshall them as conceptual decisions for the Adaptive Engine.

Encapsulation is the key to reducing the size of the Adaptive Engine's models, by providing identical interfaces to any Context Interpreter equipped with the appropriate vocabulary. Despite nearly any change to the features of the Contextual Environment of the Learner, the interface to the Context Interpreter remains unperturbed.

5.1.7 Frequency

This principle readily integrated into the separated architecture. The partition of the state of the system, so that a time-sensitive contextual view of the Learner and their environment is maintained by the Context Interpreter, which is then periodically updated as required by the Adaptive Engine.

This, combined with the translation into an agreed vocabulary, means that rapid, small changes in contextual axes will not create instability in the system, while methods have been outlined to permit the Context Interpreter to notify the user of an updated state.

5.1.8 Range

The injective mapping of broad-domain contextual values to discrete concepts greatly simplifies the potential size of Adaptive Models. The maximum range of the basis for exchange is that of the list variables presented by the Adaptive Engine for Contextual Decision.

Thus, a wide variety of axes, and wide variety within those axes can be represented easily as changing concepts, and only substantive changes are transmitted, where the

concepts in question change.

5.1.9 Importance

As demonstrated by the different categories of Experiment, the categories of importance for contextual axes are well-supported by the mechanisms. In fact, it is true to say that the mechanisms do not differentiate the importance of the influence for the purposes of transport.

The key principle underlying this architecture is that, ideally, the Adaptive Engine should not rely on the Context Interpreter to produce answers for central decisions in the Narrative. Influcences of that sort are better modelled directly, and experience with other systems shows that this form of deep modelling is highly effective. However, it does come with a penalty of size and complexity

5.1.10 User Empowerment

Empowering the Learner avoids the significant risk inherent with all automatically reconfiguring systems, namely that the Learner loses their sense of control, becomes jaded and loses interest. This is particularly important where deep learning tasks are being undertaken. The current implemenation provides for user control and feedback, it is thought that a full implementation of the Context Interpreter will be capable of providing options for many of the decisions made, and descriptions for the others.

There are a number of uses for this apart from helping the Learner feel included in the process. Of prime use amongst these is encouraging a dialogue between the Learner and the Context Interpreter, to permit the Interpreter to ask for answers to questions it cannot deduce itself, and to allow for preferences to be set.

Many of these features are readily implementable with the mechanisms provided, particularly because control of the display and translation of the queries from the Context Interpreter lies with the Adaptive Engine. This supports refreshing query windows and layout sensitive to the course content presented.

5.1.11 Performance

The use of a web service format introduces the question of performance under scale in a fully featured system. An evaluation of these factors will require the development of a feature-rich Context Interpreter. However, initial examinations of current schemes show that the payload size of each mechanism is relatively small, and that speed penalties are negligible with regard to the time to rebuild a course. The Narrative author is in full control of the size and frequrncy of these messages, they can determine what mechanisms are called with what data. The only complete model sent at once is that of the Learner, and this has not been found to be of size sufficient to overload the system.

This is primarily due to the use of core SOAP data types such as, for example, multi-dimensional arrays to represent adaptivity arrays, greatly lowers the overhead for transmitting information, and the fact that the data exchanged is in the form of Strings, which are easily translated between SOAP, Jess and Java.

5.2 Analysis

By conforming to the principles laid down, the separated architecture described provides a highly customisable compound system, automatically responsive to a wide variety of influences.

5.2.1 Access to Models

The access granted by the mechanisms provided is indirect. In fact, the mechanisms provide no guarantee that Contextual decisions will be accounted for by the system. This permits the Adaptive Engine to maintain control over the 'shared model', which becomes a conceptual, rather than actual entity. The 'Double Blind' architecture created through the guiding principles is intended to enhance the modularisation of the system, creating a component-like relationship rather than an overly integrated one. Thus, the architecture presented, that of a separate high level interpreter, can be

viewed not just in terms of the example Adaptive System, but as a general architecture for applying context to applications.

5.2.2 Three Questions Revisited

Having classified a number of Context-Aware systems during the Survey(*cf. Chapter 2*), it is useful to apply the same '3 Questions' to the combined Context-Informed Adaptive Hypermedia System presented.

5.2.2.1 What is Context?

Defined during the survey, Context in the case of this system can be viewed as the set of useful influences relevant to the adaptation, which could not be specified in advance, or which are too complex to model deeply within the Adaptive Engine's core models.

The values of these influences are translated into an agreed vocabulary by the Context Interpreter, which then transmits them to the Adaptive Engine.

5.2.2.2 Where can Context be Used?

The principles of Obfuscation and Encapsulation govern the use of Context in the system presented. In effect, context can be used 'anywhere' and 'nowhere', Contextual hooks are defined by the Adaptive Engine, which are filled by the Context Interpreter purely on the basis of the Contextual Axes at the time.

Context influences the entire system, any element involved in adaptation can potentially be affected, if the Adaptive Engine permits it. This influence is specified by the Adaptive Engine, which can control its granularity through the use of different methods, and by defining different option lists for the Context Interpreter.

In summary, Context can be used anywhere where the Adaptive Engine allows it and the Context Interpreter recognises a translatable alteration to the state of the models.

5.2.2.3 Where can Context be Gathered?

The substantial feature of gathering context in this architecture is that it is done from an extremely wide variety of sources, at different levels and for different purposes. There are a number of levels at which the Context Interpreter may gather data. There is the commonly-used sensor data view, where raw data is gathered and mapped to concepts or choices within concepts.

Of perhaps greater value is the higher level translation function which the Context Interpreter serves. The gathering of Context from other applications and domains, such as for example other Adaptive Engines, yields real potential for tailored content to be provided to the Learner.

5.3 Conclusions

This chapter has analysed the Design and Implementation of Context-Informed Adaptive Hypermedia in terms of the founding principles of the project. In doing so, a number of features of the system are highlighted, including the nature of the shared vocabulary and 'shared model' as well as the pattern of usage for the system.

In addition to these valuable insights, a number of requirements for a fully-featured Context Interpreter have been revealed. These include the ability to translate between vocabularies and the maintenance of the state of Contextual axes, a state which is not visible to the Adaptive Engine in between synchronisations.

Overall, the principles delineated have been followed and this has resulted in a set of mechanisms which will encourage the design of an effective Context Interpreter under the same principles.

Chapter 6

Conclusions

In summarising the project as a whole, this chapter presents the body of work and the numerous insights gained through the research process. Initially, the project is evaluated with regard to the Objectives outlined in the Introduction (*cf. 1.2*). This evaluation is followed by some possible avenues for future investigation, including the areas of future development towards creating a fully operational system. Finally, a number of closing remarks summarise the project's achievements.

6.1 Evaluation of Objectives

The first step in examining Context-Informed Adaptive Hypermedia has been completed, by forming an initial view of what Context is in terms of this domain, and how it might be used to enrich the eLearning experience and empower learners with a broader, more adaptive and responsive system.

There appears to be considerable promise in the possibility of adding Context Interpreters to Adaptive Hypermedia systems, as the technology infrastructure for Pervasive Computing advances, the seamless integration of the information available from Context with Core adaptive models will permit system designers the freedom to create a wide variety of content without worrying about the sources of Contextual data, or an ever-growing list of marginally useful inputs.

6.1.1 Survey and Definition of Context

The survey undertaken reviewed a number of Context-Aware and Adaptive Hypermedia systems. In examining the area of Context-Awareness, a characteristic simplicity was recognised in most systems described. Many of the systems presented were relatively simple in their logic, and did not contain complex internal logic. They were, despite their obvious qualities, too simple to reveal the true power of Context. The process of examining Context-aware systems, and a review of the literature concerned with Context, revealed that there were many definitions. This document presented its own definition of context, which has numerous advantages:

1. It provides a method for delineating between 'system' and 'context' concerns. This is particularly important in applications such as Adaptive Hypermedia, which contain their own representational logic. This delineation was characteristic, rather than specific, and demonstrated that Context is a mutable entity based on the specific concerns of an application[1].

2. The Definition provided yields an indication of the point at which Interpretation should end with Context. Instead of simply being collected sensor data, Context is expressed in terms of the system itself, necessitating the *high-level* Context Interpreter described.

The second portion of the Survey included an analysis of Adaptive Hypermedia systems from a number of sources, and identified their models and methods for content creation and sequencing. This, combined with the Context-aware survey, provided the evidence of a clear opportunity to bridge the divide between the two areas, supplying Adaptive Hypermedia with a simpler way of taking unknown concerns into account, while giving Context-awareness a forum for development and experimentation with semantically complex systems.

[1]For example, the case of Location which is a relatively peripheral concern in eLearning, but is a central concern in mLearning

6.1.2 Designing Mechanisms for Context-Informed Adaptive Hypermedia

The design presented is based around a separated architecture, in which a high-level Context Interpreter gathered heterogeneous contextual inputs and translated them transparently into decisions relevant to the Adaptive Engine. The architecture is based around ten guiding principles, including encapsulation and simplicity. The overall aim of the design was to provide access to the wide variety of Contextual axes to an Adaptive Engine in a transparent fashion. The design presented is highly modularised, and demonstrates a number of mechanisms for providing the Context Interpreter access to the state of the Adaptive Engine and its models, depending on the preference of the Course Designer. The 'double-blind' quality of the design, intended to encourage transparency and pluggability requires that the Context Interpreter be capable of translating a variety of information into operations on a list of entities meaningful to the Adaptive Engine, the shared knowledge of the compound system. This format for exchange enables the architecture to support a wide range of contextual axes of varying importance, with potentially rapidly changing and highly variable value ranges. The Context Interpreter effectively guarantees that only substantive changes to the Context of the Learner will be propagated to the Adaptive Engine, since less important variations will have no impact in the translated concept space. The specific mechanisms defined to support these exchanges allow different levels of expressiveness, requiring different model support within the Adaptive Engine. These are supplemented with a mechanism for empowering the Learner to interact with and be notified of the decisions taken on their behalf. These mechanisms combine to describe the interaction between the Context Interpreter and the Adaptive Engine and the Learner and the Context Intepreter. The mechanisms provided are open-ended and relatively un-restricted, creating few pre-conditions on the payload of their exchanges.

6.1.3 Prototyping and Testing

Prototyping and testing was performed on the APeLS Adaptive Hypermedia System, two courses, provided with the system were used as a basis for modification to include Context. Content was added in both cases, and narratives and JSP code was altered to accommodate a variety of tests. The Contextual Mechanisms were defined as Jess functions and as an interface for JSPs, and the resultant parameters were translated over SOAP to the Context Interpreter, implemented as a Web Service. A database back-end was included in the Context Interpreter, to perform Learner Feedback, but the system overall was developed only to provide specific test-case responses to the Adaptive Engine. This was a useful method to develop this early design, as the focus of investigation lay with the Mechanisms themselves. Three Scenarios for testing were examined. These were divided based on the level of involvement of context in the adaptation process. The Low-level Scenario was concerned with providing Terminal Emulation information to the Adaptive Engine. This scenario was implemented in a number of ways, using different mechanisms to express the need either implicitly or explicitly. The Mid-Context scenarios were based on the concept of a broken narrative, with contextual decision points. In this case, multiple distinct paths were open to the Adaptive Engine each of which was identical from the perspective of the core concept space. The Context Interpreter was able to provide a broader conceptual view and determine the most appropriate path. This scenario also included an investigation into User Empowerment, by permitting the Learner to verify and alter the choice made by the Context Interpreter. In the High-Context Scenario, the Context Interpreter provided the complete state of one of the models in the system. The Context Interpreter held a record of a profile for a Learner previously unknown to the Adaptive Engine. While it would be quite possible to determine the profile of the Learner through testing, it is more convenient for the Learner to be automatically recognised. Furthermore, it is possible that Context Interpreters will be capable of aggregating the experience of Learners over a variety of courses and systems, so that each new course they undertake

accounts for this experience. The examples implemented to demonstrate and test the capabilities of the Mechanisms defined reveal only the first portion of a large number of possibilities. Variations in structure and interaction between mechanisms, as well as increasingly complex narratives will reveal the true potential of the mechanisms. However, the examples presented do grant significant insight into their effectiveness, and this combined with the evaluation performed in this document offer valuable insights into the design of the Context Interpreter itself.

6.2 Future Work

The work presented represents the initial investigation of an area with considerable potential. The choice of the mechanisms of interaction for investigation was motivated by the considerable body of work in Adaptive Hypermedia and Context-Awareness, the synergy between the two areas has yielded a useful structure for the design of the Context Interpreter.

6.2.1 Design and Implementation of the Context Interpreter

There are a number of tasks which must be performed by the Context Interpreter. The most important of these is to translate whatever contextual inputs it may have (including User input) into substantive changes expressable through the mechanisms provided. The High-level of operation of the Context Interpreter permits enormous internal variation and an almost unlimited scope for input, the range of which is entirely invisible to the Adaptive Engine. However, these changes are intended to be highly visible to the User, who will have the opportunity to interact with such decisions.

6.2.1.1 Gathering Context

Context can be gathered at all levels by an Interpreter fitting the architecture described. One potential source of context with considerable merit is in the translation of concepts and profiles from one application to another. At its simplest, and perhaps

most powerful, this is expressed in the High-Context Experiment demonstrated. However, other associations might be made between differing application domains, where adaptive learning profiles could help in tasks such as service composition or information searching. Sensor-oriented contextual information must be translated into changes in concept choices. It is likely that an intermediate contextual model will be created within the Context Interpreter, which will then be sub-selected and translated to use in the Adaptive Engine.

6.2.1.2 From Context to Concept

The automatic correlation of inputs with the Adaptive Concept Space is a vital component of the eventual complete system, there is considerable work in this area, through the use of Ontologies or Topic Maps, vocabularies can be translated and transliterated as required, over multiple steps if necessary. This process is, however, not trivial, there are numerous considerations including the issue of partial or conflicting translations, incomplete expressions and the question of aligning elements so that they do indeed agree. One possible solution would entail the generation of a profile over time, gathering the usage history of the User, and employing it to support decisions. It is likely that there will be a need for case-specific structures to be associated with the Context Interpreter in different domains and instances; it is hoped that the level of automation can be increased, however.

6.2.2 Generalisation

There is no conceptual bar to this architecture being applied to other systems and application domains. There are numerous application domains which would benefit from the ability to transparently include Contextual axes. In each case, the list of axes which belong to Context may vary, but they can all be recognised as supplementary enrichments to the core semantic view of the system. As software move in the direction of service oriented architectures, the ability to refer to an encapsulated source for

additional information will improve User experience in most applications with relatively little overhead at the application level. This is due to the fact that the extra information is supplied in a useful vocabulary automatically. The prevalence of opportunities to aggregate Context over numerous applications will likely depend on the facility for translating meaningfully between them, and relating the implications of one decision to another.

6.3 Final Remarks

The Mechanisms presented in this document are reflective of a potentially extremely powerful tool in future design of software. The opportunity to link applications via Context, with the ultimate goal of all applications using all informationa available to them is highly compatible with the vision of ubiquitous computing systems and service oriented applications. While this work represents only the earliest stages of development, it has demonstrated some of the promise of a later, more powerful system which draws its capabilities by bridging and incorporating valuable results from multiple research areas.

Bibliography

[1] Bardram, J. E., Kjær, R. E., and Pedersen, M. Ø. Context-aware user authentication - supporting proximity-based login in pervasive computing. In *UbiComp 2003, the Fifth International Conference on Ubiquitous Computing* (Seattle, Washington, USA, 2003).

[2] Brusilovsky, P. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction 6*, 2-3 (1996), 87–129.

[3] Brusilovsky, P., Eklund, J., and Schwarz, E. Web-based education for all: a tool for development adaptive courseware. *Computer Networks and ISDN Systems 30*, 1–7 (1998), 291–300.

[4] Brusilovsky, P., and Nijhavan, H. A framework for adaptive e-learning based on distributed re-usable learning activities. In *Proceedings of World Conference on E-Learning, E-Learn 2002* (Montreal, Canada, 2002), AACE, pp. 154–161.

[5] Chen, G., and Kotz, D. A Survey of Context-Aware Mobile Computing Research. Tech. Rep. TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000.

[6] Clarke, D. Learning domains or bloom's taxonomy. `http://www.nwlink.com/~donclark/hrd/bloom.html`, 1995.

[7] Clarke, L. Adult learning through adaptive systems. Master's thesis, University of Dublin, Trinity College, Dept. of Computer Science, 2003.

[8] Conlan, O., Wade, V., Bruen, C., and Gargan, M. Multi-model, metadata driven approach to adaptive hypermedia services for personalized elearning. In *Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems* (Malaga, Spain, 2002).

[9] Consortium, A. Project aura web page. `http://www-2.cs.cmu.edu/~aura/`.

[10] Dagger, D., Conlan, O., and Wade, V. An architecture for candidacy in adaptive elearning systems to facilitate the reuse of learning resources. In *World Conference on E-Learning in Corporate, Government, Healthcare and Higher Education (eLearn 2003)* (2003), pp. 112–116.

[11] Dagger, D., Conlan, O., and Wade, V. Towards anytime, anywhere learning: The role and realization of dynamic terminal personalization in adaptive elearning. In *Ed-Media 2003, World Conference on Educational Multimedia, Hypermedia and Telecommunications* (Hawaii, USA, 2003).

[12] De Bra P., S. T., and P., B. Aha! meets interbook and more... In *AACE ELearn 2003 Conference* (Phoenix, Arizona, 2003).

[13] DeBra, P., Aerts, A., Berden, B., deLange, B., Rousseau, B., Santic, T., Smits, D., and Stash, N. Aha! the adaptive hypermedia architecture. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia* (2003), ACM Press, pp. 81–84.

[14] Dey, A., Abowd, G., and Salber, D. A context-based infrastructure for smart environments, 1999.

[15] Don Box, DevelopMentor and David Ehnebuske, IBM and Gopal Kakivaya, Microsoft and Andrew Layman, Microsoft and Noah Mendelsohn, Lotus Development Corp. and Henrik Frystyk Nielsen, Microsoft and Satish Thatte, Microsoft and Dave Winer, UserLand

Software, Inc. . Simple Object Access Protocol (SOAP) 1.1. Note 08, W3C, May 2000.

[16] Foundation, A. Tomcat home page. http://jakarta.apache.org/tomcat/.

[17] Foundation, A. S. Axis home page. http://ws.apache.org/axis/.

[18] Foundation, A. S. Xindice developer guide. http://xml.apache.org/xindice/guide-developer.html.

[19] Foundation, A. S. Xindice home page. http://xml.apache.org/xindice/.

[20] Friedman-Hill, E. Jess home page. http://herzberg.ca.sandia.gov/jess/.

[21] Griswold, W. G., Boyer, R., Brown, S. W., and Truong, T. M. A component architecture for an extensible, highly integrated context-aware computing infrastructure.

[22] Griswold, W. G., Boyer, R., Brown, S. W., Truong, T. M., Bhasker, E., Jay, G. R., and Shapiro, R. B. ActiveCampus - Sustaining Educational Communities through Mobile Technology. UCSD CSE technical report CS2002-0714, Department of Computer Science and Engineering University of California, San Diego, July 2002.

[23] Knowles, M. *The Modern Practice of Adult Education: From Pedagogy to Andragogy.* Cambridge/Prentice Hall Regents, New Jersey, 1980.

[24] Lonsdale, P., Baber, C., and Sharples, M. A context awareness architecture for facilitating mobile learning. In *MLEARN 2003: Learning with Mobile Devices* (London, UK, 2003).

[25] Schilit, B., Adams, N., and Want, R. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications* (Santa Cruz, CA, US, 1994).

[26] Schmidt, A., and Beigl, M. There is more to context than location: Environment sensing technologies for adaptive mobile user interfaces, 1998.

Appendix A

UML Class Diagrams for the Implementation

This Appendix contains the UML Class diagrams employed to describe the implementation of the Contextual Mechanisms described in the Project. The first Diagram is of the Interpreter structure, the Interpreter interface included a number of implementations, concerned with different experiments, all are members of package **ie.tcd.cs.kdeg.ae.contextInterpreter**:

- LanguageSelector: provided the responses for Narrative Choice based language selection.
- AnonymousUserLocator: provided a profile, if one was available for an unknown user.
- CandidateGroupSelector: manipulated content group lists or content lists.
- SQLQuestion: provided the raw query XML for user feedback.
- UserLanguageUpdate: processed the Learner Model to add a language preference.

The second diagram (A.2) aggregates the designs of the different modules, and the alterations to the LearnerObject class.

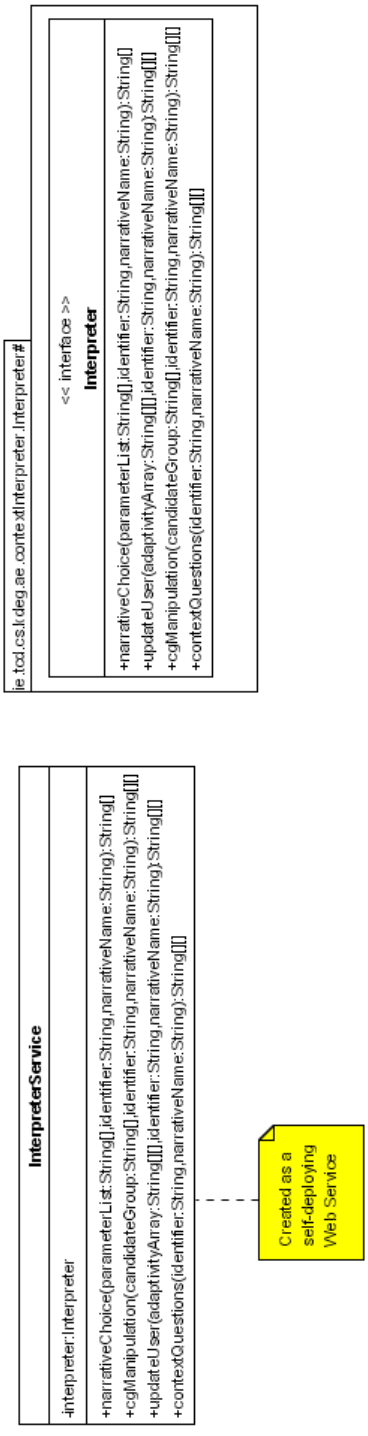

Fig. A.1: Interpreter Class Diagram

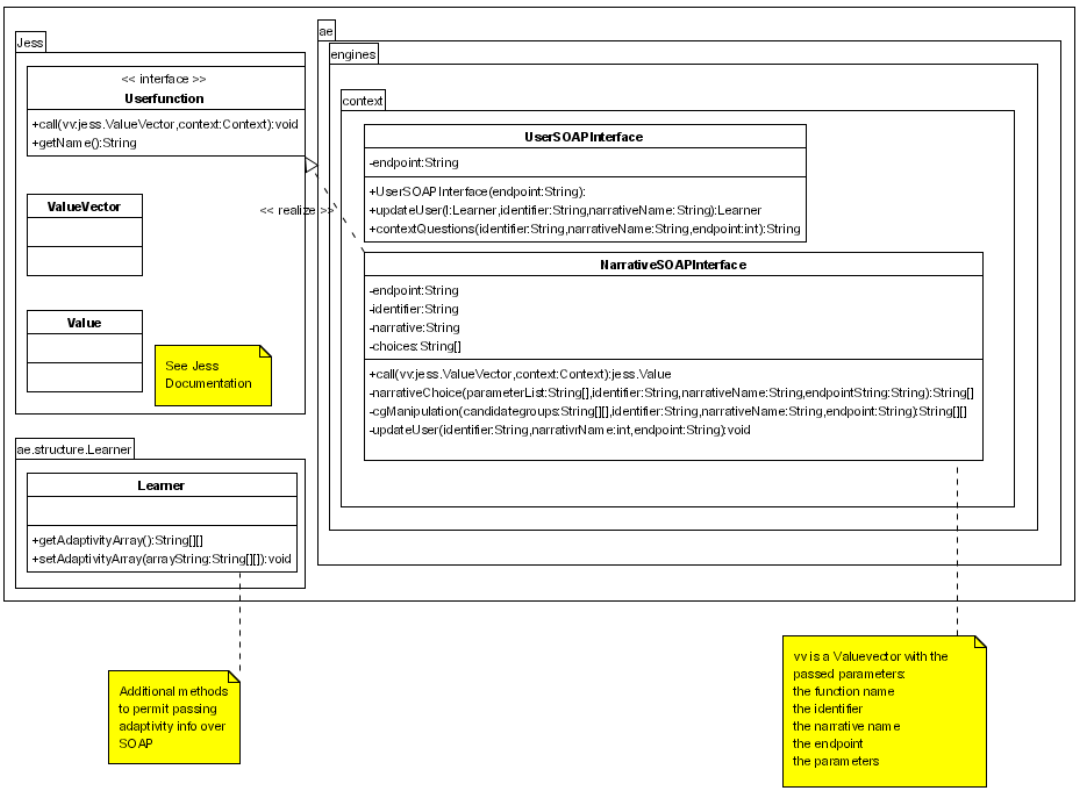

Fig. A.2: Interpreter Class Diagram