

A Thesis Submitted for the Degree of PhD at the University of Warwick

Permanent WRAP URL:

<http://wrap.warwick.ac.uk/111668>

Copyright and reuse:

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it.

Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk



A Visual Adaptive Authoring Framework for Adaptive Hypermedia

By

Javed Arif Khan

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

Doctor of Philosophy in Computer Science

Supervisor: Dr. A. I. Cristea

University of Warwick, Department of Computer Science

August 2018

Contents

| | | |
|------|--|----|
| 1 | Introduction..... | 16 |
| 1.1 | Background and Motivation | 16 |
| 1.2 | Aims | 18 |
| 1.3 | Problem Description and Challenges | 18 |
| 1.4 | Research Questions | 19 |
| 1.5 | Objectives | 20 |
| 1.6 | Research Contributions | 22 |
| 1.7 | Thesis Outline | 23 |
| 2 | Background and Related Research..... | 25 |
| 2.1 | Introduction..... | 25 |
| 2.2 | What is an Adaptive Educational Hypermedia System?..... | 26 |
| 2.3 | Authoring of Adaptive Educational Hypermedia Systems | 27 |
| 2.4 | Frameworks for Adaptive Hypermedia..... | 28 |
| 2.5 | Authoring Systems for Adaptive Hypermedia..... | 36 |
| 2.6 | Generic Problems with the Authoring in Adaptive Hypermedia Systems | 55 |
| 2.7 | Authoring Imperatives..... | 58 |
| 2.8 | Interoperability between Adaptive Educational Hypermedia Systems..... | 59 |
| 2.9 | Visual Programming Systems..... | 60 |
| 2.10 | Conclusion..... | 63 |
| 3 | Methodology..... | 64 |
| 3.1 | Literature Review and Background Review..... | 64 |
| 3.2 | User Centred Design and Implementation..... | 65 |
| 3.3 | Implementation Methodology: The Visual Platform | 67 |

| | | |
|-----|---|-----|
| 3.4 | Experimental Methodology | 70 |
| 3.5 | Summary and discussion | 78 |
| 4 | Theoretical Framework: Author Roles & Functionality Matrix for Adaptive Authoring Tools.. | 79 |
| 4.1 | Introduction..... | 79 |
| 4.2 | Author Role Functionality Matrix..... | 80 |
| 4.3 | Conclusion | 87 |
| 5 | Creating a Visual language for LAG and constructing VASE..... | 89 |
| 5.1 | Introduction..... | 89 |
| 5.2 | LAGBlocks: the visual language for adaptive hypermedia..... | 89 |
| 5.3 | Enhancing the LAG programming language..... | 97 |
| 5.4 | VASE 1.0: the initial approach towards a visual adaptive authoring tool | 97 |
| 5.5 | VASE 2.0: The Extended Adaptive Authoring System | 104 |
| 5.6 | Conclusions..... | 120 |
| 6 | Evaluation of VASE: the Visual Adaptive Strategy Environment | 121 |
| 6.1 | Introduction..... | 121 |
| 6.2 | Hypotheses..... | 122 |
| 6.3 | Non-professional Programmers' Case Study..... | 123 |
| 6.4 | LAG Expert Group Case Study | 124 |
| 6.5 | VASE 1.0 evaluation with University lecturers with eLearning experience..... | 127 |
| 6.6 | The Large Scale Final Experiment: Evaluation of VASE 2.0 | 132 |
| 6.7 | Discussions and Hypotheses Validation..... | 157 |
| 6.8 | Revisiting the Comparison of Authoring Systems for Adaptive Hypermedia | 163 |
| 6.9 | Conclusions..... | 164 |
| 7 | Conclusions and Recommendations for Future Work | 169 |
| 7.1 | Answers to Research Questions..... | 169 |

| | | |
|-----|---|-----|
| 7.2 | Addressing the Research Objectives | 173 |
| 7.3 | Original Contributions..... | 176 |
| 7.4 | Recommendations for Future Work..... | 177 |
| 8 | References..... | 179 |
| | APPENDIX A: LAG Grammar | 188 |
| | APPENDIX B: LAG Grammar Semantics..... | 189 |
| | APPENDIX C: LAG Extension..... | 192 |
| | APPENDIX D: Novice Group Evaluation | 193 |
| | APPENDIX E: Non-professional programmer evaluation results..... | 194 |
| | APPENDIX F: The CAF Language Format..... | 199 |
| | APPENDIX G: ADE 0.2 and LAG 4.0 Demonstration Courses..... | 200 |
| | APPENDIX H: LAG Strategies..... | 205 |

List of Figures

| | |
|---|----|
| Figure 1: Adaptive Hypermedia Application Model (AHAM) | 28 |
| Figure 2: LAOS framework (5 Layers) [20]..... | 29 |
| Figure 3: An example of course material shown in InterBook in the Netscape Browser [28]..... | 37 |
| Figure 4: AHA Adaptation Rules Form | 39 |
| Figure 5: Subject Matter Concept Space Editor (SMCS) in ACCT..... | 41 |
| Figure 6: Narrative Model Builder in ACCT..... | 42 |
| Figure 7: APeLS architecture..... | 43 |
| Figure 8: ACTSim Composition Tool..... | 45 |
| Figure 9: Authoring Pages Separately (Left) or via <i>Template Pages</i> (Right) | 47 |
| Figure 10: Consistency in presentation through Template Pages in GRAPPLE authoring | 48 |
| Figure 11: Creating the ‘Milky Way’ domain model with the GRAPPLE Domain Model tool..... | 49 |
| Figure 12: Adding two PRT relations in the CAM Model of the ‘Milky Way’ | 51 |
| Figure 13: CAM created in the CAM Model, by dragging PRTs from the left menu | 51 |
| Figure 14: PEAL Adaptive Strategy Editor..... | 52 |
| Figure 15: PEAL Graphical Editor..... | 53 |
| Figure 16: CodeBlocks - IF Block | 61 |
| Figure 17: Blockly Block | 62 |
| Figure 18: Phases of user centred process (source www.usability.gov [46])..... | 66 |
| Figure 19: Functionality Matrix for PEAL..... | 81 |
| Figure 20: LAGBlock to control the User Access to Goal Model instances..... | 91 |
| Figure 21: LAGBlock - To specify Goal Model Concept label..... | 92 |
| Figure 22: LAGBlock - To set the User Knowledge level for a Goal Model Instance..... | 92 |
| Figure 23: LAGBlock - If Construct | 93 |
| Figure 24: LAGBlock - to represent the LAG User Model..... | 94 |

| | |
|---|-----|
| Figure 25: LAGBlock - PM construct – to represent the LAG Presentation Model..... | 95 |
| Figure 26: LAGBlock - GM construct - to represent the LAG Goal Model..... | 95 |
| Figure 27: LAGBlock code (left) versus LAG code (right)..... | 96 |
| Figure 28: ‘For Each’ Adaptive Strategy in LAGBlocks..... | 97 |
| Figure 29: VASE Interaction with external tools..... | 98 |
| Figure 30: Screenshot of VASE 1.0..... | 99 |
| Figure 31: Dimming Strategy in LAGBlocks in VASE 1.0..... | 100 |
| Figure 32: VASE 1.0: the start..... | 101 |
| Figure 33: VASE 1.0 Interface - Initialization (on left pane) Implementation (on right pane)..... | 101 |
| Figure 34: VASE 1.0 Interface – red boxes to represent various features | 102 |
| Figure 35: VASE 1.0: To show how compactness works | 103 |
| Figure 36: VASE 2.0 System Architecture..... | 105 |
| Figure 37: VASE 2.0 Architecture Diagram | 106 |
| Figure 38: VASE 2.0 Logic Blocks | 108 |
| Figure 39: VASE 2.0 Logic Blocks complex usage..... | 109 |
| Figure 40: VASE 2.0 User Model Blocks | 110 |
| Figure 41: VASE 2.0 Goal Model Blocks..... | 111 |
| Figure 42: VASE 2.0 Presentation Model Blocks 1 of 2 | 112 |
| Figure 43: VASE 2.0 Presentation Blocks 2 of 2 | 113 |
| Figure 44: VASE 2.0 Dynamic LAGBlocks | 114 |
| Figure 45: VASE 2.0 Dynamic LAGBlock addition | 115 |
| Figure 46: VASE 2.0 Additional LAGBlock options | 116 |
| Figure 47: VASE 2.0 Collapse block option | 116 |
| Figure 48: VASE 2.0 Expand Block option | 117 |
| Figure 49: Beginner Intermediate and Advance strategy in VASE 2.0 | 118 |
| Figure 50: User interface screenshot of the working system (VASE 2.0) | 119 |

Figure 51: Amazon workers' requirement to do HIT 158

List of Tables

| | |
|---|-----|
| Table 1: Research Problem mapping to Research Questions and Objectives | 23 |
| Table 2: Comparison of well-known adaptive hypermedia authoring systems..... | 56 |
| Table 3: Raw Data Collected from Users and their interpretations | 128 |
| Table 4: Authoring System Requirements..... | 131 |
| Table 5: Questionnaire mapping onto the research questions | 134 |
| Table 6: Normality of Answers to Questionnaire for VASE 2.0 and PEAL..... | 137 |
| Table 7: VASE 2.0 Results for questions related to <i>user role</i> for H1 | 140 |
| Table 8: PEAL Results for questions related to <i>user role</i> for H1 | 142 |
| Table 9: PEAL versus VASE overall results for the questions related to <i>user roles (H1)</i> | 143 |
| Table 10: VASE 2.0 Results for <i>Benefits</i> questions for H2..... | 144 |
| Table 11: PEAL Results for <i>Benefits</i> questions for H2..... | 147 |
| Table 12: PEAL versus VASE Results for the questions related to <i>benefits (H2)</i> | 149 |
| Table 13: VASE 2.0 Results for <i>usability</i> questions for H3..... | 150 |
| Table 14: PEAL Results for <i>usability</i> questions for H3..... | 152 |
| Table 15: PEAL versus VASE Results for the questions related to <i>usability (H3)</i> | 153 |
| Table 16: VASE 2.0 Results for <i>Quality</i> questions for H4 | 154 |
| Table 17: PEAL Results for <i>Quality</i> questions for H4..... | 155 |
| Table 18: PEAL versus VASE Results for questions related to <i>Quality (H4)</i> | 156 |
| Table 19: Summary of PEAL versus VASE Results for question related to each category | 156 |
| Table 20: Comparison of adaptive hypermedia authoring systems and VASE 1.0 and VASE 2.0..... | 162 |

Acknowledgements

My research interest in hypermedia originates from a project on hypermedia structures.

In 2010, I visited the Computer Science department at the University of Warwick. There I met my supervisor, Dr. Alexandra Cristea and found our common interest on adaptive hypermedia. She introduced me to the field of adaptive hypermedia by demonstrating the MOT (My Online Teacher) and ADE (Adaptive Delivery Engine) Systems.

They had very basic features; over the course of the following year, MOT was developed iteratively by Jonny Foss. MOT was further developed and enhanced during the GRAPPLE FP7 project.

The MOT system's storage architecture was changed from a tree to a graph based structure, which meant that it could be more flexible in storing adaptations. MOT evaluation revealed that PEAL, the authoring system used by MOT, was not adequate to create adaptive specification, especially for non-computer-savvy authors. This meant that such authors without technical knowledge seriously struggled to use and to create adaptive specifications from scratch.

The evaluation listed followed major issues with the PEAL authoring system. After reviewing the authoring system and the LAG language, it was decided to look into how a new authoring system can be designed to lower the knowledge barrier for authors, so that the author doesn't have to learn a new language for adaptation, just to create or to re-use existing adaptive specification. Devoting myself to this research for over five years, I've learned a great deal in this research area and gained international research experience. A huge gratitude is owed to my supervisor Dr. Alexandra I. Cristea. She has guided and helped me with her broad knowledge and great kindness.

I would like to thank all the committee members, especially Dr. Mike Joy and Dr. Jane Sinclair. They gave me a different view on my research, making it more rigorous and balanced. Within my group, many colleagues kindly helped me in various ways. I would specifically thank Dr. Jonny Foss and Dr. Joshua Scotton.

Declaration

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. I hereby declare that, except where acknowledged, the work presented in this thesis has been composed by myself and has not been submitted elsewhere for the purpose of obtaining an academic degree.

Javed Arif Khan

Signature: _____

Date: _____

Publications

Some of the material in this thesis is based on a paper that I have previously published (as first author, with a contribution of over 80%) as described below.

The Introduction presents my own research questions, whilst Chapter 2 describes related research. Chapter 3 describes the research methodology on data collection, analysis and evaluation.

Chapter 4's description of the functionality matrix and user-roles in adaptive hypermedia is loosely based on the following paper:

Khan, J. A, Cristea, A.I, & Stewart, C. (2011) "Adaptive Authoring of Adaptive Hypermedia towards, Role-based, Adaptive Authoring", CATE 2011, pp. 734-042. Publication date July 2011.

Chapter 5 presents a visual programming framework (VASE) for the LAG language and it is based on the ideas that were published in the following papers:

Khan, J. A and Cristea, A.I. (2015) "Adaptive Hypermedia: A simple way to create Adaptive Strategies using LAGBlocks". In S. Carliner, C. Fulford & N. Ostashewski (Eds.), Proceedings of EdMedia: World Conference on Educational Media and Technology 2015 (pp. 195-202). Association for the Advancement of Computing in Education (AACE).

Khan, J. A. (2007). Node Classification in Hypermedia Systems. In C. Montgomerie & J. Seale (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2007 (pp. 1464-1469). Chesapeake, VA: AACE.

Chapter 6 presents various evaluations of VASE 1.0 and VASE 2.0, and is reflected in the following paper:

Khan, J. A. and Cristea, A.I. (2018) "A large-scale evaluation of a visual language for adaptive hypermedia", The 3rd International Conference on Information and Education Innovations (ICIEI'18), London, June 30-July 2, 2018 (has been submitted & accepted).

Abstract

In a linear hypermedia system, all users are offered a standard series of hyperlinks. Adaptive Hypermedia (AH) tailors what the user sees to the user's goals, abilities, interests, knowledge and preferences. Adaptive Hypermedia is said to be the answer to the 'lost in hyperspace' phenomenon, where the user has too many hyperlinks to choose from, and has little knowledge to select the most appropriate hyperlink. AH offers a selection of links and content that is most appropriate to the current user. In an Adaptive Educational Hypermedia (AEH) course, a student's learning experiences can be personalised using a *User Model* (UM), which could include information such as the student's knowledge level, preferences and culture. Beside these basic components, a *Goal Model* (GM) can represent the goal the users should meet and a *Domain Model* (DM) would represent the knowledge domain. *Adaptive strategies* are sets of adaptive rules that can be applied to these models, to allow the personalisation of the course for students, according to their needs.

From the many interacting elements, it is clear that the authoring process is a bottleneck in the adaptive course creation, which needs to be improved in terms of interoperability, usability and reuse of the adaptive behaviour (strategies). Authoring of Adaptive Hypermedia is considered to be difficult and time consuming. There is great scope for improving authoring tools in Adaptive Educational Hypermedia system, to aid already burdened authors to create adaptive courses easily.

Adaptation specifications are very useful in creating adaptive behaviours, to support the needs of a group of learners. Authors often lack the time or the skills needed to create new adaptation specifications from scratch. Creating an adaptation specification requires the author to know and remember the programming language syntax, which places a knowledge barrier for the author. LAG is a complete and useful programming language, which, however, is considered too complex for authors to deal with directly.

This thesis thus proposes a *visual framework (LAGBlocks) for the LAG adaptation language* and an *authoring tool (VASE)* to utilise the proposed visual framework, to create adaptive specifications, by manipulating visual elements. It is shown that the VASE authoring tool along with the visual framework

enables authors to create adaptive specifications with ease and assist authors in creating adaptive specifications which promote the “separation of concern”. The VASE authoring tool offers code completeness, correctness at design time, and also allows for adaptive strategies to be used within other tools for adaptive hypermedia. The goal is thus to make adaptive specifications easier, to create and to share for authors with little or no programming knowledge and experience.

This thesis looks at three aspects of authoring in adaptive educational hypermedia systems. The first aspect of the thesis is concerned with problems faced by the author of an adaptive hypermedia system; the second aspect is concerned with describing the findings gathered from investigating the previously developed authoring tools; and the final aspect of the thesis is concerned with the proposal, the implementation and the evaluation of a new authoring tool that improves the authoring process for authors with different knowledge, background and experience. The purpose of the new tool, VASE, is to enable authors to create adaptive strategies in a puzzle-building manner; moreover, the created adaptive strategies could be used within (are compatible with) other systems in adaptive hypermedia, which use the LAG programming language.

Abbreviations

| | |
|---------|--|
| AEH | Adaptive Educational Hypermedia |
| AH | Adaptive Hypermedia |
| AHA! | Adaptive Hypermedia for All! |
| AHAM | Adaptive Hypermedia Application Model |
| AHS | Adaptive Hypermedia Systems |
| ALE | Adaptive Learning Environment |
| AM | Adaptation Model |
| CAF | Common Adaptation Format |
| CAM | Concept Adaptation Model |
| CRT | Concept Relationship Type |
| DM | Domain Model |
| GAM | Generic Adaptivity Model |
| GAHM | Goldsmiths Adaptive Hypermedia Model |
| GAL | GRAPPLE Adaptation Language |
| GAT | GRAPPLE Authoring Tool |
| GM | Goal and Constraints Model |
| GRAPPLE | Generic Responsive Adaptive Personalized Learning Environment |
| HTML5 | Hypertext Mark-up Language (version 5) |
| LAG | Layers of Adaptation Granularity |
| LAG-XLS | LAG XML Learning Styles XXI |
| LAOS | Layered WWW AH Authoring Model and their corresponding Algebraic Operators |
| LMS | Learning Management System |
| LO | Learning Object |
| LOM | Learning Object Metadata |
| PEAL | Programming Environment for Adaptation Language |
| PM | Presentation Model |
| PRT | Pedagogical Relationship Type |
| UM | User Model |
| VASE | Visual Adaptive Strategy Environment |

1 Introduction

1.1 Background and Motivation

The number of internet users has grown exponentially: it has increased from 738 million in 2000 to 4.012 billion in 2018, up 7% year-on-year increase. The new 2018 Global Digital report from ‘We Are Social’ states that there are now more than 4 billion people around the world using the internet [1]. An increasing number of people, businesses, and establishments are turning to eLearning, as they appreciate its usefulness and its suitability. The global eLearning market is estimated to reach US\$325 billion by 2025, with growth of around 7.2% over the next decade [2].

However, current learning management systems (e.g. Blackboard [3], Sakai [4], etc.) still offer a static approach to the delivery of learning materials. This means that every learner is given the same set of learning material. Adaptive Educational Hypermedia (AEH) provides a more personalised and customised approach to the field of eLearning than the out-dated static methods.

AEH enhances the usability of hypermedia, by building a model of various qualities of a learner and applies this information to adapt the content and the navigation to the requirements of the learner. *Adaptive strategies* are sets of adaptive rules to indicate the conditions under which adaptation behaviours can be applied in AEH [5]. The AEH approach has been shown to be useful, as it displays more relevant content, according to the information stored in various models (user, goal and presentation model.)

However, a known issue in adaptive hypermedia is the authoring process. A review of the literature [6, 7, 8] revealed that the authoring process is a bottleneck in adaptive course creation and it needs to be improved, in terms of interoperability, usability and reuse of the adaptive behaviour (strategies). Adaptive Hypermedia authoring is considered to be challenging and laborious [9]. Thus, a hypermedia system should make it easy and natural for the already burdened authors to create adaptive courses easily. One way of tackling this is via the idea of creating an adaptive course once and using it many times with students with, e.g., different knowledge and preferences [10], or any other characteristics.

However, authoring adaptive materials is not a simple task, as an author may be pressed for time, or simply

lack the skills needed, to create new adaptive materials from scratch, and thus any improvements in the reuse of adaptation specification (application of adaptive behaviour rules) is a major help in the authoring process.

1.2 Aims

The research presented in this thesis aims *to create a new authoring tool for adaptive hypermedia authors with little or no programming experience*¹.

Firstly, through analysis of the existing authoring tools, the research aims to generate a set of requirements for a generic Adaptive Hypermedia Authoring System (AHAS). Then, this thesis attempts to address the problems faced by authors in creating adaptive strategies. This is achieved by allowing authors to create adaptive strategies in a puzzle-building manner, via a system called VASE. This thesis implements a visual programming technique to create a visual extension for authoring adaptive strategies in Adaptive Hypermedia.

To evaluate the new environment, the thesis compares a previous generic, complete authoring tool and the new one (PEAL versus VASE) and their authoring paradigms. Additionally, this research also investigates ways of extracting adaptive strategies that have been generated from authoring tools to make adaptive strategies interoperable, to be able to work across different systems.

1.3 Problem Description and Challenges

Even if the Adaptive Hypermedia approach has been proven to be useful [11], an author faces an array of problems to create an adaptive course and to reuse previously created material with different adaptation

¹ Please note that this also caters to authors who may have programming experience but lack the time necessary to learn a new programming language, required to create adaptive behaviour via adaptive strategies.

types. For instance, to create a personalised, rich learning experience for every student, first, the content of the lesson has to be prepared. Different options of the content have to be created for different students, allowing for different paths through the content. Metadata have to be added for the labelling and annotation of the different paths. Lastly, a mechanism must be defined, to guide the student through the different paths. This introduces a complex and convoluted authoring process, if no assistance is provided to the author by the authoring system.

Whilst adaptation specifications are very useful in creating adaptive behaviours, to support the needs of learners, authors often lack the skills needed to create new adaptation specifications from scratch [12]. A wide range of flexible adaptation specifications can only be created by using, as a base, an adaptive programming language, such as LAG [13] (Language for Adaptation Granularity). This poses however additional difficulties, as creating adaptation specifications requires the author to know and remember the programming language syntax [11].

Summarising, this research addresses the following major problems in relation to adaptive authoring;

- The current authoring process is a bottleneck in Adaptive Hypermedia, as demonstrated in the previous literature.
- There is a clear demand for better Adaptive Authoring tools. However:
 - Adaptive Strategies are too complex to create, especially for the novice author (non-programming-savvy authors);
 - Authors need to learn adaptation languages to perform adaptations.

1.4 Research Questions

In order to address the problem described in the previous sections, and as Adaptive Specifications are difficult to use, which creates a bottleneck in Adaptive Hypermedia, the following overarching research question was articulated, and then refined into sub-questions, to address each aspect of the problem and

provide a structured contribution.

RQ1 - What can be done to simplify and improve the adaptive specification authoring for adaptive hypermedia?

The research question above led to the following sub-questions.

1.4.1 Refined Research Questions

RQ 1.1 How can an adaptive authoring system render adaptive strategy creation easy for authors with different needs, e.g. knowledge and experience?

RQ 1.2 Are there benefits in using visual programming techniques to create adaptive specifications and if so, what would these benefits be?

RQ 1.3 How can an adaptive authoring system assist authors in an adaptive specification's correctness and completeness, as well as interoperability?

RQ 1.4 How can the visual programming techniques and technologies be implemented, in order to enhance the adaptive authoring system and adaptation language, and thus provide a high level of effectiveness, efficiency and satisfaction amongst authors?

1.5 Objectives

The research questions led to the following set of objectives, as below.

Objective 1 – *Conduct extensive theoretical background research into the area, to find the problems faced by the authors in adaptive strategy creation and to propose a theoretical framework to improve adaptive authoring for authors with different needs.*

Conduct an extensive background research in the research area to propose a theoretical framework which caters for different authors with different needs {RQ1.1}.

Objective 2 - *Get feedback from users for their preferred way of authoring an adaptive strategy.*

Conduct a survey which reflects the users' preferred ways to create adaptive programs {RQ1.1 & RQ1.2}.

Previously developed authoring tools for creating Adaptive Specifications were aimed at the programming-savvy author; this research focuses on creating an authoring tool which is suitable for non-programming-savvy authors - any author without any programming language knowledge. It is believed that this would increase the uptake of adaptive authoring amongst the authors who don't have any knowledge about any programming languages.

Objective 3: - *Propose a visual programming framework for implementation, to provide complete and correct adaptive specifications, and further sustain the "separation of concerns" principles, and interoperability.*

The principle of "separation of concerns" refers to allowing for reusability of the different components created during the (often, very cumbersome, and thus expensive) authoring process. As such, at a minimum, content should be separated from adaptive, personalised behaviour.

LAG [13] (further described in Section 2.4.5) has been chosen as the basis for building the visual framework upon, as it is the language which has been used by many adaptive hypermedia tools, for instance MOT [14], ADE [15], AHA! [16] and PEAL [17]. LAG is also one of the languages which enforces authoring imperative described in the section 2.7, which ensure, arguably, correctness and completeness. Thus, creating a visual framework compatible with LAG would automatically ensure correctness and completeness. This includes research into visual programming techniques to find if the LAG [9] language grammar can be represented visually, while keeping the compatibility with other LAG tools; to design new visual constructs to represent the entire LAG programming language {RQ1.3}. In other words, this means:

- To create the LAG visual *Adaptive Specification* by using visual programming techniques;
- To represent the entire LAG language grammar in visual programming techniques;
- To enable authors to create adaptive specifications in a puzzle-building manner.

Objective 4 – *Implement a visual programming framework to enhance adaptive authoring systems and adaptation languages. Propose a suitable framework for adaptive authoring, to allow authors to create visual adaptive specification, by manipulating visual elements to create adaptive specifications, thereby lowering the programming threshold for authors. {RQ1.4}*

The framework should support at least the following, but not be limited by them:

- To be able to create, edit and share existing adaptive specification with other authors;
- To create an authoring tool where visual blocks can be manipulated to create adaptive specifications with little knowledge of programming language;
- The adaptive authoring system based on it should be able to interface with at least one adaptive delivery engine which uses an adaptive programming language to apply adaptations. As a proof of concept, the tool will be interfaced with ADE [15] (Adaptive Delivery Engine), as ADE uses the LAG programming language to apply adaptation. This means that adaptive specifications created in should be able to execute in ADE.

Objective 5 - *Conduct a series of experiments that investigate the appropriate approach and features to design an adaptive authoring system, and to test the practical development of the newly added features in an adaptive authoring system, addressing the acceptance of the visual form of adaptive authoring in the evaluations.*

This objective includes all case studies performed, including comparing the new visual authoring tool with non-visual authoring tools, like PEAL {RQ1.4}.

1.6 Research Contributions

This thesis contributes in the following ways to the general research community.

- It has provided a **language visual extension** for LAG (LAGBlocks), to help the author to create adaptive strategies visually without writing any adaptation language syntax (see Chapter 5 Section 5.2).

- It creates an **authoring environment** (VASE), in which LAGBlocks can be manipulated, to create adaptive strategies. This adaptive authoring environment allows code completeness and correctness at design, which minimises mistakes in adaptive strategy creation (see Chapter 5 Section 5.5.1).
- It has developed a **Functionality Matrix** for role-based authoring in Adaptive Hypermedia (see Chapter 4 Section 4.2).
- The work has implemented a **LAG Converter** – to convert visual the representation of adaptive strategy to text, to make visual strategies interoperable with text based tools, such as ADE [15] (Adaptive Delivery Engine) (see Chapter 5 Section 5.5.3).

The table below summarises at a glance the mapping of the research objectives on the research questions, as well as on the research problem they address.

Table 1: Research Problem mapping to Research Questions and Objectives

| Research Problem | Research Question | Research Objective |
|---|-------------------|--------------------|
| To conduct extensive research into the area to explore the problems faced by the authors in adaptive strategy creation and reuse. | RQ 1.1 | Objective 1 |
| To get feedback from the potential users about the preferred way for adaptive strategy authoring. | RQ 1.1, RQ 1.2 | Objective 2 |
| Propose a visual programming framework to improve on adaptive specification authoring. | RQ 1.3 | Objective 3 |
| To implement a visual programming framework to create an adaptive authoring tool. | RQ 1.4 | Objective 4 |
| To compare the visual authoring tool with a non-visual authoring tool of equivalent completeness and correctness. | RQ 1.3, RQ 1.4 | Objective 5 |

1.7 Thesis Outline

The remainder of this thesis is structured as follows. **Chapter 2** discusses general issues surrounding authoring in Adaptive Hypermedia Systems, existing definitions, frameworks and issues concerning authoring in adaptive educational hypermedia systems. **Chapter 3** describes the methodology used in this research. **Chapter 4** describes the user-roles and the proposed functionality matrix implementation in

Adaptive Hypermedia. **Chapter 5** reports on the initial design and demonstrates the design and the redesign of the authoring tool following the feedback from the user evaluations. **Chapter 6** contains the evaluation of the proposed visual authoring system in its many iterations and, finally, **Chapter 7** provides conclusion of this research, it also presents the future work.

2 Background and Related Research

2.1 Introduction

Adaptive Hypermedia (AH) [2] enhances the usability of hypermedia, by constructing a model of various characteristics of a user, such as knowledge and preferences, and applies this information to alter the content, to suit, e.g., the needs of a user. In contrast to conventional e-learning systems, where all users are presented the same series of hyperlinks, adaptive hypermedia adapts what the learner sees to the learner's abilities, goals, interests and knowledge [4].

However, although the AH approach has been shown to be beneficial [2], an author faces an array of problems, in order to create an adaptive course and to reuse the material created earlier with different adaptation types. For instance, in order to create a rich learning and personalised experience for every user, firstly, the content of the lesson has to be prepared. Different alternatives of the content have to be created for different users; this leads to different paths through the content. Metadata need to be attached for the annotation and labelling of the different paths. Lastly, a method must be defined to guide the user through the different paths. This introduces an inefficient and convoluted authoring process [10].

Authors may often lack the skills needed to create new adaptation materials from scratch, and hence any improvement in the reusability of adaptation strategies (the adaptive behaviour) is a major help in the authoring process. Several authoring tools have been designed to help authors in adaptive courseware creation. However, most of these tools are designed to address a niche problem in the area and don't address most of the problems experienced by the majority of authors. This chapter provides a brief introduction to some of the main features offered by the most popular Adaptive Educational Hypermedia systems developed earlier. It also highlights prominent issues surrounding the authoring of adaptive educational hypermedia systems in academia.

Outcomes of this Chapter: This chapter works as the theoretical background for all the work proposed in this thesis. It also explores theoretical ideas, techniques, methods and implementation

solutions for the research problems discussed in Chapter 1, as well as responding to: **Objective 1:** *Conduct an extensive theoretical background research into the area, to find the problems faced by the authors in adaptive strategy creation and to propose a theoretical framework to improve adaptive authoring for authors with different needs.*

2.2 What is an Adaptive Educational Hypermedia System?

Hypertext” refers to a set of information nodes connected with a set of links in between the nodes, when the information stored in the nodes are not only text, but also graphics, image, audio, animation and video, is known as “Hypermedia”. The main strength of hypermedia systems is that they have a flexible structure and give user great deal of freedom to browse and interact with the information contained within them. It is precisely this freedom that gives rise to many problems; one of the problems that seem inherent within hypermedia systems is that users tend to lose their way in the maze of information within the system. This is commonly referred to as the ‘lost in hyperspace’.

Adaptive Educational Hypermedia (AEH) adapts what the user sees to the student’s goals, abilities, interests, knowledge and preferences. Adaptive Hypermedia is the solution to the ‘lost in hyperspace’ problem, where the user has too many hyperlinks to choose from, and has limited knowledge to select the most appropriate hyperlink. “Adaptive Educational Hypermedia (AEH) personalises presentation and content for every student in order to deliver the suitable material to each student”, providing a solution to the problems of traditional hypermedia systems such as the ‘one-size-fits-all’ approach [18]. Adaptive Hypermedia is a generalised pattern for web pages which can adapt the content towards the users’ need. Instead of each user being presented with the same content and hyperlinks, content can be modified to each individual user or group of users, which is especially helpful in educational environments [11, 8, 19]. AEH presents a collection of links and content that is most suitable to the current user.

In an Adaptive Educational Hypermedia (AEH) course, a student’s learning experiences can be personalised using a *User Model (UM)* [13] which could include information such as the student’s knowledge level, preferences and culture. A *Goal Model (GM)* [13] can be used to store a user’s goals

that need to be achieved, often dictated by the institution or the student's own learning objectives and a *Domain Model (DM)* [20] is used to represent the knowledge domain or the subject area, for instance, sports, medical etc. User Model, Goal Model and the Domain Model are manipulated by *adaptive strategies* [20], to create and deliver an adaptive course for students with different needs and preferences. The adaptive strategy creation, as we shall see, still form the most difficult challenge in the already cumbersome process of authoring of adaptive educational hypermedia, which is considered next.

2.3 Authoring of Adaptive Educational Hypermedia Systems

Adaptive Hypermedia enhances the usability of hypermedia, by building a model of various traits of an individual user, such as knowledge and preferences, and applies this information to modify the content to the needs of the user. When AH is applied in an education domain, the student's learning experiences can be personalised, by using a user model, which might incorporate information such as the student's knowledge level, preferences, and academic and personal goals. Various Adaptive Educational Hypermedia systems have been developed and tested in different domains and proven their usefulness for improved learning experience in teaching [11, 6]. However, authoring of adaptive courseware for learners with different needs still remains convoluted. There is a genuine need to provide better support for authors in AEH systems [2]. Creating courseware in an Adaptive Hypermedia System (AHS) is considered to be difficult and requires time and effort from the author, which makes the authoring process a bottleneck, and assistance must be provided to help the author during the authoring process. The reason adaptive authoring is considered to be a bottleneck is because it involves definitions of both the adaptive content (the content to be used in the AHS) and the adaptation behaviour (or adaptive strategies, as mentioned at the end of the last section, containing a set of adaptive rules defining how that content will be used in the AHS). In order for adaptive hypermedia to be developed, frameworks are needed, to underpin the structure and adaptation methods of the AHS.

2.4 Frameworks for Adaptive Hypermedia

Various frameworks have been designed to simplify the design of AH systems. Only relevant frameworks to this research are presented here.

2.4.1 AHAM (Adaptive Hypermedia Application Model)

The AHAM model [21], [22] (see [Figure 1](#)) extends the Dexter hypermedia model to define adaptation [23] and is one of the basic and most well-known frameworks of adaptive hypermedia. Dexter consists of a 'Runtime layer, 'Storage Layer; and a 'Within Component Layer'. AHAM extends this through the definition of three models. 'Domain Model' stores fragments of information and how these fragments are structured. The Domain model contains concepts, with each concept containing a number of attributes. The 'User Model' stores information about the user, information that describes the user's knowledge of the concept. The 'Adaptation Model' describes how the relationships between concepts should be interpreted, to form pedagogical relationships that interact with the user model and can be used to guide the user in a course.

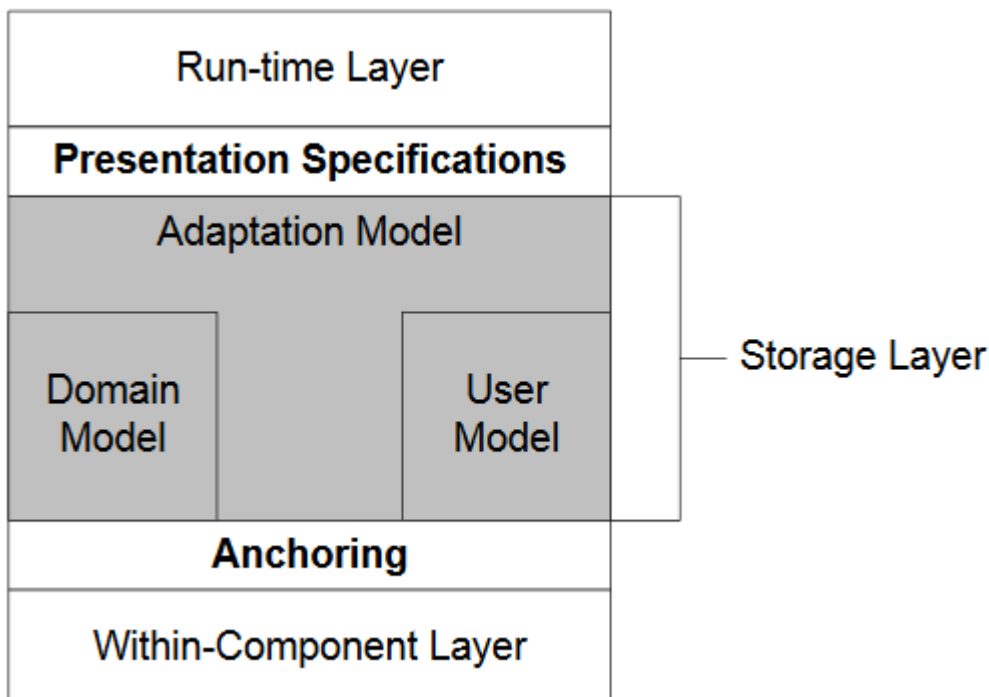


Figure 1: Adaptive Hypermedia Application Model (AHAM)

2.4.2 Munich Reference Model

The Munich Reference Model [24] uses UML to specify three separate meta models: the ‘domain meta-model’, the ‘adaptation meta-model’ and the ‘user meta-model’. The Munich model’s meta-models all exists in the ‘Storage Layer’ of the Dexter hypermedia model. Otherwise, this model follows exactly the structure of the AHAM model, and can be used in a similar fashion.

2.4.3 LAOS (Layered Authoring-Model and Operators)

LAOS (Layered AHS Authoring-Model and Operators) is an AH authoring model (see [20]).

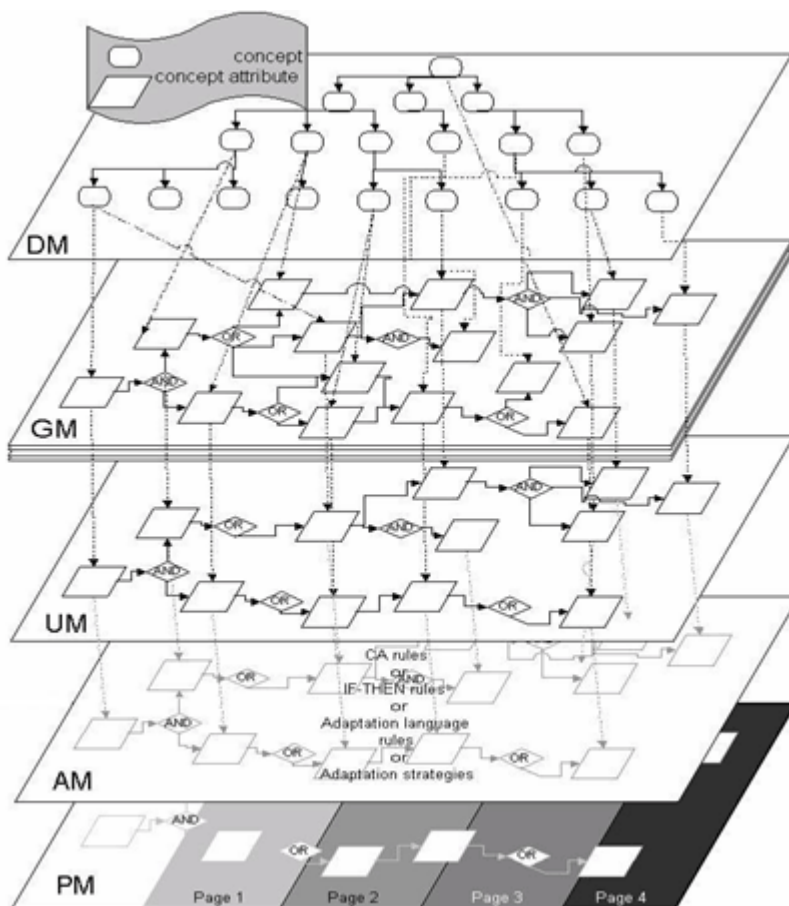


Figure 2: LAOS framework (5 Layers) [20]

Thus, the first difference to the AHAM and Munich models is its purpose, which is that of authoring, and not of modelling generic adaptive hypermedia. The LAOS model divides key components of AHS into five layers and is meant to be both a generic and a dynamic method of authoring AHS. The following five layers are found in the LAOS model [25]:

1. Domain Model (the content/media)
2. Goal Model (the objectives)
3. User Model (the users)
4. Adaptation Model (the adaptation)
5. Presentation Model (the presentation).

They are further described in the following sections in greater detail, as it is the main model this work is based on, due to it being on one hand the most expanded framework at the time, as well as being the framework which the LAG language has been built upon.

2.4.3.1 *Domain Model (DM)*

The *Domain Model* describes the resources of the hyper-document that are needed for the AHS. These resources represent the content that is usually authored in any learning environment. A hierarchy of knowledge concepts (with attributes) is used to identify the content for the AHS [25].

2.4.3.2 *Goal and Constraints Model (GM)*

The *Goal Model* contains information about intentions or objectives of the course. “The goal and constraints model contains all information (resources and links between them) of the adaptive hypermedia system important for a delivery purpose. According to LAOS, within the educational domain, this model filters, regroups and restructures the domain model, with respect to an Instructional goal.” [25]. This model provides the organisational information for the AHS and gives structure to the Domain Model.

2.4.3.3 *User Model (UM)*

The User Model stores attributes of the user, such as their specific interests or existing knowledge. “The user maps should contain all the necessary variables and initial values that are necessary to represent the user (for educational applications, the learner). Typical variables are knowledge level, interests, learning styles, etc. The user model can be an overlay to either the goal and constraints or the domain model (e.g., the knowledge of a certain topic represented by a goal and constraints model concept). Alternatively, the user model can also contain free variables (such as background knowledge)” [25]. The information in this model is used to specify what is relevant in the Goal Model and Domain Model for the current user.

2.4.3.4 *Adaptation Model (AM)*

The Adaptation Model defines the adaptation of the content in the AHS – this is further defined in the LAG model. “The adaptation model is the only part of the LAOS model describing the dynamics of the adaptation process. Whilst the other layers are concerned with the static properties, variables and values required for personalization, the adaptation model brings them together and specifies how they will be used. Hence we can consider that the other sub-models in LAOS specify the ingredients of the personalization process, whilst the adaptation model specifies the recipe. The adaptation model itself is based on the LAG model, the model of three layers of granularity” [25]. This model brings together all the other models and defines their interaction for production of the final AHS.

2.4.3.5 *The Presentation Model (PM)*

The Presentation Model contains information about adaptation for different devices on which the AHS will be displayed “Presentation has to take into account the physical properties and the environment of the presentation and provide the bridge to the actual code generation for the different platforms. Presentation makes the difference between different devices of display, such as handheld devices, desktops, laptops, etc. This part in the LAOS model is concerned with the formatting so that the information appears nicely in the page...” [25]. This model contains device specific information about the aesthetic display of the AHS.

Combined, these five layers build up a coherent model of the AHS, where variables in the system belong to different models and define factors including the content, user profile, adaptation rules and presentation details. LAOS allows for multi-multi relations of any type; however, the current implemented language only supports the relatedness relations, for compatibility with the MOT authoring environment.

In short, the **five semantic layers of LAOS** are the following.

1. Domain model (DM), containing the basic concepts of the contents, and their representation (such as learning resources).

2. Goal and constraints model (GM), a constrained version of the domain model. The constraints are based on educational goals and motivations.
3. User model (UM), represents a model of the learner's educational traits.
4. Adaptation model (AM), a more complex layer that determines the dynamics of the AH system. Traditionally, this layer is composed of IF-THEN rules and therefore the LAOS version also translates such rules at the lowest level.
5. Presentation model (PM), is provided to reflect the physical properties and the environment of the presentation; it reflects choices, such as, the appropriate background contrast to support a learner with poor eyesight.

Each of these semantic layers is composed of semantic elements. LAOS allows flexible composition of the defining semantic elements of the layers, according to each learner's personalization requirements.

2.4.4 The LAG model

The fourth model in LAOS, the Adaptation model, is further divided into three layers, according to the LAG model [13], standing for stands for 'Layers Adaptation Granularity': adaptation strategies, adaptation programming constructs and adaptation IF-THEN basic rules. This is to help with the conceptualisation and instantiation of the adaptation behaviour, which in the past has been embedded into other models, or not modelled at all (and only left to be defined at programming time). The LAG model provides thus three levels of user-interaction in the authoring process (adapting to different user skill levels): Adaptation Strategies for the layman author, Adaptation Language for competent computer-literate authors, and Adaptation Assembly Language for the most 'hard-core' of computer-literate authors.

This multi-layered approach is beneficial in two ways: firstly, authors can choose to work at the most suitable level for them and secondly, LAG provides a "step towards standardization of adaptive techniques" which permits techniques of adaption to be exchanged between applications as well as

standardised adaptive authoring [13]. LAG is based on the same layered structure as LAOS [26] and provides interaction with all five layers in the form of variables.

This LAG model is also the basis of the LAG adaptation language, which is further described in section 2.4.5.

2.4.5 The LAG language

The LAG language [11] (see also APPENDIX A: LAG Grammar and APPENDIX B: LAG Grammar Semantics) is an authoring language for adaptive behaviour, initially defined at the Technical University of Eindhoven, and further being developed at the University of Warwick. It is based on the LAG model [13] (see also section 2.4.4), instantiating its second layer: that of an adaptation programming language. The LAG language is described as an adaptation language, since it is a language that is adaptable to the author and it also describes the adaptation of content in adaptive hypermedia. This is the language on the basis of which this project is being developed, due to the fact that it is the most complete adaptation language currently known, besides producing correct adaptation specifications. Thus, if the visual environment to be created can reproduce its power and be converted to it, it will have the same expressivity, with the hope that it will be, however, much more usable by authors.

Each LAG strategy defines the adaptation behaviour for a given set of knowledge concepts (for example, questions and answers or topics and lessons). Each strategy must contain two sections: an initialization and an implementation. The initialization defines the initial state of the AHS, allowing the author to differentiate between users and provide the correct content, objectives and adaptation rules. The implementation defines the inner workings of the AHS. Here is where the adaptation of content is specified, most often through user navigation in the hyper-document.

Using the same if-else structure as found in many C style programming languages (as per third LAG model layer - see also section 2.4.4), the author can specify clearly which knowledge concepts are applicable to which users and thus which concepts should be displayed after what interaction by the user. LAG is

designed to be content independent, as long as the concepts labels, or titles, remain constant between different versions of the adaptive content. This is particularly useful for AHS where many different versions of the content are required, as well as for reuse of the adaptation behaviour. In the following, the LAG language will be illustrated based on a simple strategy.

2.4.5.1 Beginner-Intermediate-Advanced LAG adaptation strategy

The Beginner-Intermediate-Advanced strategy in LAG strategy shows concepts based on their weights and labels. The idea in the LAOS framework [26] is that weights and labels are added in a separate layer from the domain map (an instantiation of the Domain model), in the goal and constraints map (GM) (an instantiation of the Goal and constraints model), which represents pedagogic knowledge. Thus, a possible pedagogical division is to label concepts as beginner, intermediate, or advanced. Labelling these concepts outside the domain model means that a different labelling is possible for the same domain model concepts, for a different instance of the goal and constraints map. Alternatively, the pedagogical strategy created can be reused for different domain models. A simplification of this strategy (some of the basic rules) is shown below:

```
IF UM.level == beginner AND GM.Concept.label == beginner THEN
PM.GM.Concept.show = true
IF UM.level == intermediate AND GM.Concept.label == intermediate THEN
PM.GM.Concept.show = true
IF UM.level == advanced AND GM.Concept.label == advanced THEN
PM.GM.Concept.show = true
```

The snippet above decrees that beginner users should be shown beginner labelled concepts, intermediate users should be shown intermediate concepts, and, finally, advanced users should be shown advanced concepts. The full strategy can be found in the APPENDIX H: LAG , Beginner-Intermediate-Advanced.

2.4.5.2 LAG language versus XML languages

LAG is an adaptation programming language, and, as such, it follows the structural programming language structure. On the other hand, it is designed for the web, and thus is needed also to be exchanged between different systems, e.g., a client and a server. XML languages, on the other hand, are well-known to be excellent for system-to-system conversions. These are very verbose to be used directly to program by

hand, but they are certainly useful for interfacing. Therefore, export-import into such formats is desired. The good news is that converting the LAG language to an XML specification is relatively straightforward: for each LAG construct, an XML element can be defined, with sub-elements that enforce the grammar. Thus, all the parts of a LAG adaptation strategy need to be converted, i.e.:

```
<description> the description of the adaptive strategy
</description>
<initialization> user's initial view of the system
</initialization>
<implementation> the interaction loop user system
</implementation>
```

To illustrate this conversion, a code snippet containing a more complicated if condition, such as the one using the 'enough' construct in a type-based adaptation, is shown below:

```
IF enough2(UM.GM.showall>3, PM.GM.Concept.type==conclusion, 2)
THEN PM.GM.Concept.show = true
```

The code states that if the user model showall variable has reached the value 3 or more, the user should be shown the conclusions. The equivalent potential XML code is shown below. Clearly, the XML format is more verbose and shouldn't be used in direct programming by hand.

```
<if>
<enough number="2">
<condition>
<attribute> UM.GM.showall </attribute>
<operator> &gt; </operator> <value> 3 </value>
</condition>
<condition>
<attribute> PM.GM.Concept.type </attribute>
<operator> == </operator> <value> conclusion </value>
</condition>
```

² 'enough(cond1, cond2, ..., condn), ensures that conditions are fulfilled out of the n conditions listed.

```
</enough>
<then>
<attribute> PM.GM.Concept.show </attribute>
<operator> = </operator> <value> true </value>
</then>
</if>
```

Other adaptation languages, such as LAG-XLS [19], are represented directly in XML. LAG-XLS suffers however of the same issue as the code illustrated above, in terms of verbosity. Additionally, LAG-XLS supports different learning styles, but would require further development to support more extensive personalisation and collaboration.

Thus, given its generic nature, completeness and correctness, the LAG language is used as basis for the visual system developed further in this thesis (see Chapter 5). In the following, systems aimed at authoring adaptive hypermedia are visited and described.

2.5 Authoring Systems for Adaptive Hypermedia

The authoring systems presented here are the ones that have had a major impact on the research field.

2.5.1 Interbook

InterBook (see Figure 3) is one of the earliest authoring systems for adaptive hypermedia. It implements the ELM-ART Architecture [27]. InterBook [28] uses a specific concept-based approach and is authored via a Microsoft Word file. Domain concepts are automatically identified, according to the headings within the Microsoft Word document. Using this popular file format was designed specifically to make the authoring burden easier – by using a well-known and familiar environment. However, special annotations (comments in the file) must be used, to specify related and background (pre-requisite) concepts. The metaphor behind the InterBook approach is an electronic textbook, which can be any hierarchical structured hypermedia. An electronic textbook is also one of the most popular metaphors for representing online course material. Almost any kind of course material can be represented as an electronic textbook.

During the authoring process, after editing the Microsoft Word document, the document is converted to an HTML file and the annotations converted into adaptation rules. Further adaptation rules can then be added by the author by using the LISP programming language. Content is not disjointed from adaptation rules

during authoring and thus the scope for reuse is limited, as reuse relies on authors changing the adaptation rules. This puts such reuse outside the range of beginner users (non-programmers who are not able to use the LISP programming language) and limits the range of adaptation for novice users.

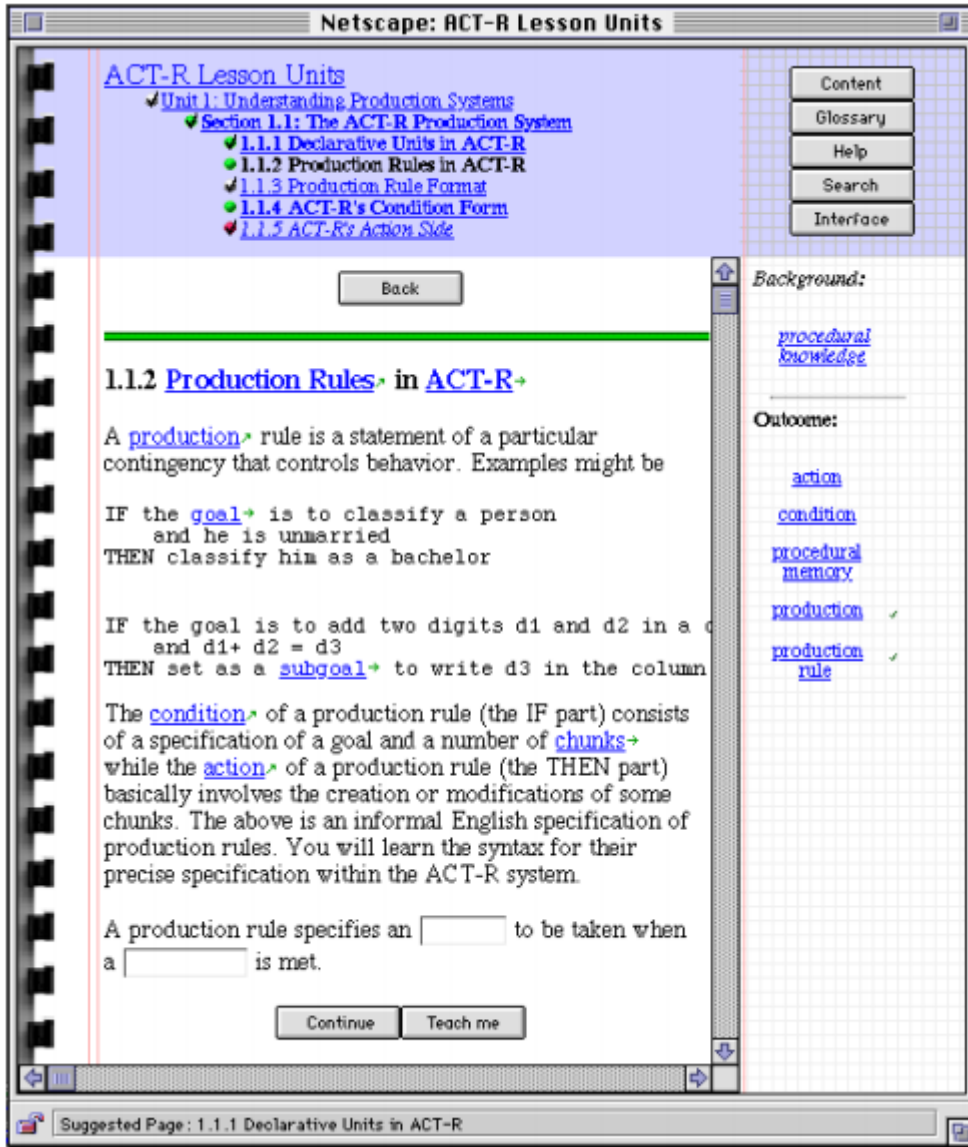


Figure 3: An example of course material shown in InterBook in the Netscape Browser [28]

2.5.2 AHA! (Adaptive Hypermedia Architecture)

AHA! [29] was an open source project, with a web-based adaptive engine built on Java servlet technology. It offers authoring through Java Applets with general purpose user-model and adaptation rules. It uses XML extensively and supports a MySQL database. AHA! provides *content adaptation*, by conditionally

selecting pages, fragments or objects and through *link adaptation*, by conditionally changing the colour of link anchors and adding icons.

In Figure 4, the adaptation rule form shows that for a rule, its condition can be edited, as well as the action to be performed if the condition is true and the optional action, or if the condition is false. Through the "Propagating" flag one can indicate that changes made to the user model by this rule may trigger other rules.

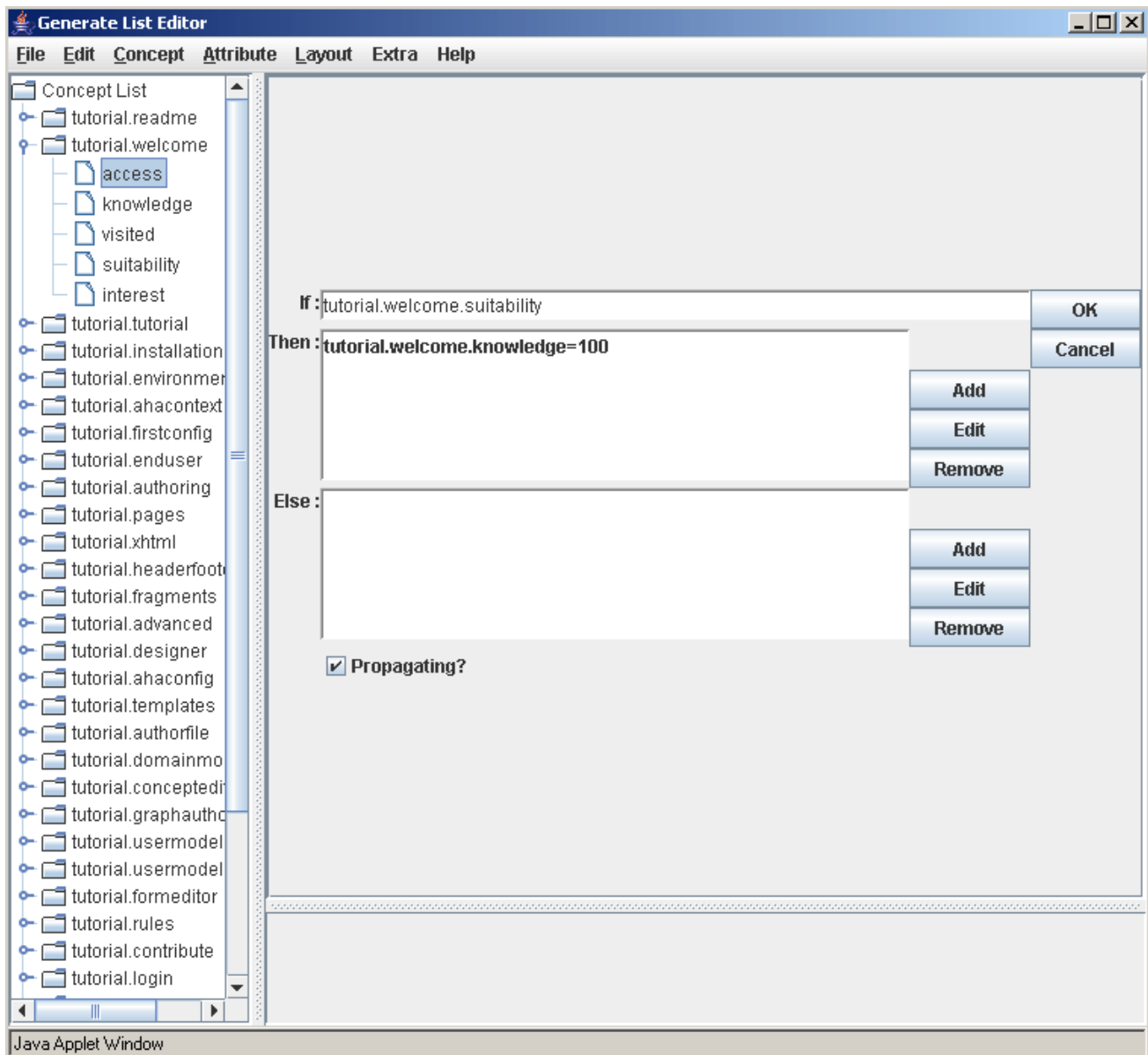


Figure 4: AHA Adaptation Rules Form

As can be seen from the above description, AHA! came with its rich and relatively complete (for the time) authoring support system. Some of the tools had easy to use, drag-and-drop interfaces, and familiar paradigms for the users. The AHA! authoring system introduces a number of concepts which are useful, such as graph-based authoring and frame-based input of data, which are all aimed at making the authoring burden easier. However, the many different tools and the different levels of complexity meant that authoring with the whole toolset was not accessible for novice authors. Moreover, the adaptation still requires programming skills, and is difficult.

2.5.3 Adaptive Course Construction Toolkit (ACCT)

The Adaptive Course Construction Toolkit (ACCT) [30] was created to provide the course developer with tools to design, test and deploy adaptive personalised eLearning, based on pedagogical support and instructional design principles.

2.5.3.1 Narrative Structures in ACCT

ACCT is built on ‘narrative structures’, which are model-based representations of pedagogical principles, which can be further used as templates. For this purpose, an XML format was used:

```
<concept id="030" type="model" xmlns:xlink="http://..">  
  
  <name lang="en"> Webquest </name> ...  
  
</concept>
```

2.5.3.2 Subject Concept Space Editor (SMCS) in ACCT

The Subject Matter Concept Space Editor (SMCS) is used to create the relationships and inter-relationships within a domain of information (see Figure 5).

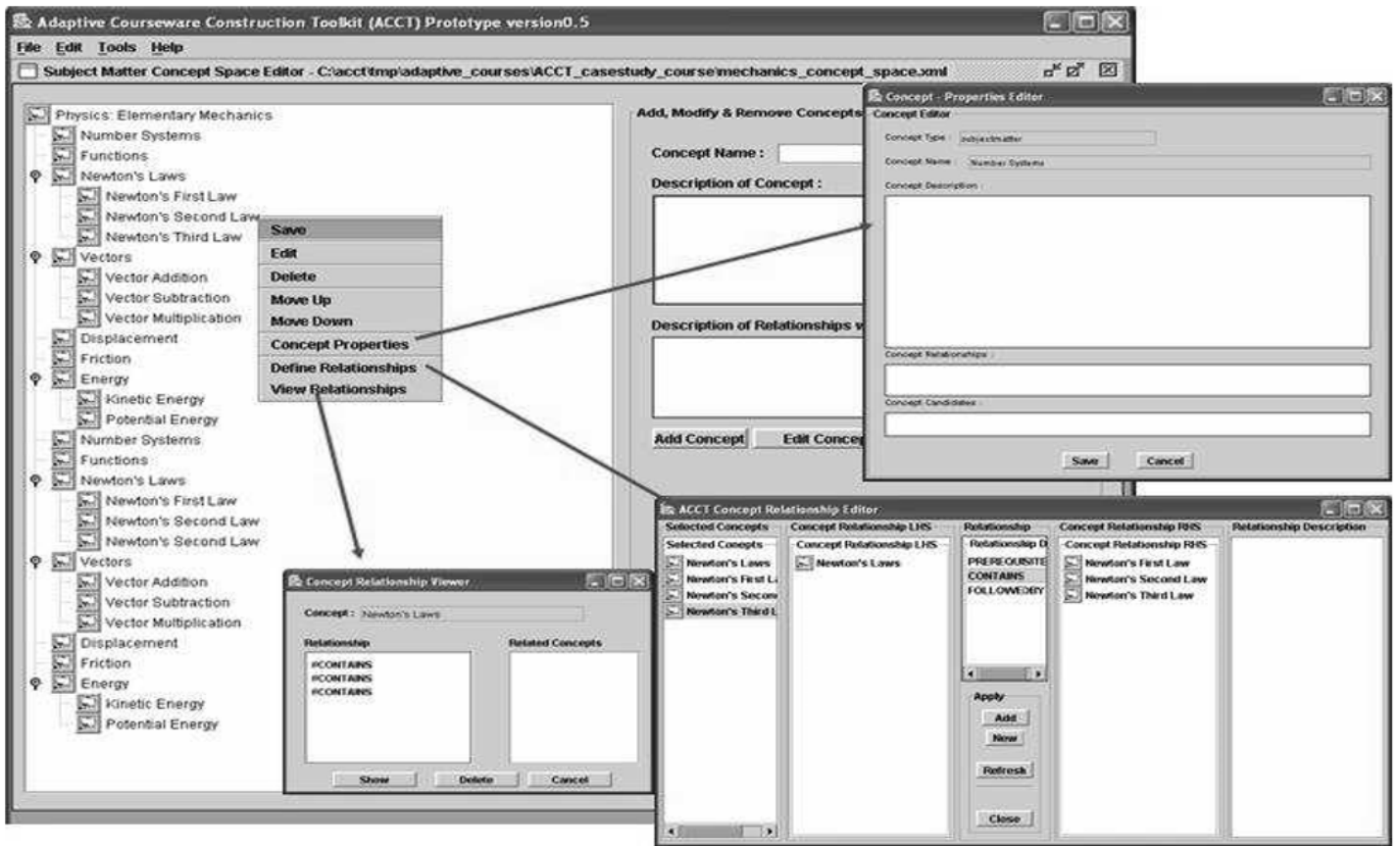


Figure 5: Subject Matter Concept Space Editor (SMCS) in ACCT

2.5.3.3 Narrative Model Builder in ACCT

The Narrative Model Builder is the one used to create narrative structures (see Figure 6).

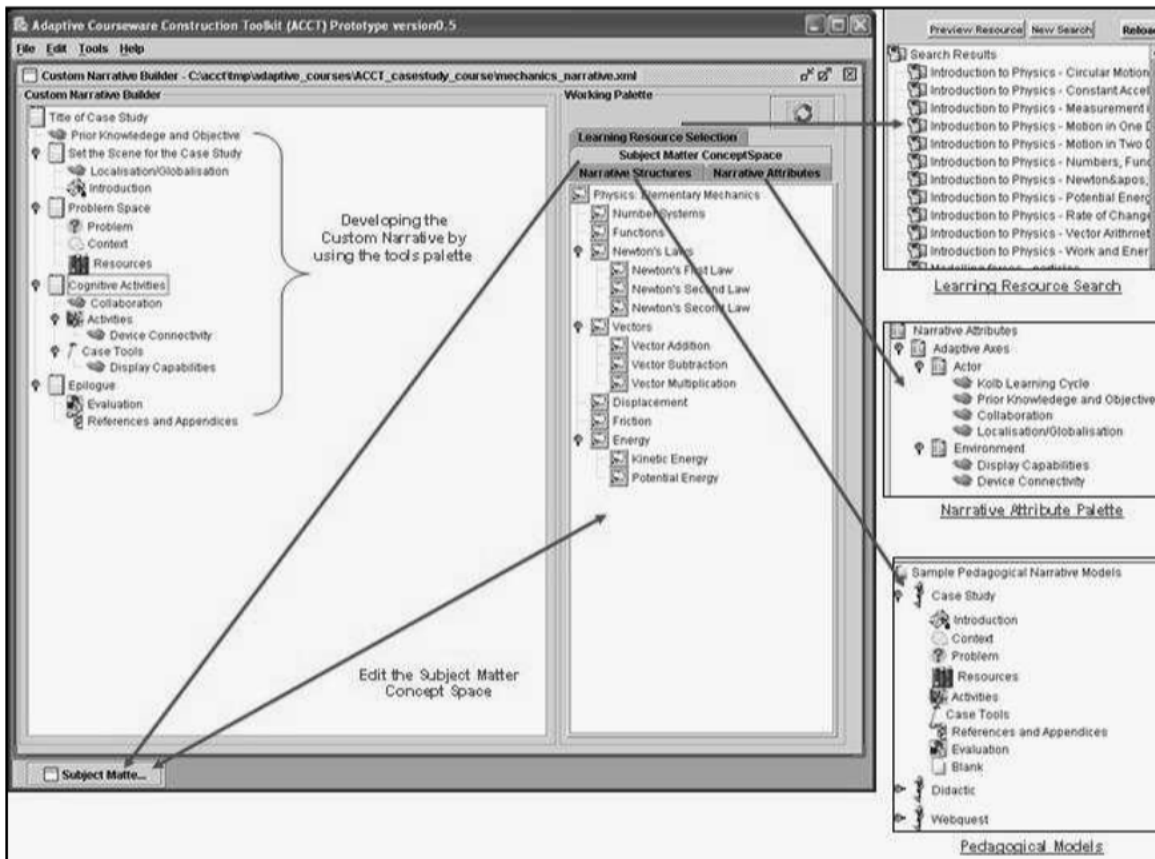


Figure 6: Narrative Model Builder in ACCT

The ACCT provides the course developer with the ability to verify their course in real-time through the use of an APeLS service interface, which is an important point to mention.

The ACCT clearly also supports the paradigm of separation of concerns. It also uses drag&drop interfaces. The adaptation is however still to be written in XML (as a narrative). Also, the type of adaptation supported by ACCT is relatively reduced.

2.5.4 Adaptive Personalized eLearning Service (APeLS)

APeLS [31] is a personalised eLearning service based on a generic adaptive engine. Authors evaluate the usability and effectiveness of using the multi-model, metadata-driven approach for producing rich adaptive eLearning solutions that remain content and domain independent. One of the strengths of APeLS is that it can utilise many pedagogical approaches and a variety of models to produce a wide range of highly flexible solutions.

At Trinity College Dublin, the Adaptive Personalised eLearning Services (APeLS) was used for 4 years as a part of undergraduate degree level course. Initial findings from the student evaluation show that despite the majority of students having little or no experience of online learning or the subject matter, the introduction of the personalized SQL course in 2000 had a beneficial effect on students' performance in the final examinations. Most students felt that the courses generated based on the prior knowledge pretest instrument produced courses they expected and desired, however approximately 40% of students stated that they desired more control over the course content. This was reflected in the comments of a number of students where they stated that they 'played' the pre-test instrument to create short personalised courses that could be tackled in short periods of time. They would re-personalize the course for each session with this aim. This finding highlights the innovation some learners can display in utilizing personalized eLearning. In terms of authoring, APeLS followed the model of ACCT, in that it has a Narrative Model, and additionally introduced a terminal model (see Figure 7).

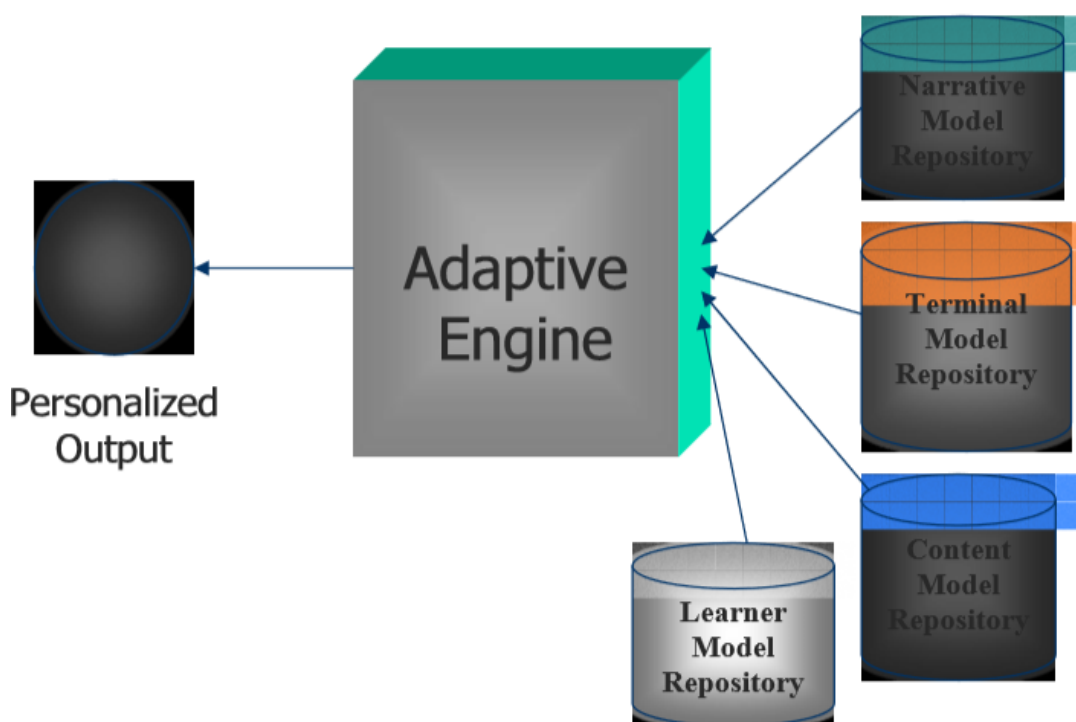


Figure 7: APeLS architecture

Terminal descriptions need to be created in XML, similar to the narrative model in both ACCT and APeLS:

```

<terminal type="high_spec">
  <browser>
    <screen_size>1280x1024</screen_size>
    <color>yes</color>
    <sound>yes</sound>
    <scrollable>yes</scrollable>
    <CcppAccept>
      <li>text/html</li>
      <li>text/plain</li>
      <li>image/jpeg</li>
      <li>..... .</li>
    </CcppAccept>
  </browser>
  <network> ..... </network> .....
</terminal>

```

There is no conclusive information on how an author would need to create the information required for all these components. It is possible that it requires authors to create XML snippets, including for (relatively primitive, similar to ACCT) adaptive behaviour specification. What is clear is that there is a modular approach, with separation of concerns.

2.5.5 ACTSim (A Composition Tool for Authoring Adaptive Soft Skill)

ACTSim [32] teaches interpersonal relationship skills such as interviewing, sales or telecommunication skills. It separates the content and adaptivity of the simulation. This allows the adaptivity to be altered and the content to be reused. Content is initially developed and adaptivity is placed across the content, ensuring the two remain separate. Soft skill simulations operate upon the dialogue, which occurs within the simulation. This involves the dialogue being decomposed into smaller components (nodes), which relate to arrows (edges) to form a graph (see Figure 8).

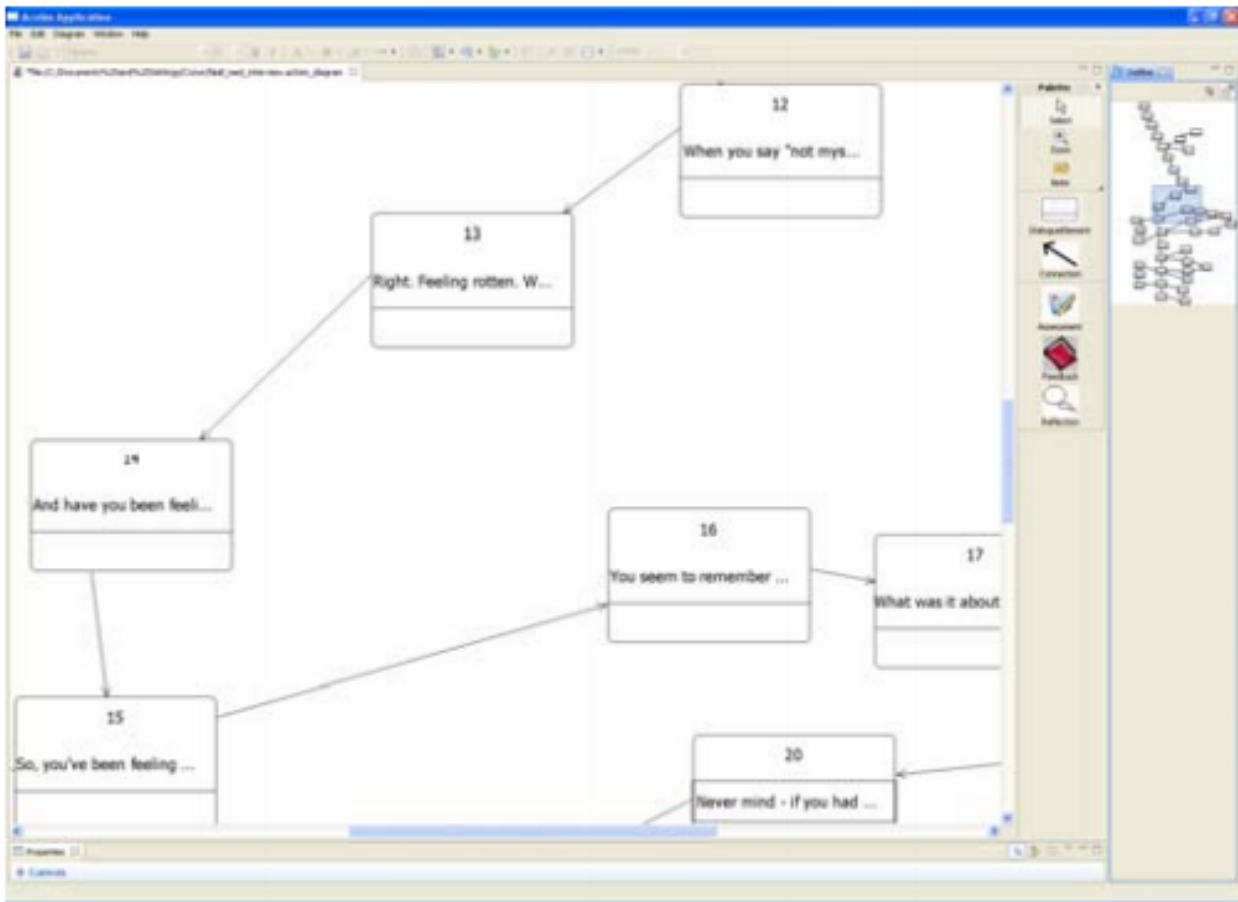


Figure 8: ACTSim Composition Tool

Navigational aids were also required by the composition tool to assist the author moving across the dialogue model. These are particularly useful if the simulation becomes large and complex. Navigational aids include a map of the dialogue model, zoom functionality and an ‘arrange’ feature. The results for this were very positive with authors describing the composition tool as “intuitive” and “user friendly”. This approach requires an authoring process to alleviate the complexity of composing the simulations. This authoring process not only needs a methodology for creating the models but also incorporates a pedagogical frame work so as to support educationally sound simulations.

Thus, as can be seen, graphical paradigms exist in ACTSim, as well as the idea of separating the adaptivity and the content. However, placing the adaptivity on top of the content means that the adaptivity would only be applicable to that specific content. Also, visual ways of creating the adaptation specification are not considered.

2.5.6 The Generic Responsive Adaptive Personalized Learning Environment (GRAPPLE) and the Grapple Authoring Tool (GAT)

GRAPPLE [33] is an enhanced learning environment that guides learners through the learning experience, automatically adapting to personal preferences, prior knowledge and learning goals, amongst others. GRAPPLE includes authoring tools to create adaptive learning material for the learners. The Grapple Authoring Tool (GAT) [34] has three main components: a Domain Model authoring tool (DM), for creating a conceptual representation of an application domain (or "course"), a Pedagogical Relationship, a Type authoring tool (PRT), for defining types of pedagogical relationships between concepts and their associated adaptation, and a Conceptual Adaptation Model (CAM) authoring tool (also called 'Course tool') for defining the pedagogical structure of a course. The GAT toolset is set up to allow for very general types of relationships and adaptation rules. Adaptation should be based on many different aspects, including the user's background, knowledge and goals, but possibly also on the characteristics of the device used for learning.

Summarising, the Grapple Authoring Tool (GAT) has three main components.

1. Domain model authoring tool (DM)

For creating a conceptual representation of an application domain (or "course").

2. Pedagogical relationship type authoring tool (PRT).

For defining types of pedagogical relationships between concepts and their associated adaptation.

3. Conceptual adaptation model (CAM)

Authoring tool (also called 'Course tool') for defining the pedagogical structure of a course.

GAT has different interesting features: it allows for separation of concerns, by separating domain and adaptation. In fact, adaptation, as said, is further separated into pedagogical, reusable relationships, and the strategies – the conceptual adaptation models. GAT uses drag&drop interfaces and even has a different interface for beginner authors and one for advanced authors. However, GAT has the same issue as other adaptation strategy creating tools, which is that the strategies themselves are not reusable – as they refer to some specific content (only the pedagogical relationships are reusable). This puts a considerable burden

on GAT authors, and, unsurprisingly, evaluations of GAT have shown the difficulty of authoring the behavioural part [35], [8].

2.5.6.1 Authoring pages in GRAPPLE

Authoring in GRAPPLE [36] is flexible in many ways. Figure 9 below shows how the ‘leaves’ of the graphs or hierarchical structures– the ultimate pages – can be authored in GRAPPLE. GRAPPLE introduces the notion of using *template pages* instead of creating each course page by hand. Figure 9 shows the difference between the two authoring processes:

- On the one hand writing (or importing) individual pages separately is easy, as it only requires basic knowledge of XHTML (or a web page authoring tool). It requires discipline to ensure that all pages that should have the same layout indeed do, and it involves repetitive work if later that layout needs to be changed (everywhere).
- Using *template pages* ensures that through a single template all pages that should have the same layout and presentation are consistent, and any change to the layout automatically applies to all pages based on the same template.

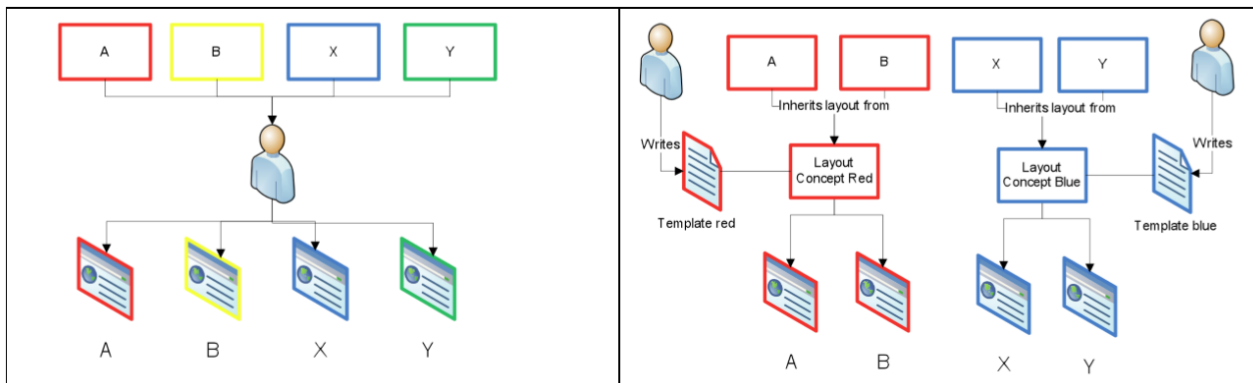


Figure 9: Authoring Pages Separately (Left) or via *Template Pages* (Right)


A typical page of a course is shown in Figure 10. This page describes the planet Jupiter. Pages describing other planets look very similar because they are all based on the same template.

Paul De Bra (debra@win.tue.nl) has read 7 pages and still has 22 to read - [list of read pages](#) - [pages still to be read](#)
Options in stand-alone mode: [change password](#) [logout](#)

Jupiter

Is Planet of: [Sun](#)

Image of Jupiter



Information

Jupiter is the fifth planet from the Sun and the largest planet within the Solar System.[10] It is two and a half times as massive as all of the other planets in our Solar System combined. Jupiter is classified as a gas giant, along with Saturn, Uranus and Neptune. Together, these four planets are sometimes referred to as the Jovian planets.

The following [Moon\(s\)](#) rotate around [Jupiter](#):

- [Callisto](#)
- [Ganymede](#)
- [Io](#)
- [Europa](#)

Next suggested concept to study: [Saturn](#)

Figure 10: Consistency in presentation through Template Pages in GRAPPLE authoring

Some parts are included because of a global *layout* definition. These include the navigation (accordion) menu on the left, the header, the footer and the “page” (or the *resource* of the requested concept, to be exact). In the navigation menu the status (suitability) of links is not only indicated through the link colour but also through coloured balls that were used in other systems such as Interbook [28]. The structure of the “page” is based on a template, which is an XHTML file with some GALE tags added to it.

2.5.6.2 Domain Model authoring in GRAPPLE

To structure the above pages, the GRAPPLE authoring toolkit uses another layer of separation, by introducing domain concepts, bound together in a graph – the Domain Model. The tool for creating such structures, the Domain Model tool, is illustrated in Figure 11.

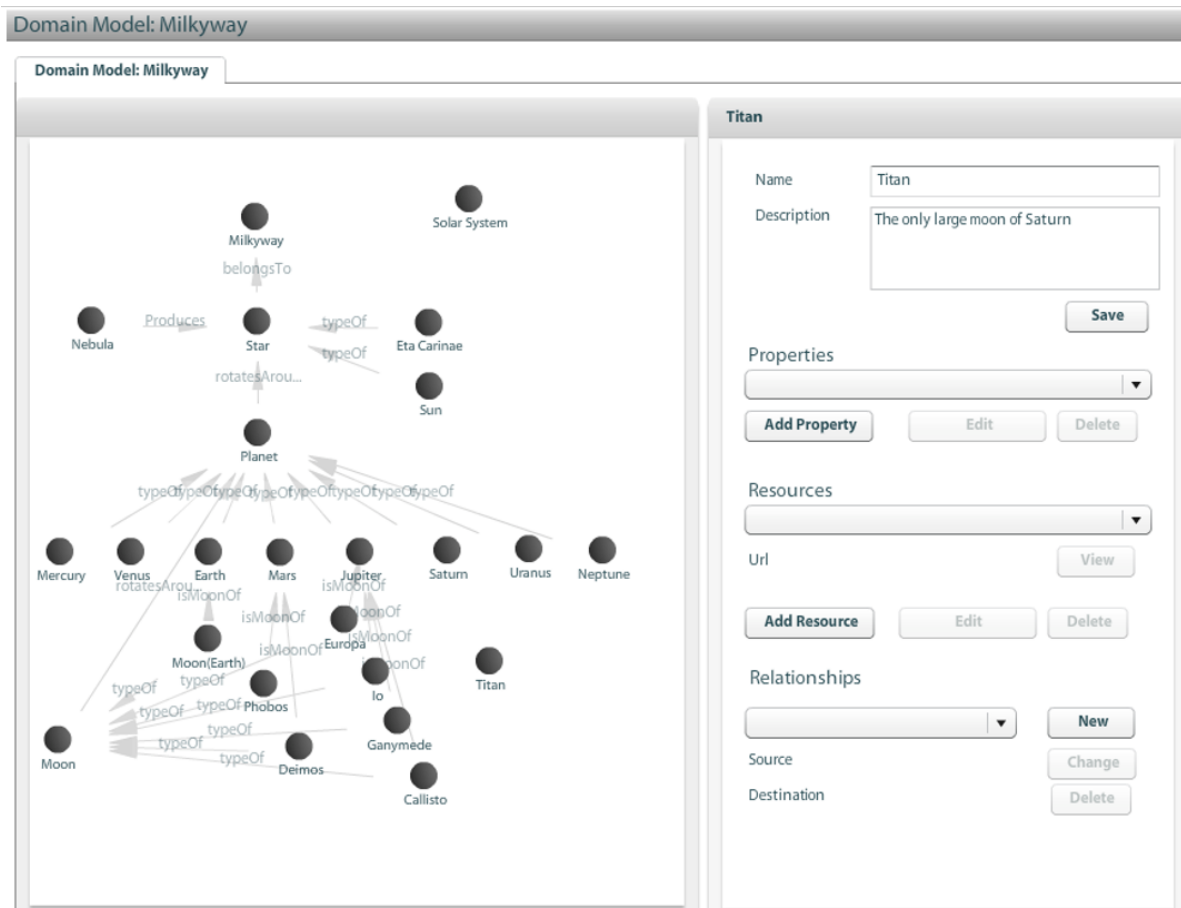


Figure 11: Creating the ‘Milky Way’ domain model with the GRAPPLE Domain Model tool

2.5.6.3 Pedagogical relationship type (PRT) authoring in GRAPPLE

The hardest part, which requires programming, is creating reusable pedagogical relationships. These represent the building blocks of the adaptive behaviour definition. The usage of programming gives the approach some flexibility. This is done via the PRT tool and is reserved to the experts. ‘Beginner’ authors do not create the PRTs, although they can use them, as can be seen in the next section 2.5.6.4. For instance (see [34]): G-Prerequisite (and G-Prerequisite-Parent) sets the suitability of a concept to true or false depending on the knowledge of its prerequisites. Actually this PRT can set the suitability for a number of

concepts at once as both the source and target sockets may contain multiple concepts. The (simplified) code is:

```
%target% { #suitability & !('${%source%#knowledge}>70') }
```

Thus, different authoring roles exist in the GRAPPLE toolkit. Also, as can be seen, the GRAPPLE authoring approach has already introduced the idea of building blocks to create larger structures from in adaptation.

2.5.6.4 Conceptual Adaptation Model (CAM) in GRAPPLE

For finalising the definition of the adaptive behaviour, a GRAPPLE author will need to compose the required adaptation by using a drag & drop method of putting together PRT pieces. Such a simple adaptive strategy is illustrated below (Figure 12), where the concept ‘Planet’ (previously defined via the DM tool) needs to be visited before the concepts ‘Neptune’, ‘Uranus’, etc. are visited; also, the concept ‘Earth’ needs visited before the concept ‘Moon’ is visited. This represents the pedagogical view on how the material needs read by learners, established by a teacher and author.

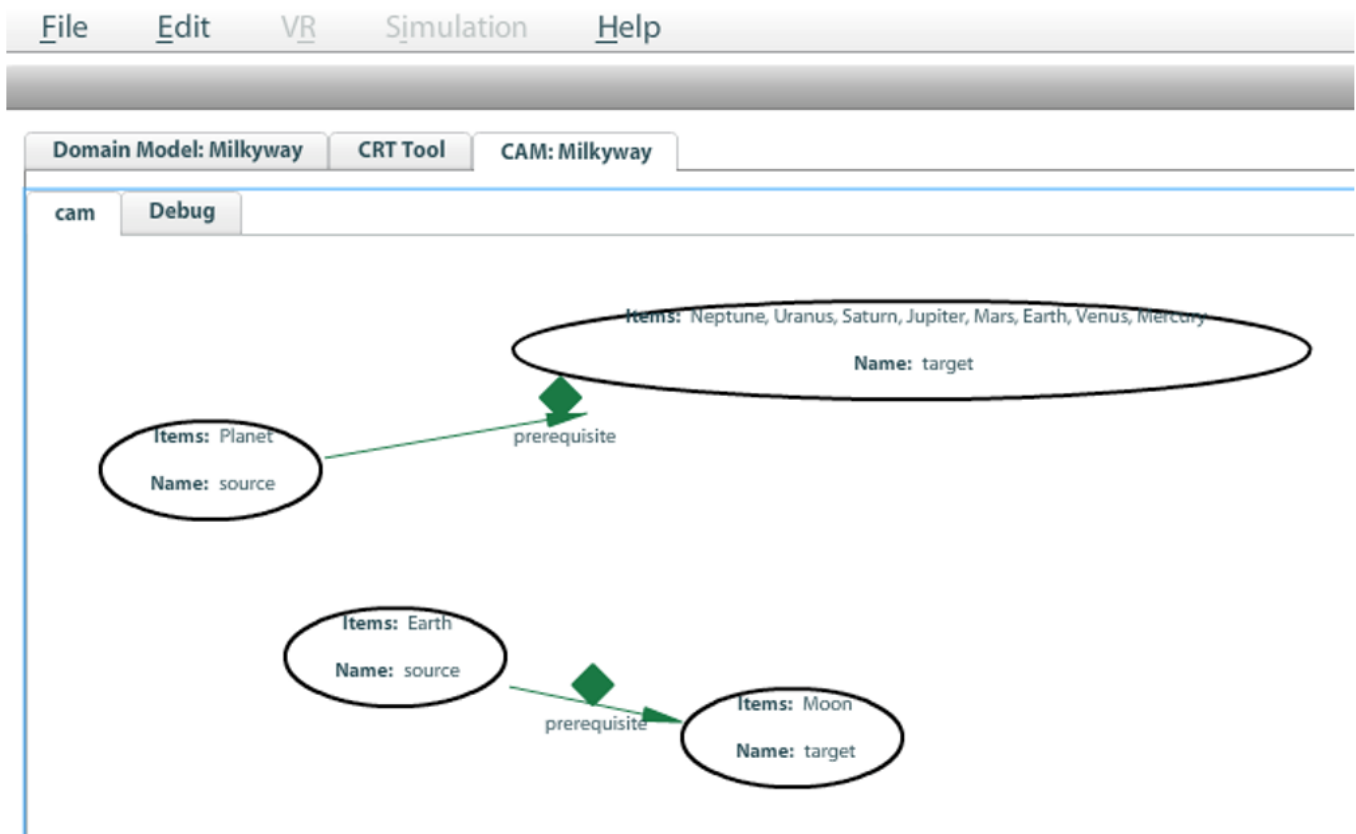


Figure 12: Adding two PRT relations in the CAM Model of the ‘Milky Way’

A more complex adaptation strategy, as created via the CAM tool, is illustrated in Figure 13. Here, the left side shows the panel of PRTs which have been already created, and can be used to compose the more complex strategies in the main window. Thus, the CAM tool combines the domain concepts created via the DM tool, and the PRTs created via the PRT tools, into a coherent whole.

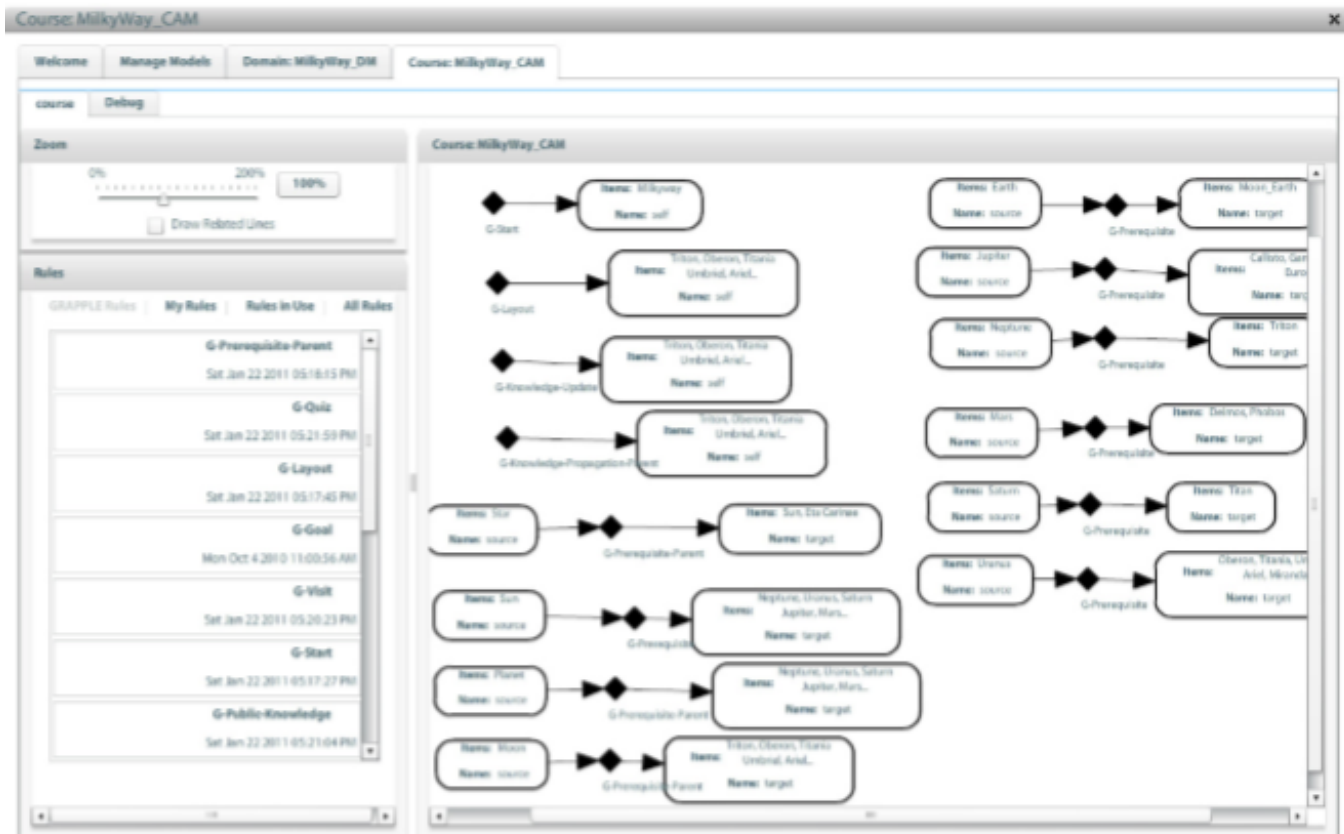


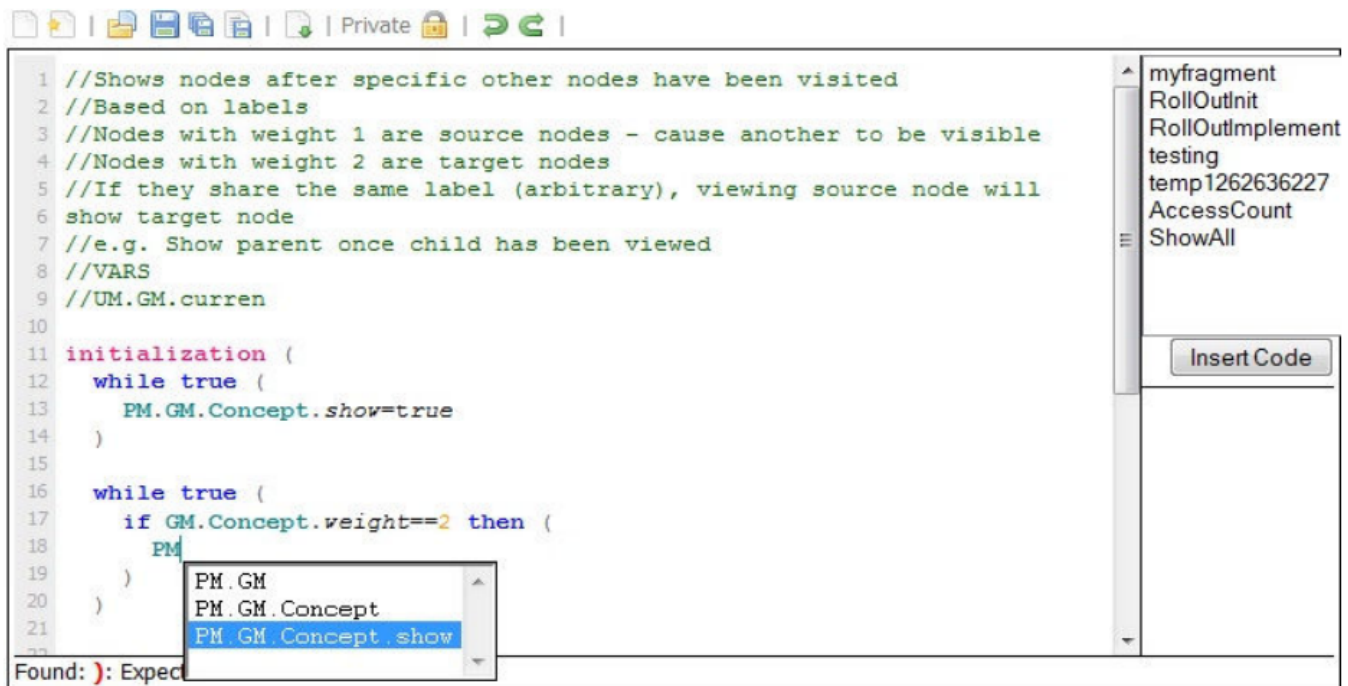
Figure 13: CAM created in the CAM Model, by dragging PRTs from the left menu

However, one of the main weaknesses of the GRAPPLE authoring approach is that the behaviour, whilst edited separately, is not reusable. This is due to the fact that the CAM strategies are already bound to certain concepts. Thus, they cannot be directly applied to different domain concepts. The only reusable pieces of adaptive behaviour are the PRT building blocks, representing very low level adaptation, normally. One could possibly ‘hide’ a whole strategy within a PRT; however, it’s usage would be rather awkward, as it would be unclear to the author how the internal working are to be (a type of ‘black-box’ approach).

2.5.7 MOT (My Online Teacher) and PEAL

2.5.7.1.1 The PEAL System

PEAL allows for text-based strategy authoring within the LAG adaptation programming language. It uses the text editor called ‘Strategy Editor’ (see Figure 14). It also allows for graphical authoring within a ‘Graphical Editor’ (see Figure 15).



```
1 //Shows nodes after specific other nodes have been visited
2 //Based on labels
3 //Nodes with weight 1 are source nodes - cause another to be visible
4 //Nodes with weight 2 are target nodes
5 //If they share the same label (arbitrary), viewing source node will
6 show target node
7 //e.g. Show parent once child has been viewed
8 //VARS
9 //UM.GM.curren
10
11 initialization (
12   while true (
13     PM.GM.Concept.show=true
14   )
15
16   while true (
17     if GM.Concept.weight==2 then (
18       PM
19     )
20   )
21 )
```

myfragment
RollOutnit
RollOutImplement
testing
temp1262636227
AccessCount
ShowAll

Insert Code

Found:): Expect

Figure 14: PEAL Adaptive Strategy Editor

The PEAL text-based editor allows advanced adaptation behaviours to be created, by using the LAG language. However, the strategy editor is aimed at programming savvy authors.

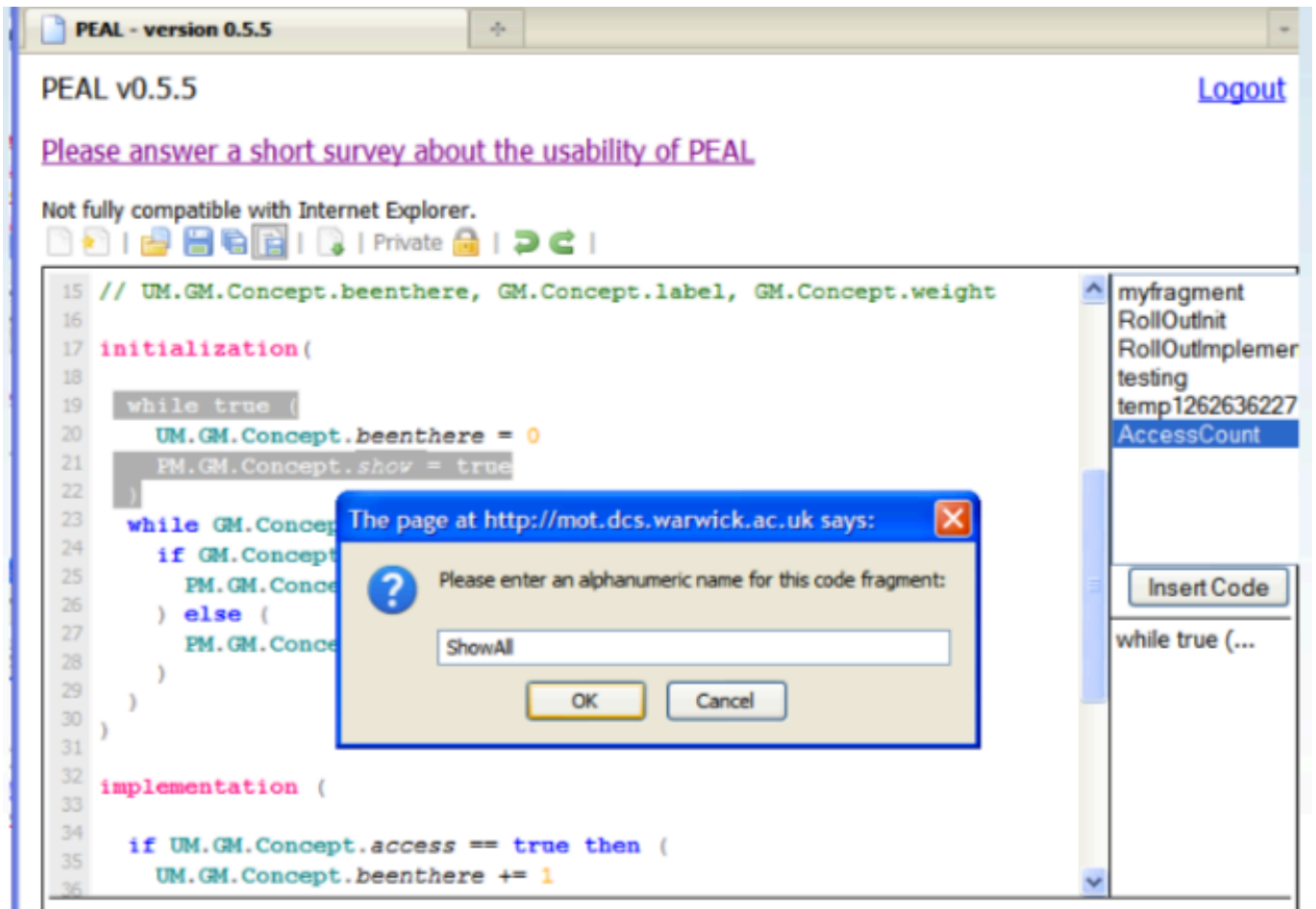


Figure 15: PEAL Graphical Editor

The PEAL Graphical Editor allows for graphical creation of strategies. This editor is aimed at non-technical authors.

2.5.7.1.2 PEAL Modular Architecture

PEAL converts the above visual representation of an adaptive strategy into LAG, by traversing the tree of visual element objects. Each visual element has a toLAG method. This method calls the toLAG function of the child element(s) in its visual element containers, thus returning its own LAG representation and that of its descendants.

The initialization consists of one assignment statement and one condition-action statement. The condition-action statement has a comparison as its condition. It also has one assignment to perform, if the condition is true and another assignment to perform if the condition is false.

The PEAL tool is designed to create directly, or convert thus, adaptation strategies into the LAG adaptation language, and separates adaptation behaviour from content, abiding by the ‘separation of concerns’. Due to the separation of concerns, based, from a theoretical point of view, on the LAOS framework, the adaptation programmer can be a different person from the strategy author.

Finally and importantly, although LAG strategies are not aimed to be written by non-technical authors, any author should be able to use adaptive strategy created by other authors, thus promote reuse.

2.5.7.1.3 Database Design

The PEAL database contains a single ‘Users’ table, where username and password are stored, used only for authentication purpose. To offer more personalised authoring environment for different authors (with different skill level and preferences), the PEAL authoring environment needs a new database to store detailed information about the authors and to be able to update this information as the author progresses with the authoring environment; for instance, a user-role and permissions associated with each author, a stencil set used or available to each author. The authoring system must keep this information updated, so the system-wide changes could be made to cater for better user support.

2.5.7.1.4 PEAL Evaluation

PEAL [17] was evaluated in the past with the help of five programming-savvy persons, as they represent the type of author at which this tool is targeted at. Non-programmers are supposed to be using ready-made, reusable strategies, without getting into the details of programming.

Amongst strengths, the following were mentioned: “element suggestion and possible variables.”; “syntax highlighting” “cultured keywords.”; “simple screen.”; “use of other strategies.”; “completion suggestions”; “code highlighting and indentation”; “save and reuse code fragments”; “highlighting and word completion”; “reusable code snippets”.

Amongst weaknesses, the following were mentioned: “Sometimes lines of code jump - automatically indented when the cursor moves.” “No search function (or search and replace)”. “If there are multiple errors only one is displayed in the status bar.” “Some display-size bugs exist with text entry box.” The

programmers were also asked to state which editor they would prefer to use to edit the LAG language, between regular text editors (used previously for editing the language) and PEAL. Without exception, they all voted for PEAL [17]. This preference for a visual language has motivated also the work in this thesis. However, PEAL had some clear limitations, as described below, which needed addressed.

2.5.7.1.5 Limitations of the PEAL editor

These initial evaluations highlighted that PEAL [17] is an environment which is useful to some degree, and represents a paradigm which is needed. However, the evaluation revealed some major limitations in the editor which were as follows.

- Parsing LAG for visual representation is not scalable, for instance, large strategies get too large and complicated to show on a single screen.
- Changing the drop-target size (and thus resize the potential parent) when an element is dragged and held over a valid parent, e.g., when an attribute is dragged over one of the operand positions of an assignment or comparison element.
- Colouring the Strategy Description tab bright red until a sufficiently verbose strategy is entered, thus enticing the author to create the description for the lay person.
- The context menu (displayed upon right-click on a visual element) should show options to insert any valid children (e.g. assignment action element into the then block of if-then), an option to delete the selected element and all its children, and a 'Help' option to display contextual documentation were among the recommendations.

Thus, PEAL as it stands might not be the best answer to describing the adaptive part in an adaptive hypermedia system. However, clearly, the visual paradigm has power, and is worth further exploration. Before new solutions are proposed, in the following, generic problems that occurred in the past with all proposed authoring systems for adaptive hypermedia are described.

2.6 Generic Problems with the Authoring in Adaptive Hypermedia Systems

The table below (Table 2) compares the most well-known authoring approaches which have been described above, based on desired features related to the current research (as per research questions; see Section 1.4). The first column checks if the particular tool expresses adaptive behaviour, as this is the main target for this research, and so on. All features are of a binary nature (Y for yes and N for no).

| System | Adaptive behaviour | Ease of use | Completeness | Correctness | Visualisation | Interoperability | Separate authoring roles |
|---|--------------------|-------------|--------------|-------------|---------------|------------------|--------------------------|
| Interbook | Y | Y | Y | Y | N | N | N |
| AHA!: AMT | Y | Y | N | N | N | N | Y |
| AHA!: Domain Model Tool | Y | Y | N | N | Y | N | Y |
| AHA!: Graph Author Tool | Y | Y | N | N | Y | N | Y |
| AHA!: Concept Editor | Y | Y | N | N | N | N | Y |
| AHA!: strategy editor | Y | N | N | N | N | N | Y |
| AHA!: form editor | Y | N | Y | Y | Y | N | Y |
| ACCT: Subject Concept Space Editor (SMCS) | N | Y | Y | Y | N | N | Y |
| ACCT: Narrative Model Builder | Y | Y | Y | Y | N | N | Y |
| APeLS | Y | N | N | N | N | N | Y |
| ACTSim | Y | Y | N | N | Y | N | N |
| GRAPPLE GAT: DM | Y | Y | Y | Y | Y | Y | Y |
| GRAPPLE GAT: PRT editor | Y | N | N | N | Y | Y | Y |
| GRAPPLE GAT: CAM editor | Y | Y | Y | Y | Y | Y | Y |
| MOT | Y | Y | N | N | Y | Y | Y |
| PEAL | Y | N | N | N | Y | Y | N |

Table 2: Comparison of well-known adaptive hypermedia authoring systems

Table 2 illustrates quite clearly that none of the previous approaches have all desired features for the

current research. Thus, the extension of an existing system, or the creating of a completely new system for adaptive authoring became necessary.

Overall, although the AH approach has been shown to be useful, an author faces a multitude of problems in order to create adaptive courseware and to reuse previously created material with different adaptation types. For example, in order to create a personalised learning experience for each user, first, the content of the lesson has to be prepared. Different alternatives of the content have to be created for different users, which leads to different paths through that content. Metadata need to be added for the labelling and annotation of the different paths. Finally, a mechanism must be defined to guide the user through the different paths. This introduces a costly and complicated authoring process [5]. So, summarising, the issues are:

- A bottleneck in adaptation is the authoring process;
- Authoring of adaptive courseware for learners with different needs is difficult;
- Adaptive Strategies are complex to create;
- Creation of adaptive courseware remains cumbersome.

Additionally [8], information interchange is very cumbersome between AEH systems. There are many systems available to authors for adaptive educational material but there is no de-facto standard for authoring adaptive content efficiently which could be used across the board to enhance reusability and compatibility between different systems. To add to the problem, authoring remains a daunting task for the ever-busy author, to create adaptive courses to more and more diverse, in most cases, global student audiences with diverse needs.

Concluding, it is imperative to find a better authoring system, which can help authors in course creation with minimal effort, towards maximum effect. In order to do this, we need to ask some fundamental questions regarding the authoring process and the tools. Similarly, adaptive systems might need to be able to export to various adaptation languages, in order to be compatible with a wide variety of applications [8].

The following, summarises the main imperatives that result from issues with authoring, inspired by the needs of teachers and authors, and not students.

2.7 Authoring Imperatives

Authors may be unfamiliar with the complexity of the authoring process. It is important that any authoring system caters to the author's needs. A study [37] on authoring tools proposed a set of generic authoring imperatives to make authoring easier; these include *Complexity Imperatives* and *Support Imperatives*, as follows below.

Complexity Imperative:

- *Separation of Concerns*
- *Use of frameworks and Use of standards*

Support Imperative:

- *Adaptive Functionality*
- *Simple Access to Content*
- *Shallow Learning Curve and Familiarity*

The Complexity Imperatives address the issue of reuse. The separation of different tasks into different tools is known as the 'separation of concerns' [13] and is useful to promote the reuse of static and dynamic materials, separately. In its simplest form, this principle states that adaptation behaviour and the content of a course should be authored separately. Besides the obvious implications of reuse, this separation allows the two parts to be authored by different authoring roles [23]. This is vital if the aim is to not only simplify the authoring process, by displaying a simpler view of the authoring system to the author with less experience with the system, but also to spread the authoring load and encourage reuse of adaptive strategies.

To apply the authoring imperative, separation of concern must be included in the design, which will allow different authors to work on different parts of the course.

Furthermore, common standards should be followed wherever it is possible to do so. For instance, if a new standard is released by W3C organisation, then it should be utilised by the authoring tool. This will mean that any device which can read the standard will automatically be compatible with the tool which implements the standards. Alternatively, if standards are lacking, frameworks should be used, such as AHAM [13] and LAOS [11].

The support imperatives address the help and support offered to the author: how easy it is for the author to learn to author with the system (with the aim being a shallow learning curve and familiarity), how easy or difficult it is to access content, and if the functionality truly is adaptive to the author.

These imperatives in section 2.7, set out a number of important issues to consider when designing new authoring systems, including for this research.

The additional, important issue of interoperability are visited next.

2.8 Interoperability between Adaptive Educational Hypermedia Systems

Information interchange is very cumbersome between AEH systems. There are many systems available to author adaptive educational material, but there is no single set (*de-facto*) standard for authoring adaptive content efficiently, which could be used across many adaptive educational systems, to enhance reusability and compatibility. To have a common standard for all adaptive systems might seem ideal, a feasible solution might be to mirror the standards used elsewhere: for instance, learning management systems (LMS) export to and import from various e-learning standards (IMS- QTI, LOM, IMS-CP, etc.). Similarly, adaptive systems might need to be able to export various adaptation languages, in order to be compatible with a wide variety of applications [11].

From the research [38] [19], it is evident that, to capture adaptation and to use it in a different AEH system is very challenging, due to the lack of de-facto standard between different AEH system. In [9], the authors

considered supporting multiple adaptation language outputs as a desirable feature, besides developing new languages targeted at specific levels of access (transformed into wrapping levels).

The aim in this thesis was to develop a methodology which will enable adaptive contents to be moved and reused in another adaptive educational hypermedia system, according to the features offered by the target system. Interoperability, in the sense of plug&play functionality available for any reasonably conceivable environment, was one of the main reasons VASE 2.0 was created from scratch, as a complete re-write of VASE 1.0, by using a framework which offered interoperability features. Interoperability of the authoring system is discussed in Chapter 2 , Section 2.9.1.3 and Chapter 5, Section 5.4.3 and evaluated in Chapter 6.

After understanding the issues with adaptive hypermedia, with specific focus on its authoring, and, based on the objectives, a simple solution (sustained also by the success of PEAL, discussed in section 2.5.7.1.4) is to apply visual programming. The current state of the art in this area is analysed next.

2.9 Visual Programming Systems

Visual programming languages have a unique ability to make programming a more intuitive experience. The need to have knowledge of the programming language syntax is not necessary. Thus, the programming focus can be directed towards solving the actual problem, rather than trying to recall the programming language syntax. Many graphical programming systems exist, attempting to break down the barriers to learn computer programming. Instead of working only with text and recalling language syntax and rules, many graphical programming systems allow users to manipulate and interact with visual objects, to build their programs.

Block-based programming offers visual units of work, called Blocks. Blocks can be dragged out onto a workspace, where they can be arranged and connected together, to build a program. Block-based programming tools are substantially more learnable than text-based programming languages for many reasons, including the following.

- **Forgetting** - Users do not need to memorize programming syntax. All programming constructs are accessible visually in the Graphical User Interface (GUI).
- **Feedback** - Graphical constraints and feedback can be used to prevent users from making syntactical errors, rather than simply reporting the syntactical error.
- **Real-world Metaphor** - Blocks offer stronger real-world metaphors than text. For example, blocks look like puzzle pieces, which allow users to understand which blocks can and cannot fit together, just by looking at the block connector shape.

2.9.1 OpenBlocks Design

OpenBlocks [39] is an open-source Java library for creating user-interface (UI) elements for blocks-based programs. It only requires users to connect puzzle-piece-like objects, called blocks, of varying shapes and colours, to build their program. The Block shapes not only look like puzzle pieces, but also are connected like puzzle pieces, by matching the blocks at their connector shapes.

2.9.1.1 Block Manipulation

These pieces can be dragged and joined with a mouse, with very little knowledge about the specification language. The OpenBlocks framework improves visual feedback via the shape of a construct gives clues on how to use the construct. For instance, if a construct will look like in Figure 16, it is obvious to the user that the construct will only fit a type which is compatible with its connector. Blocks are manipulated directly by the user, via mouse-click and drag-n-drop actions.



Figure 16: CodeBlocks - IF Block

2.9.1.2 User Interface of OpenBlocks

The OpenBlocks interface includes many of the basic features of a block programming environment, such as graphical blocks, block drawers that contain these blocks and a canvas, where block programs are built.

2.9.1.3 *OpenBlocks Summary*

The main features of OpenBlocks are:

1. Essential programming environment features are provided
2. Multiple ways to configure specific UI elements
3. Customisable programming environment
4. Easy to extend for programmers
5. Easy to understand for authors

While OpenBlocks allows different shapes to be connected like puzzle pieces easily, it lacks other functional features, such as interoperability, robust type checking, and specification validation, amongst others. The solution to these problems was a new framework developed by Google developers, called Blockly (see next section). This block programming framework offered all of the features of OpenBlocks and some new ones. It works in any browser mobile or desktop without any additional plugin, making it fully interoperable between different browsers on different operating systems. This is especially important if a tablet device is used to quickly create an adaptive specification.

2.9.2 *Blockly*

Blockly [40] (Figure 17) has the ability to prevent incorrect combinations from being constructed, addressing correctness issues within programming. Each of Blockly's connection types (value inputs/outputs, next/previous statements) can be labelled with type information, so that invalid connections will refuse to connect. This provides authors with instantaneous feedback and avoids many simple mistakes, hence improving correctness of adaptive specification during the authoring phase.



Figure 17: Blockly Block

The Blockly framework offers auto-complete, type-cast protection and many other useful features such as copy, paste, completeness and correctness.

This framework is an inspiration for the work in the current thesis, and the design of the VASE framework and visual authoring program (see Chapter 5).

2.10 Conclusion

In this chapter, we have discussed languages, models and frameworks for Adaptive Hypermedia and analysed the work done in the area of Adaptive Hypermedia System design. We have presented a wide variety of authoring tools in Adaptive Educational Hypermedia System. Although most tools are designed to make the author's job easier, they are usually still convoluted to master, and most authors struggle to reap the benefits of the features offered by the systems.

In addition, we have also presented some of the frameworks which were used during the development of the new authoring system. The main reason was to introduce the reader to the frameworks used during the implementation of the new authoring tool.

In the following chapter, we will detail the methodology used throughout this thesis.

3 Methodology

One of the fundamental parts of any research is to choose the appropriate methodology. A methodology is described as guidance in solving a particular problem using suitable tasks, tools and techniques [41]. The research in this thesis includes a range of methodological approaches, employing both quantitative and qualitative data collection and analysis. In this chapter, the methodological approaches in each section are outlined.

Outcome of this chapter: This chapter provides general guidance for the whole research approach adopted with this thesis. It explores the approaches, techniques and methods used during the research, and therefore gives useful suggestions to other researchers who would like to follow-up a similar type of research. Furthermore, it discusses the quantitative and qualitative analysis methods that are used in the analysis of the result generated from the experiments. In order to create a coherent understanding of the methodology and its application within this research, it has been grouped as follows. Firstly, the next subsection deals with the methodology employed in the literature review. Next, the user centric approach taken throughout the work in this thesis is briefly explained. Then, the implementation methodology is sketched. Finally, the experimental methodology is detailed, for all 4 experiments performed for this thesis.

3.1 Literature Review and Background Review

The literature review is very essential in any research study; it serves as guidance on what has been researched in the area, it is very useful in identifying gaps in the area and serves as the starting point in any research. Work on the literature review is reflected in Chapter 2. It started in 2010 and has continued until the date of the thesis submission, to ensure that recent work is also analysed and included. This includes an initial identification of major fields related to the area, and their exploration, based on the state of the art at the time (e.g., via Google Scholar, paying attention to the best venues in the field – e.g., based on the acceptance rate and Core A rating for conferences, as well as based on, additionally, impact factor, for journals). Research gaps in adaptive authoring, usability and user modelling were identified in the

literature in this way – especially the gap regarding the lack of easy to use authoring tools for adaptive hypermedia and e-learning.

3.1.1 Adaptation Language Research

One of the areas that became prominent relatively rapidly, both in terms of research needed of the literature, as well as of systems, frameworks, standards, languages, etc. was that of visual programming. Thus, visual programming frameworks were primary sources of research, as they are used in other areas for their *usability* [39], and they inspired here the new proposal of a visual framework for adaptive hypermedia authoring. The visual framework delivered by this thesis needed to be easy to use, as it was evident in the earlier authoring system evaluations which used LAG as the adaptation language, that high-level programming was not fulfilling this goal. As a result of this, in the new framework, each LAG construct was represented in a visual form, to provide affordance to the author. Additionally, extensions to the LAG language were also suggested, to improve the LAG language features; these have been presented in A User Centred Approach.

3.2 User Centred Design and Implementation

Moreover, as the goal of this research is to simplify and improve adaptive specification authoring in adaptive hypermedia (see also the research questions, Section 1.4), this applies to target users. As these end-users are the focus of this research, the user-centred design (UCD) methodology is used throughout the research in design specific experiments, to extract their needs, requirements and constraints [42]. This user-centred design methodology has been used as a research methodology in many researches involving application oriented research in general, and more significantly, users (e.g. [43], [44] and [45]). User centred design implies that users, usability, and user characteristics are taken into account throughout the processing.

The user centred (UC) approach outlines the phases throughout a design and development life-cycle while focusing on gaining a deep understanding of who will be using the product. There are multiple principles that underlie the user centred approach. Such design is based upon a clear understanding of users, tasks

and environments. UC is driven and refined by user-centred evaluation; and addresses the whole user experience. The process involves users throughout the design and development process, iteratively, as illustrated in Figure 18.

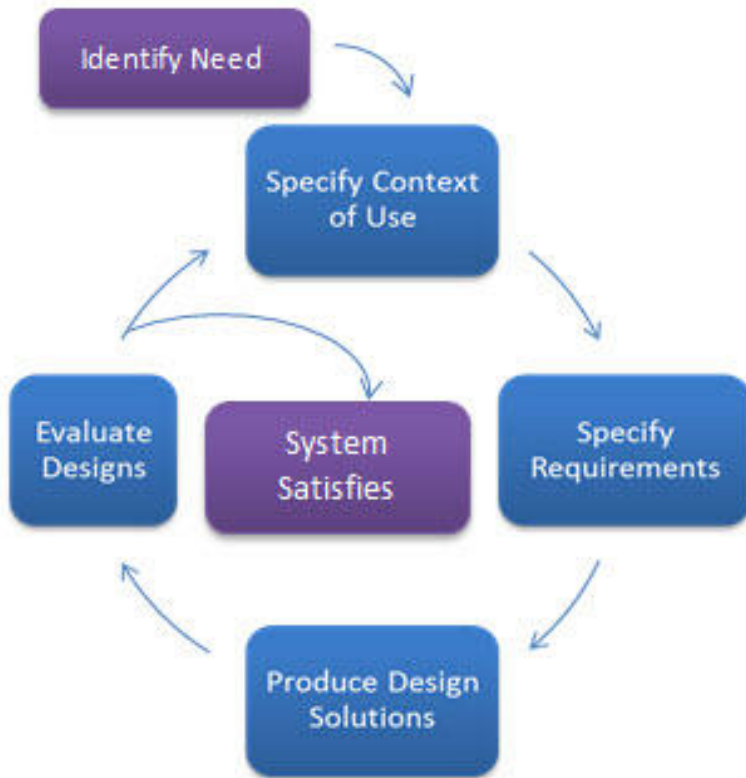


Figure 18: Phases of user centred process (source www.usability.gov [46])

The following are the general phases of the user-centred process [46]:

- **Specify the context of use:** Identify the people who will use the product, what they will use it for, and under what conditions they will use it;
- **Specify requirements:** Identify any business requirements or user goals that must be met for the product to be successful;
- **Create design solutions:** This part of the process may be done in stages, building from a rough concept to a complete design;
- **Evaluate designs: Evaluation** - ideally through usability testing with actual users - is as integral as quality testing is to good software development.

The user-centred approach has been used at every stage of this research, starting from the requirement analysis to the implementation and the evaluations. The next sections provide details on where and how this approach was used in this research.

3.3 Implementation Methodology: The Visual Platform

In order to evaluate the proposed solutions to the research questions, a platform on which the evaluations could be performed was needed, to be either selected from existing platforms, or newly developed from scratch. The software was to be designed on a modular basis; each module extends the system's functionality. This enables the system to be extended without having to rewrite the entire system from scratch. In addition, this will allow a more personalized way of displaying a set of modules suited for a particular user. This platform would then be extended as the research continued, in order to evaluate the extensions and tools that were created during the research. It was also expected that lessons learnt during the development of the platform would be also useful in answering our first sub-research question.

This process comprised the following steps:

1. Capturing System Requirements: Initially, these requirements were gathered from the literature review and from previous evaluations of existing systems;
2. Implementation of the requirements;
3. Testing and debugging;
4. System Evaluations;
5. Further implementations: Feedback was then incorporated into the system, to improve authoring in adaptive hypermedia.

The implemented system served as an evaluation tool for the user modelling, usability and functionality features. The system implementation was an iterative process, each system prototype representing an enhancement of the previous version, addressing the limitations and enhancing the system's features. For this research, the authoring system developed was called VASE, short for Visual Adaptive Strategy Environment. Further details on the system and the implementation can be found in Chapter 5.

3.3.1 Iterative System Implementation based on User Centred Implementation

3.3.1.1 PEAL Extension

PEAL [37] was evaluated with the help of programming-savvy persons, as they represent the type of author at which this tool is targeted at. This highlighted PEAL's limitations, and suggested improvements on how PEAL can be updated.

Thus, the initial thought was that to improve PEAL and to address the shortcomings reported in the PEAL evaluation. However, from the literature review on authoring tools, it was clear that by simply improving PEAL, this was not going to solve the main problems, since most of the problems reported were platform- or framework-related (see Chapter 2, Section 2.5.7.1.5). The platform and framework thus needed to be changed, to meet the requirements of adaptive authoring system specified in Table 3.

In addition, several visual programming frameworks were investigated; some of the major frameworks have been listed in Chapter 2. Few of the frameworks were prototyped, in order to select the best suited framework to design and develop the new authoring system.

3.3.1.2 Other Frameworks

Initially, two visual programming frameworks were prototyped: the GLG framework [47] was more suited than Oryx framework [48]. However, the GLG framework licensing cost was too expensive for an academic setting and it was based on aging technology. As a result, these prototypes were not taken any further. PEAL had to be redesigned to adhere to the requirements and desired features reported in the literature and reported by the user in this thesis in Chapter 2, Section 2.5.7.1.5.

Block programming frameworks offered many benefits, most of which have been listed in Chapter 2, section 2.9. As a result, block programming frameworks were prototyped next. To receive user feedback on the visual programming framework in comparison to text-based programming techniques, prior to any implementation, a short user questionnaire was conducted on SurveyMonkey (please see APPENDIX D: Novice Group Evaluation for details, as well as Chapter 6, Section 6.3). The results show (see Chapter 6, Section 6.3) that the majority of the users preferred the visual technique and found the flow of the program

easy to understand in block programming, in comparison to the text-based programming techniques. From the survey result and speaking to the users throughout this research, it was clear that block programming frameworks were best-suited for novice users.

3.3.1.3 VASE 1.0

Thus, VASE 1.0 was the first attempt to create an authoring system using the block programming framework (written in Java), called OpenBlocks [39]. Due to the features offered by the OpenBlocks [39] framework at the time of creation, this framework could only be used to create a desktop application, which required JVM (the Java Virtual Machine) and JDK. In addition, a desktop application had to be installed, with alleviated (admin) user rights, on the computer.

VASE 1.0 was created using OpenBlocks [39] (described in Chapter 2, section 2.9.1), a block programming framework written in the JAVA programming language. The LAG language grammar (APPENDIX A: LAG Grammar) was used to create blocks based on LAG constructs, called LAGBlocks.

However, some very basic functionality was missing from the framework, like *copy* and *paste* blocks or *copy* and *paste* entire section of blocks, complete list of the missing and desired features, highlighted by this research, as have been listed in Chapter 5, 6.5.3.

In addition, VASE 1.0 required an installation process and could not be used in the browser without installing a Java plugin. A new framework was thus created, to address this shortcoming, which didn't need a plugin to be installed and would work in any browser, desktop or mobile because it was based on the HTML 5 specification [49].

This meant that if VASE can be written in this framework, it would solve the majority of the problems reported by the users. A complete re-write was thus required for VASE 2.0, using web technology stack, and based on user feedback, as detailed in Chapter 5, section 5.5.

Thus, the next development iteration was to create VASE 2.0. Apart from name resemblance, VASE 2.0 is a complete re-write of VASE 1.0 in a completely different programming language and on a different

platform. Based on JavaScript, VASE 2.0 followed the HTML5 specification [49] and will work in any browser, since JavaScript can run in any mobile or desktop browser.

To rewrite VASE in a different programming language, using a different framework, and on a different platform, required a substantial additional development effort. The HTML5 specification was fundamental to this, as it offered standards consistent between browsers [49]. The final design and implementation of the adaptive authoring system have been listed in Chapter 5, Section 5.5. Next, we visit the experimental methodology applied to validate the various results of this thesis.

3.4 Experimental Methodology

3.4.1 User-centred evaluation through experiments and interviews

Several experiments were conducted to collect user feedback of the implemented system. Users were asked to use the system, as they were presented with the new authoring system. To store this interaction, a web service was created, to collect data about the users' interaction in the database. At the end of each experiment, users were asked to fill-in a questionnaire, to provide feedback on the authoring system. In addition, qualitative feedback was collected from the users who were interested in providing additional feedback. More information on the experiments and interviews can be found in Chapter 6.

This research employed different methodological approaches for each experiment, which are described briefly in the following.

3.4.2 Non-professional programmer evaluation

The first experiment was conducted to *evaluate the chosen visual programming framework*. This experiment was focused towards general usability issues and retrieving users' preference on writing code, versus using visual programming constructs. This was aimed at general non-professional programmers, without extensive programming knowledge or experience. The users expressed their opinions about the *snap-in* programming techniques (refers to an object that can be attached to another object and that will

then function as part of the whole). It was aimed at the general users, to find out how users felt about snap-in programming techniques, in comparison to writing programming syntax. This was to ensure that the chosen visual programming framework will not leave users intimidated or confused. This experiment was done before the system was designed and developed, using visual programming techniques. The feedback received from this experiment was then used to first select the appropriate visual programming implementation framework, secondly, to design and develop the first prototype of the adaptive authoring system.

3.4.3 Amazon Mechanical Turk experiment

This experiment was conducted on Amazon Mechanical Turk (AMT). The purpose of this experiment was *to compare the new visual system with the previous authoring methodology* (for an authoring system of similar power of expression). The time allocated for each human intelligence task (HIT) was limited to 1 hour. The worker requirements were set to be Master (Amazon Mechanical Turk Masters) to do the specified HIT. Amazon Mechanical Turk has built-in technology which analyses the worker performance, identifies high performing workers, and monitors their performance over time. Workers who have demonstrated excellence across a wide range of HITs are awarded the Masters Qualification. Masters must continue to pass Amazon's statistical monitoring to retain the Mechanical Turk Masters Qualifications.

The questionnaire consisted of set of questions to compare the two systems and select their preferences on the following axes: *Overall quicker to learn, usability, completeness, correctness and higher user-satisfaction*. A detailed analysis of the questions presented in the questionnaire has been presented in Section 6.6.

Users were involved in adaptive strategy creation, editing and reuse, in both the PEAL [17] (the most supportive environment for generic adaptation behaviour description available at the time of the work on the thesis, as explained in the previous chapter) and the VASE authoring systems. Finally, users expressed their perceptions about both systems and evaluated them, by answering a questionnaire, to evaluate the

different features of the systems. Details on the discussion of the evaluation have been presented in Chapters 6 and 7.

3.4.4 University lecturers with eLearning experience

The purpose was an *evaluation of the new visual system with teaching and course creation experts*. This was a face-to-face session and produced not only qualitative data, it was aimed to get an in depth, detailed evaluation of the system, via observations and semi-structured interviews, to discover new themes in the research and also to receive first-hand feedback from users who have been involved in course creation at UK universities. The interviews were conducted at convenient locations for the participants. Thus, most of the interviews were conducted on the university campus, with the exception of one, which was conducted at home, for the participant's convenience. At the end of each session, the SUS [50] test was given to participants, to record their usability scores.

This group included lecturers from UK universities, with teaching experience ranging between 6 years to 20 years of teaching experience at the university level. The aim was thus to capture qualitative feedback on the new authoring tool from users with teaching experience and who have been involved in course creation, to ensure a high quality of feedback, as well as potentially to get the feedback from potential users of the authoring system in higher education.

Firstly, users were given a quick introduction on Adaptive Hypermedia, to familiarise themselves with the terminology - for instance, Domain Model, Presentation Model, Presentation Model and Adaptation Model. At this stage, users were asked to use both systems, to create 3 adaptive courses in each authoring system, and it was left up to them to decide in which order they would use the two systems. No information about which of the two systems was the newly developed one was given, in other words, no information was given about the relation of the author to any of the systems, to avoid any preferential bias.

The final stage was to interview the university lecturers on their perceptions, in terms of their perceived usability and functionality for both systems. Details of the experiment and the related discussion can be found in Chapter 6, Section 6.5.

3.4.5 LAG author group evaluation

The purpose of this experiment was to elicit *feedback from adaptive authoring experts about the new visual system*. This experiment thus covered the missing aspect in the previous one, e.g., an experiment involving users with teaching and programming experience, who have used the LAG programming language to write adaptive strategies. This group had a good understanding on the LAG adaptive programming language and could reveal fundamental flaws with the authoring system, from an implementation perspective. This group also has a good understanding of other authoring systems in the Adaptive Hypermedia domain. Due to the strict requirements for this group - e.g., having a LAG programming language experience - it was not possible to use a large number of users, as there are very few individuals with LAG expert knowledge. Instead, sessions were arranged to retrieve qualitative feedback on the new authoring system and whether the authoring system would be useful for authors in adaptive strategy creation and reuse. Notes were taken during these sessions; details of the notes are presented in Chapter 6, section 6.4.

3.4.6 Sample Size and Limitations

One of the important features of an experiment is the sample size, and it can impact the significance of different evaluation measures. In experiments, one of the key issues is to determine the intended sample size to be studied. The question that is usually asked is “what number is reflective for the actual population?”. This question cannot be answered by a number alone. The common factors include the aim of the study, the population size and the sampling error [51]. Formally, factors to determine the sample size should include the confidence level, precision level and the degree of variability attributes. The precision level is described as “the range in which the true value of the population is estimated to be. This range is often expressed in percentage, (e.g., ± 5 Percent)”, the confidence is described as “the average value of the attribute obtained by those samples”, “equal to the true population value” and the degree of variability in the attributes being measured is described as “the distribution of attributes in the population” [51].

After carefully considering these aspects and limitation, different sizes of user populations were involved in different evaluations. For qualitative interviews, smaller-scale populations were used, in order to keep

the process practical. The main aim of the interviews was to get qualitative feedback from the users, by observation and semi-structured interview questions. Additionally, interviews with experts are usually small-scale in any research area.

For a large-scale evaluation of the final system, the sample size was calculated based on the formula as in [51].

This formula was used to calculate the sample sizes is shown below. A 95% confidence level and $P = .5$ are assumed.

$$\text{Sample Size} = \frac{(Z)^2 * P * (1-P)}{(C)^2} \quad SS = \frac{(1.96)^2 * 0.5 * (1-0.5)}{(0.05)^2} \quad SS = \frac{0.9604}{0.0025} \quad SS = 384$$

Where:

Z = Z value (e.g. 1.96 for 95% confidence level)

p = percentage picking a choice, expressed as decimal (0.5 used for sample size needed)

c = confidence interval, expressed as decimal (e.g., .05 = ± 5)

The population that was used to base the calculation on was the number of academic staff (full-time and part-time) in UK higher education, which was 198,335 in 2015, which represents 49.1% of the population [52]. These statistics were released by the Higher Education Statistics Agency for the year 2014/15 [52]. This number represents all academic staff which includes all professions, for instance, managers, directors, professional occupations, associate and technical occupations and clerical occupations. Although, as the focus of this research is not on all academic staff, the actual number of academic staff would be lower than this number. Thus, using this number is an overestimation, likely to give us a somewhat overestimated sample size. The sample size calculation was based on a 95% confidence level, a 5 confidence interval, and the resulting recommended sample size was 383.

To consider all the teachers in the whole world, the recommendation sample size is 384, so the recommendation was to collect 384 users for a large-scale evaluation.

The number of users achieved was of 427, with 47 participants taking part in the first evaluation and another 380 in the follow-up experiment in Chapter 6, section 6.6.

This number is considered a sound number for a research-based evaluation, as other case studies report on a close or a lower number of participants, in similar cases. For example, a case study investigating the user satisfaction on adaptive e-advertisement system quality, reported 267 users as the sample size [53]. A number of approaches can be used in selecting the sample size, one of the approaches is to choose a sample size which has been used in similar research; the sample size used in similar research [15], [53] ranged from 80 to 267.

From this perspective, the work return in this thesis was much higher. This thesis not only conducted an evaluation on a larger number of users, but also conducted qualitative interviews, to analyse and identify issues which may not have been possible to discover only by quantitative measures. As stated earlier, the quantitative data collection was used to gain knowledge on the overall impact of the research, while qualitative data collection was used to identify new themes in the research area.

3.4.7 Qualitative and Quantitative Analysis of Results

As described in section 3.4, this research conducted four different evaluations. In each evaluation, results have been produced. These results were either quantitative or qualitative. To provide better understanding of how these results have been compiled and analysed, the statistical tests and analyses are briefly explained.

Descriptive statistics were conducted to analyse quantitative responses. In addition, open questions were asked, and semi-structured interviews were conducted. These inclined towards discussing the advantages and disadvantages of authoring systems and what can be suggested in terms of features, preferences, limitations and usability.

The outcomes of the second and third evaluation were mainly qualitative, in order to analyse the quantitative feedback collected previously, in more depth. The qualitative content analysis method is defined as a “subjective interpretation of the content of text data through the systematic classification

process of coding and identifying themes or patterns” [54]. The approach includes three different types of analysis, including directed, conventional and summative. The summative approach was used, since it emphasises finding keywords that hold the main answers for the open questions asked. These keywords were collected from the data generated by the users and were interpreted within the underlying context [54]. Details on the use of this method are found in Chapter 6 Section 6.4 and Section 6.5.

For the numerical results, *questionnaires* were used. The questionnaires asked the participants how confident they felt with using the authoring tools. Each question used a Likert scale [55] with the options: 1 = strongly disagree, 2 = disagree, 3 = Neither agree nor disagree, 4 = agree and 5 = strongly agree.

In order to analyse the results, statistical tests were conducted, as described below.

Descriptive statistics: these tests included all the descriptive ones of Mean, Mode, Standard deviations that address to the central tendency theorem. These tests were conducted mainly in Chapters 5 and 6. The mean values and percentages were used in Chapter 6.

- *The mean:* In statistics, mean and expected value are used correspondingly, to represent one measure of the central tendency either of a probability distribution or of the random variable characterised by that distribution [56].
- *The mode:* The mode is the value that appears most often in a set of data. The mode of a discrete probability distribution is the value x at which its probability mass function takes its maximum value. In other words, it is the value that is most likely to be sampled [56].
- *The standard deviation:* is a measure that is used to measure the amount of difference of a set of data values. A standard deviation close to 0 indicates that the data points tend to be very close to the mean (also called the expected value) of the set, while a high standard deviation indicates that the data points are spread out over a wider range of values [56].

- *Pearson-Chi squared test*: is used to assess three types of comparison e.g. *goodness of fit*, *homogeneity*, and *independence*. A test of *goodness of fit* establishes whether an observed frequency distribution differs from a theoretical distribution. A test of *homogeneity* compares the distribution of counts for two or more groups using the same categorical variable. A test of *independence* assesses whether unpaired observations on two variables, expressed in a contingency table, are independent of each other. In this research Pearson's chi-squared test is used, amongst others, to establish if a set of data is normally distributed. The data of the second experiment was analysed with this test, to establish if the T-test can be applied, this test was conducted in Chapter 6, section 6.6.1.

- *paired T-test*: a T-test measures if the population mean differs for two related samples, given that the samples have both a normal distribution (as tested with the Pearson-Chi squared test above). The t-test is one of the most popular tests for comparing two samples. The t-test is used here to determine if two sets of data are significantly different from each other. This test was conducted in Chapter 6, section 6.6.1.

- *Wilcoxon signed-rank test*: is a non-parametric statistical hypothesis test used when comparing *two related samples*, or repeated measurements on a single sample, to assess whether their population mean ranks differ (i.e., it is a paired difference test). It can be used as a replacement to the paired student's T-test, T-test for matched pairs, or the T-test for dependent samples, when the population cannot be assumed to be normally distributed. Based on the data collected from the descriptive statistics, the results in this research are not normally distributed. So the need for a non-parametric test to examine the significance against the median value of 3 has emerged [56]. This test is conducted to support the results achieved earlier in the descriptive statistics, to prove the results are on the positive side of the argument. This test was conducted in Chapter 6, section 6.6.1.

- *Bonferroni corrections*: this is an adjustment made to probability (p) values, when several dependent or independent statistical tests are being performed simultaneously on a single data set. To perform a Bonferroni correction, one needs to divide the critical p value by the number of comparisons being made.

Bonferroni corrections were applied, due to the fact that I have computed significance for comparisons of several questions, as well as several categories of questions (see Chapter 6, section 6.6.1).

3.5 Summary and discussion

This chapter is focused towards exploring suitable methodological approaches that have been used within this research. The methodology is one of the fundamental parts of any research, as it draws the path for the research to follow.

For the purpose of this research, a range of methodological approaches employing both qualitative and quantitative methodologies were used. This chapter also discussed the implementation and evaluation measures employed in this research, and reflected upon the overall research, serving as an outline of the design and validation of the proposed research.

In the following, the design process of an improved authoring tool is described, in the form of a theoretical framework, based on extracted needs, written in the form of usage scenarios, aimed at the different type of authors which are expected in such an environment.

4 Theoretical Framework: Author Roles & Functionality Matrix for Adaptive Authoring Tools

4.1 Introduction

This chapter aims at defining the concept of an adaptive authoring system and partially addressing the following research objective.

Objective 1 - *Conduct an extensive theoretical background research into the area, to find the problems faced by the authors in adaptive strategy creation and to propose a theoretical framework to improve adaptive authoring for authors with different needs.*

Outcomes of this chapter: This chapter address the second part of the objectives “*to propose a theoretical framework to improve adaptive authoring for authors with different needs*”.

This chapter describes a theoretical framework to show different system features depending upon the author knowledge and experience, to help the author view only the relevant options for them, thus avoiding confusion.

Adaptive Authoring systems in prior research (AHA! [16] , WHURLE [57] and PEAL [17]) may be authoring systems which are well designed, but the framework they are based on does not support a full-blown implementation of the separation of concern from an author’s point of view. Existing research has shown that separation of concern is very important, to simplify authoring and to improve the reusability of adaptation specification and the content created [15].

Separation of concern (see Objective 3, Section 1.5) states that different authors should be able to work on different parts of a course. For this purpose, different authoring roles need to be clearly identified. Thus, here we propose an author role functionality matrix, which should enable better control on system features mapped to author-roles. Details of the functionality matrix idea are explained in the next section.

4.2 Author Role Functionality Matrix

Our approach to *author roles* considers a system that allows user recognition. This way the system can identify an author and apply necessary adaptations relevant to the user's specific role., i.e., show a customised view of the system. New authors will be given beginner user roles to start with, which are gradually upgraded, as the user interacts further with the authoring system. Furthermore, the authoring system should be able to consider relevant factors for automatic adaptations to the teacher/author. This approach may consider criteria such as author role and level of expertise, to show different authoring options. Each of these criteria can be represented in a matrix form for navigation and manipulation by the AEH for author specific adaptation management.

Author role management within an adaptive hypermedia tool would facilitate user administration by allowing each user group to have certain rights to view the system's capability, based on the author model, user's culture or user preferences, to name a few. This is to allow administrators to control the system access based on system setup factors. A system factor represents an author action relevant to authoring.

To illustrate this approach, we consider how this can be applied to the PEAL system ([17] , see also Chapter 2, Section 2.5.7.1.1) for the authoring of adaptive strategies. This is also to illustrate that the approach is generic and is applicable to other adaptive hypermedia authoring tools.

To produce a simple role-based access in the PEAL system, new authors accessing the PEAL system would not be shown the 'Strategy Editor' (section 2.5.7.1.1, [Figure 14](#)), which allows for text-based authoring, aimed at programming savvy authors; instead, they would see a simplified view of the system's capabilities, which would take into account the author's existing knowledge about the system. Thus, for new authors, the PEAL Graphical Editor (section 2.5.7.1.1), allowing for graphical creation of strategies, and aimed at non-technical authors, would be more appropriate.

To keep the approach as generic as possible, our approach proposes a functionality matrix that categorises system capabilities, based on author roles. The suggested roles within a recent research [37] are classified as: *Content Author*, *New Author*, *Intermediate Author*, *Experienced Author* and *Administrator*. The matrix

role-based approach matches the users' roles against the system features, as shown in Figure 19, in its instantiation for PEAL.

This is clearly an extensible approach, which allows for more authoring roles and system features to be added in the future (in line with the latest research). Furthermore, it allows for allocation of system features to various authoring roles, making it flexible and scalable for future changes. The PEAL capabilities (displayed in Figure 15) include, amongst others, the option of accessing the Strategy Editor, accessing the Graphical Editor, the user being able to view and/or edit the content or strategies in both private (user's own) and/or public (general multi-user area) spaces. Figure 19 shows *user roles* as Content Author (i), New Author (ii), Intermediate (iii) Author, Experienced Author (iv) and Administrator (v), where '0' means disabled and '1' means enabled.

| PEAL Capabilities | User roles | | | | |
|----------------------------|------------|------|-------|------|-----|
| | (i) | (ii) | (iii) | (iv) | (v) |
| Strategy Editor | 0 | 0 | 0 | 1 | 1 |
| Graphical Editor | 0 | 1 | 1 | 1 | 1 |
| Quick Actions menu | 1 | 1 | 1 | 1 | 1 |
| Admin Options | 0 | 0 | 0 | 0 | 1 |
| Editing Private Strategies | 0 | 1 | 1 | 1 | 1 |
| Editing Public Strategies | 0 | 0 | 1 | 1 | 1 |
| View Strategies | 1 | 1 | 1 | 1 | 1 |
| Editing Private Content | 1 | 0 | 0 | 0 | 1 |
| Editing Public Content | 1 | 0 | 0 | 0 | 1 |
| View Content | 1 | 1 | 1 | 1 | 1 |

Figure 19: Functionality Matrix for PEAL

The upcoming section discusses different scenarios, to clarify the need of author roles in AEH systems, and identify what features the different authoring roles to be implemented by an adaptive educational authoring system would need to have, thus leading to the system requirements. These scenarios are based

on the different types of users, as proposed by [37]: Content Author (i), New Author (ii), Intermediate (iii) Author, Experienced Author (iv) and Administrator (v).

4.2.1 First Scenario: The Content Author

A content author, Dr. Alam, wishes to create an adaptive course for first year Roman history students at the University of Warwick. He intends to apply an authoring system that guarantees content reuse for automatic authoring. Based on his previous experience, he is confident in using the PEAL system for the purpose of authoring his adaptive content adaptation. As Dr. Alam is not a programming savvy author, he wishes to create course contents only, and therefore wishes to use existing adaptive strategies created by other authors. He expects the system to help him in the content creation process and allow for reuse of previously produced material and assist him in choosing the best adaptive strategy.

4.2.1.1 Scenario driven requirements:

The system should be able to recognise him as a content author. The system should engage in necessary adaptations to the environment, in order to cater to his preferences and background. Furthermore, the system should also show previously produced content by him and other authors. It should also make accessible any previously created adaptive strategies (from the other authors); finally, the adaptations should be maintained by the system administrator, to address future needs, as the author becomes accustomed to the environment.

4.2.2 Second Scenario: A New Author

Dr. Yaacob, a new author, wishes to create an adaptive course for first year computer science students at the University of Warwick. She is aware that adaptive hypermedia has shown to result in a better learning experience, due to its capabilities for learner customisation [2]. She wishes to use an authoring system that ensures capabilities of automatic authoring and reusability. Based on her requirements, she has decided to use the PEAL system, for the purpose of authoring her adaptation strategies, whilst reusing the content from other authors.

However, being a new author, she lacks the full understanding of the capabilities of the adaptive system and its utilisation. As she is not a programming savvy author, she just wishes to create new strategies, by using the existing adaptive strategies, created by other authors. She expects the system to help in the creation process, and in choosing the best adaptive strategy.

4.2.2.1 Scenario driven Requirements:

The system should identify her as a new author, with a strong focus on strategy authoring. The system should enforce any necessary adaptations to the environment, in order to facilitate her preferences and background. These adaptations may include previously created content and access to graphical tools for content creation. Her time spent on content authoring and system interaction is to be recorded and considered for upgrading her role by the system administrator. The applicable adaptations are to be maintained by the system administrator, for future demands and author familiarization to the environment.

The automaticity surrounding the author recognition, environment customisation and *user role* promotion implies inclusion of the required features in the authoring process.

4.2.3 Third Scenario: An Intermediate Author

Dr. Bertrand is a moderately experienced author, with previous experience in creating strategies used within an adaptive course at the University of Paris. He is confident in creating strategies and wishes to share them with his colleagues. He is aware that PEAL has recently promoted him to an ‘intermediate author’. He takes advantage of this and contacts colleagues and other PEAL users with his new strategies. His experience allows him to be considered a ‘somewhat’ programming savvy author. However, his expectation from the system is to aid him in new strategy creation and also the selection of existing most appropriate strategies.

4.2.3.1 Scenario driven Requirements:

The system should assist him as an intermediate author and apply necessary adaptations to the environment, based on his preferences and background. He should be allowed to share his strategies with

other authors. His further interaction with the system needs consideration by the administrator, for promotion to an Experienced Author role.

4.2.4 Fourth Scenario: The Experienced Author

Professor Staley, an expert author, wishes to create an adaptive course for his final year Computer Science students at the University of Warwick. He is well-aware of the AEH system advantages, including customized learning experience and lower learning overheads for the students. Based on his profile and expertise, he wishes to use an authoring tool that allows for automatic authoring, re-use and interoperability. With his advanced understanding of the adaptive system and its capabilities, he selects the PEAL system, to author his strategies. PEAL allows him to not only edit strategies in the graphical authoring tool, but also offers the more detailed and complex (and subsequently more powerful) strategy editing tool.

4.2.4.1 Scenario driven Requirements:

The system should identify him as an experienced author and apply any necessary adaptations to the environment, in order to facilitate his role and preferences. As this is the highest-level author, he should have access to any previous strategy authoring permissions (be able to edit both public and private space). He should be able to access the textual strategy authoring tool.

4.2.5 Fifth Scenario: Administrator

Mr. Kesteven is the system administrator. He intends to have a full grasp on maintaining user roles, and their rights and access to the various system capabilities.

He wishes to be able to grant rights based on his understanding (and knowledge of any bugs or issues relevant to the running of PEAL). With his privileges, he may disable or enable access to the various system capabilities for any group of users. He may also upgrade or down grade a user role, based on a participant's activity within the system and would like to have access to the system interaction dataset for various user groups.

4.2.5.1 Scenario driven Requirements:

Mr. Kesteven should be identified by the system as an administrator. All necessary adaptations should be applied to the environment, to facilitate administrative options, such as user role matrix management. The automatic user recognition, environment adaptation and access to system administration facilities are key requirements for the administration group users.

4.2.6 Environment for the Content Author

Upon login, content authors see relevant options, the content creation toolbar adapts according to their role. This allows for the system capabilities to be filtered by user roles. Any other components can also be fine-tuned, such as the option to display a sub-set of LAG programming constructs. As the author becomes more experienced, more constructs of LAG are shown. Additionally, the environment could adapt itself and show construct-specific help, to facilitate the usage of new constructs in the adaptive specification.

4.2.7 Environment for the New Author

The ability to identify a user-specific role facilitates the adaptation of the authoring environment and assignment of associated role rights to access system options. These adaptations are applied to the user interface and relevant toolbar options are shown to the user. Permissions are also maintained, to establish the level of visibility for public, private and shared strategies and possible interactions with them in terms of the ability to modify or remove a strategy.

4.2.8 Environment for the Experienced Author

When the experienced author logs into the system, the system shows the applicable options for the relevant author role. For instance, the Strategy Editor is shown to experienced authors as it targets programming savvy authors, giving them full access to their adaptive strategies. These authors are also shown the relevant options of the system, i.e., programming constructs, such as 'IF', 'ELSE', 'FOR' and 'WHILE' loops and any additional syntax that may be appropriate for this level of strategy authoring.

4.2.9 Administrator Environment

For administrators, the system identifies their role, and presents relevant options to the user. These options may include user management, user interaction analysis and strategy assignment. The administrator may consider relevant data of interest as criteria for user promotions to subsequent groups.

4.2.10 Quick Action Menu

Users may interact with the system via quick actions; these could be displayed on a toolbar or as a mouse context menu. This arrangement facilitates ease of navigation and selection. An author should be able to organize the items on the toolbar, according to their own preferences. They should be able to visualize the content and be able to evaluate its delivery, based on specification application and tweaks. Any adjustments should be allowed to the content, to suit the needs of the learners. Also, this feature may also be used to view how existing strategies can be applied with ease, so the author can view how existing strategies will be displayed to the learner, hence making it easy for the author to choose a strategy.

4.2.11 Evaluation

After introducing these roles, via their respective requirements, a system implementer would need to perform various evaluations, in order to ensure that the roles are implemented according to their initial plan. Below, some suggestions on such implementations are given.

From a system's perspective, the evaluations could include:

- Are authors changing needs considered in this methodology?
- Do author roles and their mapping to system capabilities impact on adaptive authoring?
 - What is the impact of an adaptable toolbar or context menu to the author?

The evaluation from the author's perspective could address the following:

- Does the methodology display a less complicated view of the system to authors?
- Does the methodology improve the adaptive strategies and content reuse?
- Can the methodology be expanded, to include new user roles and system capabilities?

- Does the methodology provide the desired adaptation?
- Does the methodology improve the learning experience of the author?
- Does the toolbar provide help to the author during authoring?

The evaluation from the administrator's perspective could address:

- Does the proposed functionality matrix result in better user management?

The actual evaluations performed in this thesis are further detailed in Chapter 121.

4.3 Conclusion

This chapter has discussed a *role-based* approach to the authoring paradigm, allowing a system implementer or administrator to allocate each author to a predefined user role. This discussion is based on prior theory [37], adjusted for the area of authoring of adaptive educational hypermedia. Each role can be assigned a set of authoring system capabilities (represented by a context menu or a toolbar) which further defines certain system adaptations, according to a user role. A simple matrix can define the allocation of capabilities to the user role. These capability allocations should be able to be translated by the system (e.g. PEAL) automatically, for example into a toolbar, to provide quick. The custom tailoring of these capabilities and their assignments to various user roles allows for adaptation of the environment based on the author role – an idea less explored in adaptive hypermedia, where the user taken into account is often only the end-user (e.g., the learner in learning systems) and not the creator of the material (the author). Administrators should be able to manage the users within the system and can monitor user progress within the authoring stream. A Functionality Matrix (FM), as detailed in 4.2, allows for easier user management, because it lists all the author roles within a system, as well as their access permissions. As authors progress and improve their level of expertise, they can be promoted to higher groups and can enjoy more privileges. These privileges may include access to more advanced language constructs and system options. Several roles may be allowed to share their adaptive strategies with other users with varying level of permission, based on their active roles. Advanced privileges and membership of an expert group allows for authors to be more interactive with the environment and be more involved in other collaborative activities, such as strategy sharing and co-authoring.

The chapter highlighted some core roles which may be needed by any adaptive authoring environment. These roles are based on the level of expertise and knowledge of the language syntax. It is assumed that novice authors will become experienced with time and interaction. Any authoring system should take into account different types of users belonging to multiple roles. Though, initially, the matrix considers the user level of expertise, it may be expanded to include author/user demographics, such as knowledge of the subject, Geo-location, culture, etc. These additional demographics would provide information about the author and could be used to provide better adaptations within the authoring environment – to the author, and thus, ultimately, to the end-user as well (e.g., a student in a learning environment).

5 Creating a Visual language for LAG and constructing VASE

5.1 Introduction

This chapter addresses the following objectives.

Objective 3: - *Propose a visual programming framework to provide complete and correct adaptive specifications, and further sustaining the “separation of concerns” principles, and interoperability.*

Objective 4 – *Implement a visual programming framework to enhance adaptive authoring systems and adaptation languages. To propose a suitable framework for adaptive authoring, to allow authors to create visual adaptive specification, by manipulating visual elements to create adaptive specification, thereby lowering the programming threshold for authors.*

Outcomes of this chapter:

This chapter deals with a new visual adaptation specification creation, according to Objective 3. The creation of an adaptation specification is one of the more difficult processes in creating an adaptive hypermedia course. To reduce this difficulty, adaptation languages have started to be developed, which increase the reusability of adaptation specifications. This chapter proposes enhancements to improve the LAG language via a visual specification.

This chapter also addresses the first part of Objective 4, “propose a suitable framework”, to design an adaptive authoring system through the evaluation of the system design iteration.

Finally, this chapter presents the final update on the theoretical framework and architecture necessary for the extended features to address the research questions.




5.2 LAGBlocks: the visual language for adaptive hypermedia

As explained in Chapter 2, Section 2.4.5, the LAG language has been selected as the basis on which to build a more author-friendly language, which could be used, especially, by novice authors. As also explained, the visual authoring paradigm has been selected, due to its clear advantages, including easy of

use (see Chapter 2, Section 2.9). As a result, for this thesis, a visual language, called LAGBlocks, was created on top of the LAG language, to represent any LAG statement in a visual form. The purpose of LAGBlocks was to allow Adaptive Hypermedia authors to create an adaptive strategy, without having to type any LAG programming syntax, while keeping all the other tools which use the LAG programming language, such as ADE [15] (and MOT [58] (see also 0), unaffected. The strategies created by using LAGBlocks should be able to work in ADE without any further work from the author. Moreover, new adaptive specifications should be able to be created by mixing and merging existing ones. New strategies should be able to be created with ease, by reusing existing strategies. To save time, existing LAG strategies should be able to be created by using list LAG-Blocks templates.

Thus, to create a visual representation for LAG, which should be user-friendly, and familiar to users, the well-known OpenBlocks [39] framework was selected as the basis of the development. This framework was further extended, to create a visual representation of the LAG programming language, called LAGBlocks, to represent all of the LAG programming language grammar, including the data types and logic constructs. In the following, this process is described in more details.

LAG has three data types, and thus each needed to be represented by a unique shape, providing clues to the user on how these will fit together. The allocated shapes for each data type in the LAG adaptation language are:

- a double rectangle, , for the LAG string type;
- a rectangle with rounded edges, , for the LAG Boolean type;
- a hexagon, , for the LAG int type.

In the following, the usage and creation of LAGBlocks is illustrated by examples. The way these constructs have been created, by using the OpenBlocks framework, is also shown.

Figure 20 shows the corresponding new LAGBlock of a very frequently used construct, that of specifying that a concept *Concept* (in the goal model, *GM*) has been accessed (*access == true*). As *access* is a variable

in the user model, the construct is prefixed with *UM*. Below the LAGBlock, the corresponding code in the OpenBlocks framework is listed. One can see how the blocks fit each other, as they are defined as ‘BlockConnector’, for instance.



```
<BlockGenus name="UM.GM.Concept.access" kind="function" initlabel="UM.GM.Concept.access" color="255 155 64">
  <description><text>Allows access to a concept</text></description>
  <BlockConnectors>
    <BlockConnector connector-kind="plug" connector-type="boolean"></BlockConnector>
    <BlockConnector label="" connector-kind="socket" connector-type="boolean">
      <DefaultArg genus-name="true" label="true"></DefaultArg>
    </BlockConnector>
  </BlockConnectors>
  <LangSpecProperties>
    <LangSpecProperty key="vm-cmd-name" value="UM.GM.Concept.access"></LangSpecProperty>
    <LangSpecProperty key="is-monitorable" value="breed"></LangSpecProperty>
  </LangSpecProperties>
</BlockGenus>
```

Figure 20: LAGBlock to control the User Access to Goal Model instances

Figure 21 illustrates a LAGBlock that attaches a given label (here ‘*beg*’ for beginner user, for instance) to the current concept *Concept* in the goal model *GM*. The figure further shows the OpenBlocks code to generate this particular construct. ‘*beg*’ can be seen there to be defined a label of the type ‘*string*’, for instance.



```
<BlockGenus name="GM.Concept.label" kind="function" initlabel="GM.Concept.label" color="27 180 225">
  <description><text>Allows access to a concept</text></description>
  <BlockConnectors>
    <BlockConnector connector-kind="plug" connector-type="boolean"></BlockConnector>
    <BlockConnector label="" connector-kind="socket" connector-type="string">
      <DefaultArg genus-name="string" label="beg"></DefaultArg>
    </BlockConnector>
  </BlockConnectors>
  <LangSpecProperties>
    <LangSpecProperty key="vm-cmd-name" value="UM.GM.Concept.access"></LangSpecProperty>
    <LangSpecProperty key="is-monitorable" value="breed"></LangSpecProperty>
  </LangSpecProperties>
</BlockGenus>
```

Figure 21: LAGBlock - To specify Goal Model Concept label

Next, Figure 22 shows the corresponding LAGBlock, where the user model *UM* knowledge level *knowlvl* for the goal model *GM* is set to beginner '*beg*'. Thus, for example, for a strategy connecting beginner users to concepts dedicated to beginners, blocks such as in Figure 22 and Figure 23 would need be used. Figure 23 further also lists the OpenBlocks code to create this new construct. A similar procedure as in the examples above is used. Such code has been listed up to here also to help further implementers who would wish to use OpenBlocks for implementing slightly different concepts to the ones used in the current authoring language, which is proposed in this thesis. In the further examples, the OpenBlocks code is omitted, as it is less central to the aim of the research.



```
<BlockGenus name="UM.GM.knowlvl" kind="command" initlabel="UM.GM.knowlvl" color="255 155 64">
  <description>
    <text>Increments the number by the value attached to this ... </text>
  </description>
  <BlockConnectors>
    <BlockConnector connector-kind="socket" connector-type="string">
      <DefaultArg genus-name="string" label="beg"></DefaultArg>
    </BlockConnector>
  </BlockConnectors>
  <LangSpecProperties>
    <LangSpecProperty key="vm-cmd-name" value="UM.GM.knowlvl"></LangSpecProperty>
    <LangSpecProperty key="stack-type" value="breed"></LangSpecProperty>
  </LangSpecProperties>
</BlockGenus>
```

Figure 22: LAGBlock - To set the User Knowledge level for a Goal Model Instance

Further, Figure 23 represents only the LAGBlock construct to display an IF-condition. Hooks for placing the condition and the resulting consequence of the condition being fulfilled are also present (right side of the image).



Figure 23: LAGBlock - If Construct

Next, Figure 24 shows a snapshot of the Visual Adaptive Editor proposed (VASE). On the right side, the author can choose between the different models, that are according to the LAOS framework (described earlier in Chapter 2 Section 2.4.3), such as User Model, Presentation Model and Goal Model. Adaptation constructs can be accessed in the grey menu items below. In the right panel, the view shows what can be accessed if the author clicks in the User Model option. Various User Model related constructs are available to select, colour-coded in the same way as the User Model (orange).



Figure 24: LAGBlock - to represent the LAG User Model

Figure 25 shows the equivalent snapshot, when the author has clicked on the Presentation Model instead.

The colour code here, red, refers to all constructs related to this model.

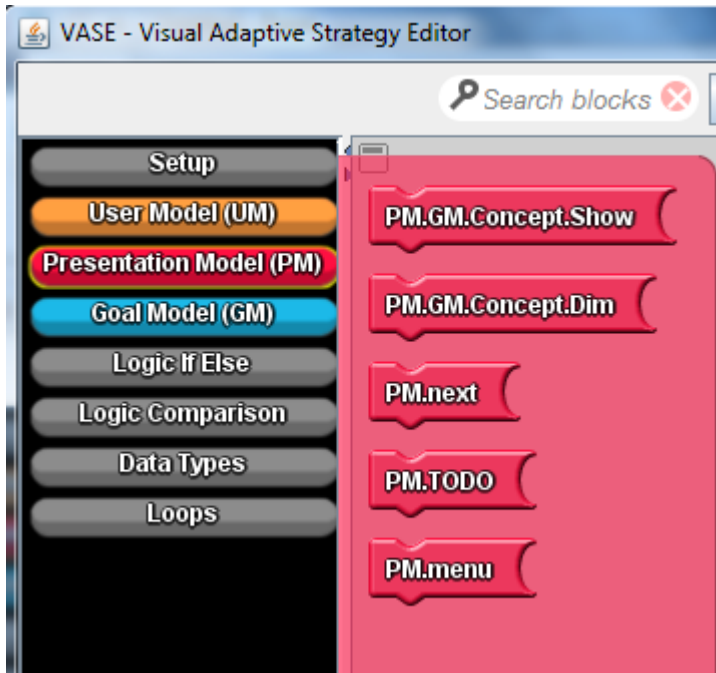


Figure 25: LAGBlock - PM construct – to represent the LAG Presentation Model

Similarly, Figure 26 displays, in the right frame, the result of an author clicking on the Goal Model. Thus, Goal Model related constructs are accessible, and colour coded (blue).

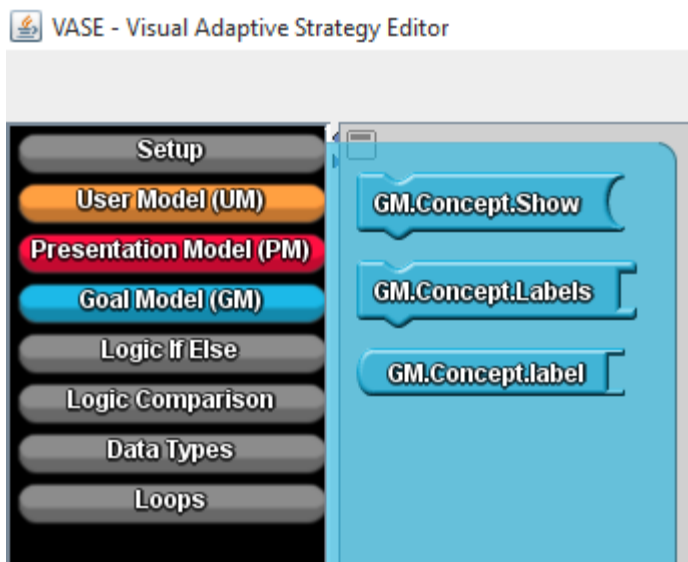


Figure 26: LAGBlock - GM construct - to represent the LAG Goal Model

5.2.1 Writing an adaptation strategy in LAGBlocks

After explaining some of the LAGBlocks in the previous section, here, an example adaptation strategy for adaptive e-learning is illustrated both in the LAG language ([11]; see also Chapter 2, Section 2.4.5), as

well as in the newly designed visual language, based on LAGBlocks (see Figure 27). The code snippet below is related to the adaptation strategy for beginner-intermediate-advanced learners, as briefly mentioned in section 2.4.5.1 (Chapter 2). It illustrates, however, a different part of that strategy, the initialisation part, where the system computes the total number of concepts that are labelled beginner, intermediate or advanced, respectively, for that particular goal model *GM*. The way the blocks fit together, and the way a loop (the for-each loop) is represented, can be seen in this figure. The complete strategy can be seen in APPENDIX H: LAG Strategies.

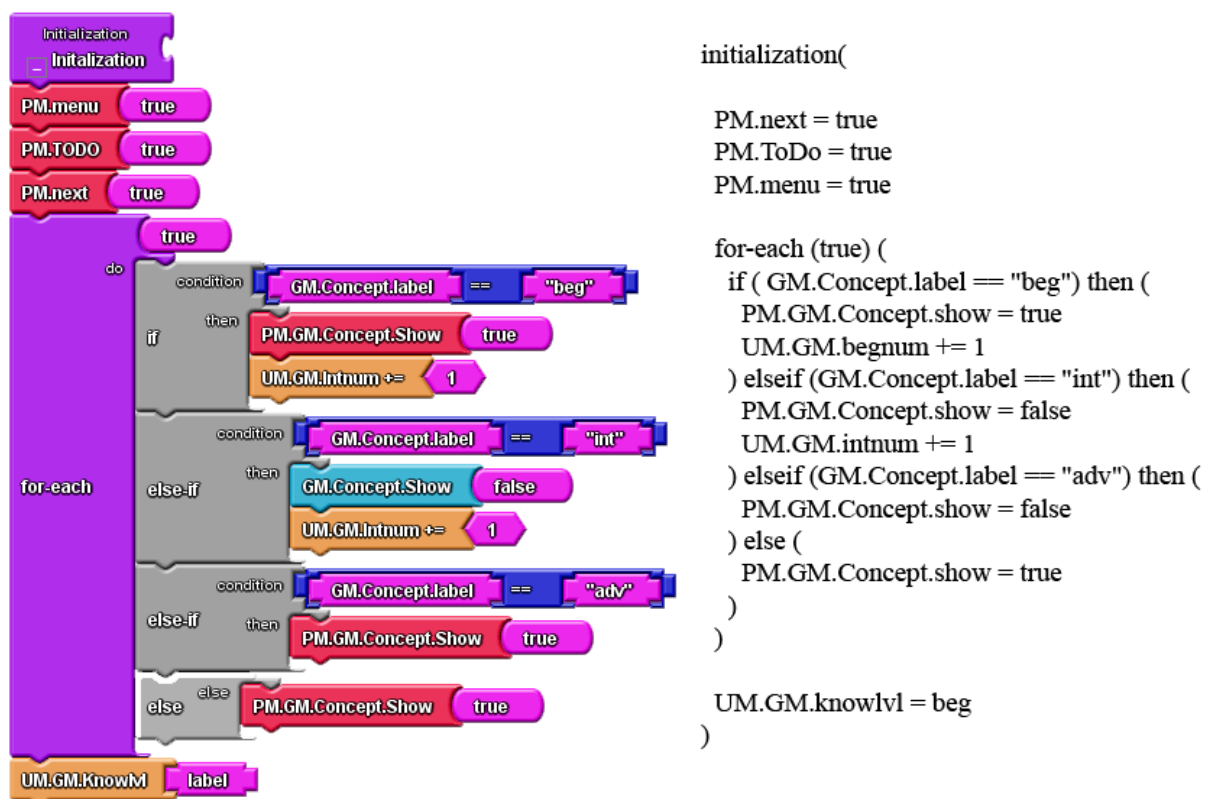


Figure 27: LAGBlock code (left) versus LAG code (right)

Another strategy snippet is illustrated in LAGBlock code in Figure 28. This is part of the initialisation of a respective LAG strategy (see complete strategy in APPENDIX H: LAG , Beginner-Intermediate-Advanced). This initialisation hides concepts labelled ‘int’ and ‘adv’, for intermediate and advanced learners, respectively, showing only concepts labelled ‘beg’, for beginners – and thus assuming that, at start, the learner is a beginner. The strategy snippet also counts how many beginner, intermediate and advanced concepts there are, respectively.

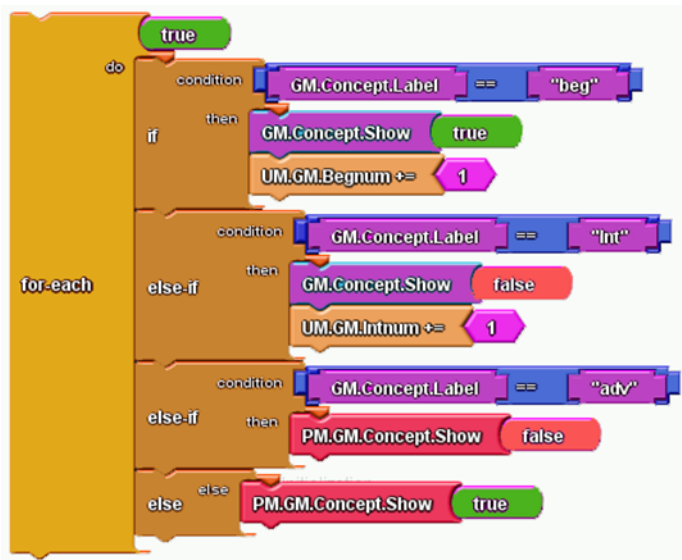


Figure 28: 'For Each' Adaptive Strategy in LAGBlocks

5.3 Enhancing the LAG programming language

This research on LAGBlocks additionally suggested new ideas to improve the LAG language, these suggestions for improving LAG adaptation language are listed in APPENDIX C: LAG Extension. Most of these features were not implemented at the LAG language grammar level, instead these were applied directly in the LAG visual extension. By applying the changes at the visual extension level, it provided improvements whilst the adaptive strategy created in the LAG language remained fully functional in other tools which used LAG as the adaptation language (for backward compatibility).

5.4 VASE 1.0: the initial approach towards a visual adaptive authoring tool

The Visual Adaptive Editor (VASE), the system that was built and used for evaluation of the theoretical development in this thesis, proposes a new approach, involving adaptive authoring (as recommended in Chapter 4). VASE was initially targeted towards creating adaptive specifications for non-programming savvy users. In order to achieve this goal, several technologies and techniques were used. The system was implemented and a pilot experiment was conducted to validate the initial features offered by the authoring system. The interaction of VASE with external tools is shown in Figure 29.

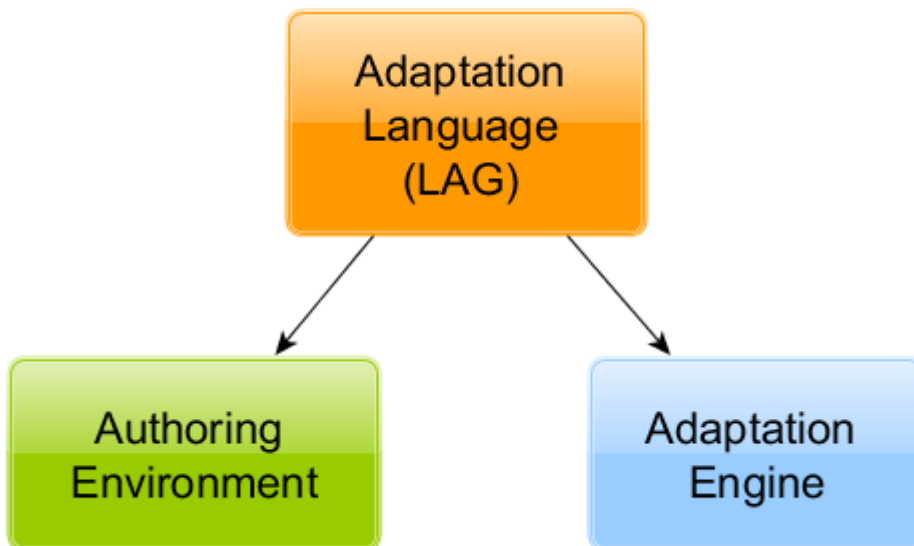


Figure 29: VASE Interaction with external tools

Initially, the system was planned to allow for the following.

- Create adaptive specifications, by authors without the knowledge or experience of an adaptation language.
- To provide personalised interfaces, according to the user-roles (see Chapter 4), i.e. the interface of the system will change, according to the user who is logged in. So for the non-programming-savvy author, an interface was to be provided, which would not confuse them.

Based on the specifications above, the first version of VASE was created, VASE 1.0. The idea was that existing adaptive strategies would be edited visually, by moving LAGBlocks around with the mouse, to alter the logic of the adaptive strategy. In order to meet the requirements mentioned earlier in this thesis for the new authoring tool, several frameworks were studied (see, e.g., Section 2.4) and were tried in a lab to see what framework could offer the best possibilities to design the new authoring tool with a uniform interface. The LAGBlocks which have matching connections can only fit with other matching blocks, ensuring code *correctness*. An auto-complete feature was also implemented, to provide code *completeness*. VASE 1.0 could already represent any LAG programming language statement.

The VASE 1.0 interface includes many of the basic features of a block programming environment, such as graphical blocks, block stencils that contain these blocks and a canvas or workspace where block programs are built and manipulated (see Figure 30).

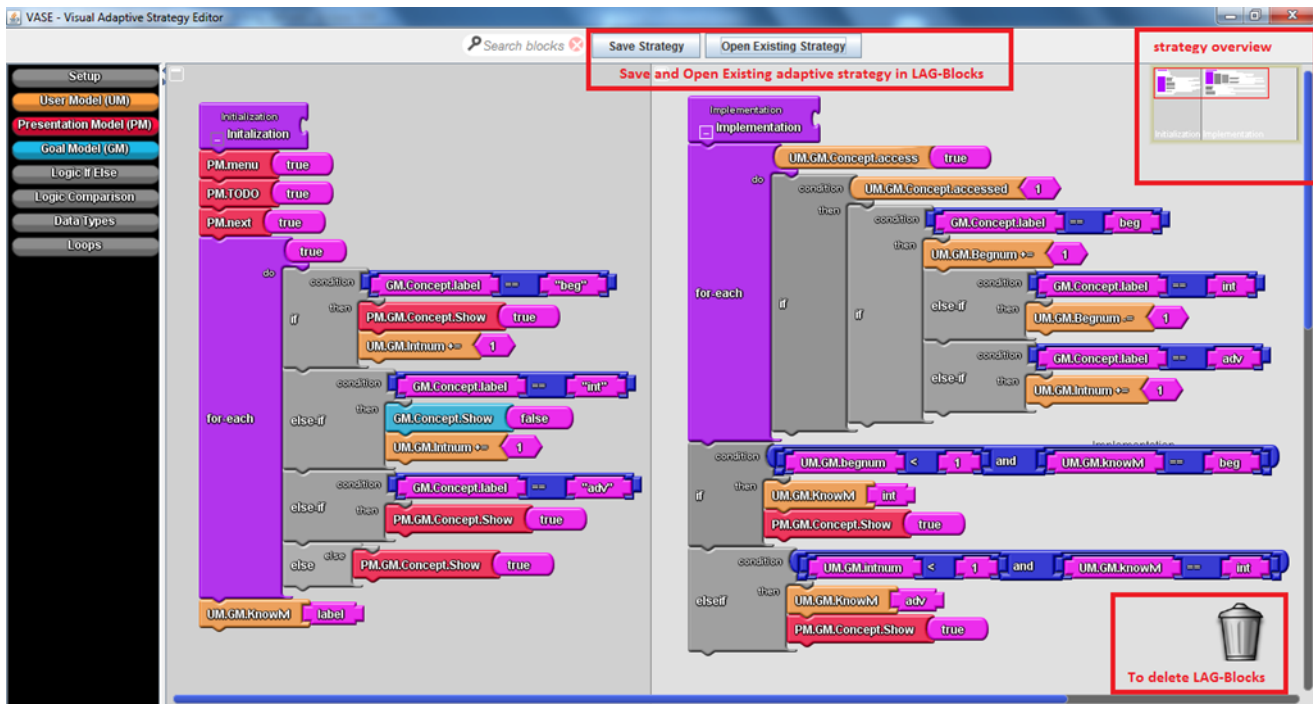


Figure 30: Screenshot of VASE 1.0

5.4.1 VASE 1.0 and the LAG strategies

By manipulating LAGBlocks, any text-based strategy written in the LAG programming language, could be represented visually in LAGBlocks. A workspace was designed, where LAGBlocks could be manipulated (drag-and-drop), to form new adaptive strategies visually. The user interaction with LAGBlocks is aimed to be intuitive, as it looks like puzzle pieces, which allow users to understand which blocks can or cannot fit together, just by looking at the block connector shape.

Initially, twelve adaptive strategies, which had been used in the past to showcase the full range of the LAG language, were represented as LAGBlocks, as a proof of concept, to ensure that existing LAG strategies can work in block programming environments. Figure 31 illustrates one such simple strategy, a dimming strategy. It checks for concepts labelled 'dim', and then dims them (sets the 'Dim' value to *true* in the

presentation model PM), initially; later, after these concepts are accessed, the dimming is removed (set to *false*). The equivalent LAG strategy can be found in the APPENDIX H: LAG , Dimming.

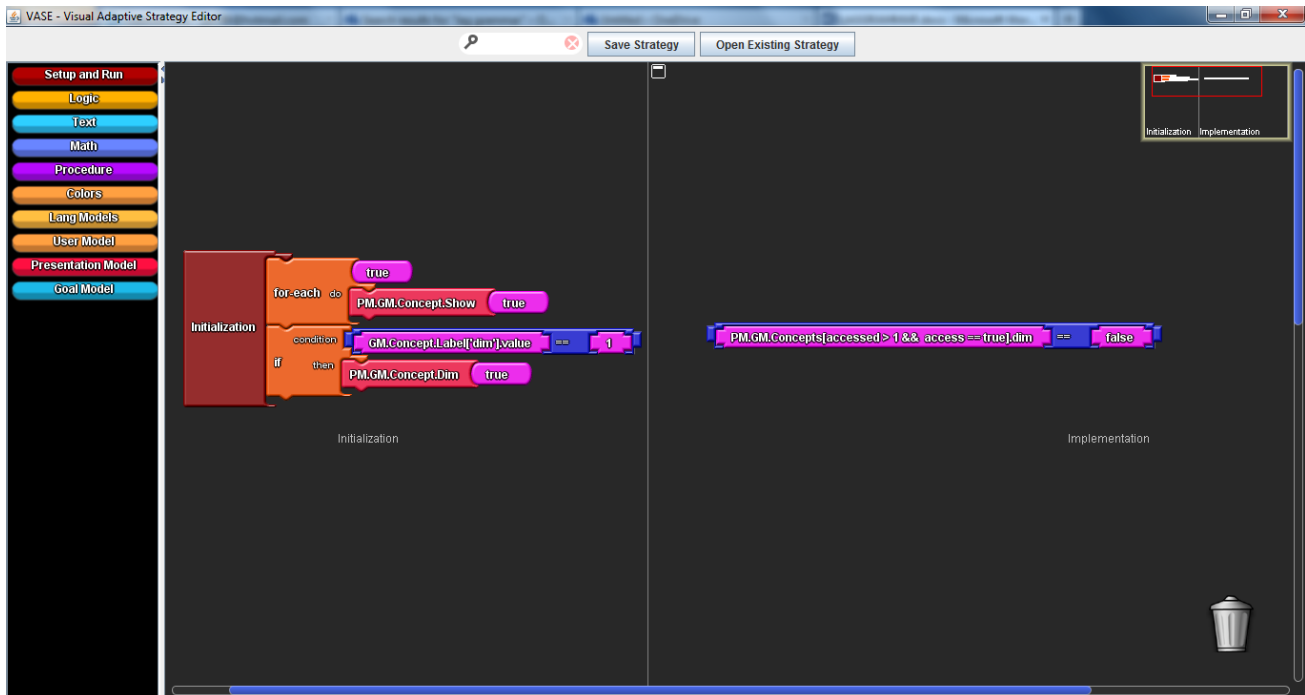


Figure 31: Dimming Strategy in LAGBlocks in VASE 1.0

5.4.2 VASE 1.0 Features

Being able to reproduce the existing adaptive strategies from LAG in LAGBlocks ensures that these will continue to work, since no change was made to the underlying LAG grammar.

Additionally, LAGBlocks were used to encapsulate very useful features, such as code *correctness* and *completeness*, as only valid constructs with matching types will connect.

In the following, other features of VASE 1.0 are showcased.

Figure 32 shows how to start a strategy, with the two initial building blocks, the one allowing for the description of the ‘initialisation’ and the other describing the ‘implementation’ loop, respectively.

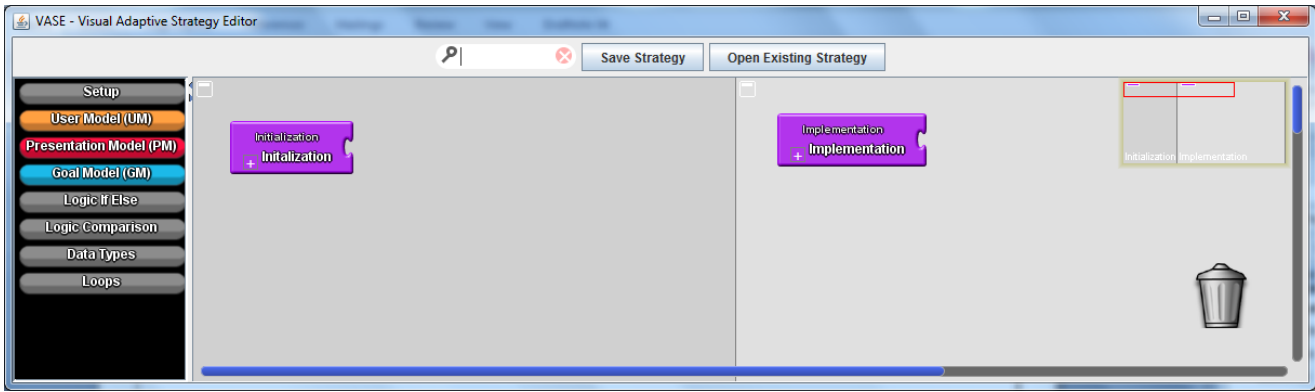


Figure 32: VASE 1.0: the start

Figure 33 showcases the basic creation windows for the LAGBlocks, inherited from the LAG language: the initialisation pane and the implementation pane (the latter describing the running loop after the initial initialisation), with a complete strategy in them.

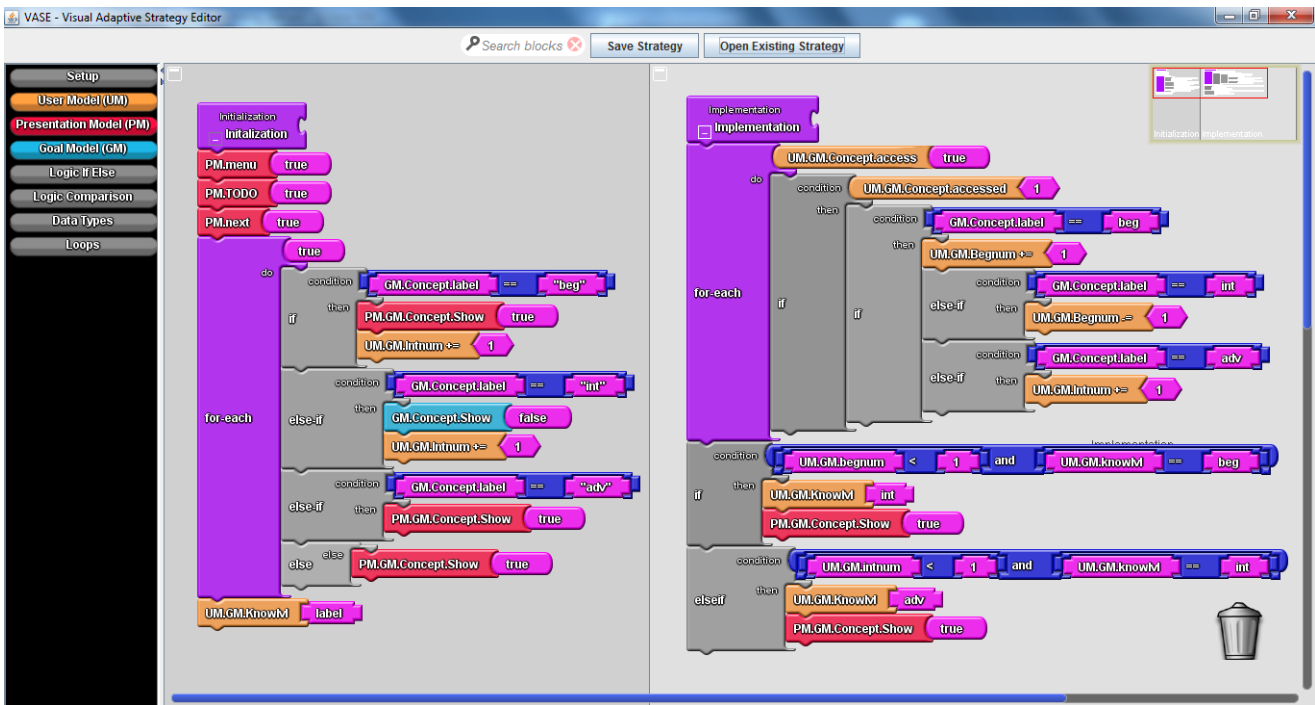


Figure 33: VASE 1.0 Interface - Initialization (on left pane) Implementation (on right pane)

Figure 34 illustrates features such as saving a newly created strategy, opening an existing strategy, a strategy overview panel, which could help switch between the initialisation and the implementation part of the LAG strategy, and a very familiar bin representation for deleting a strategy.

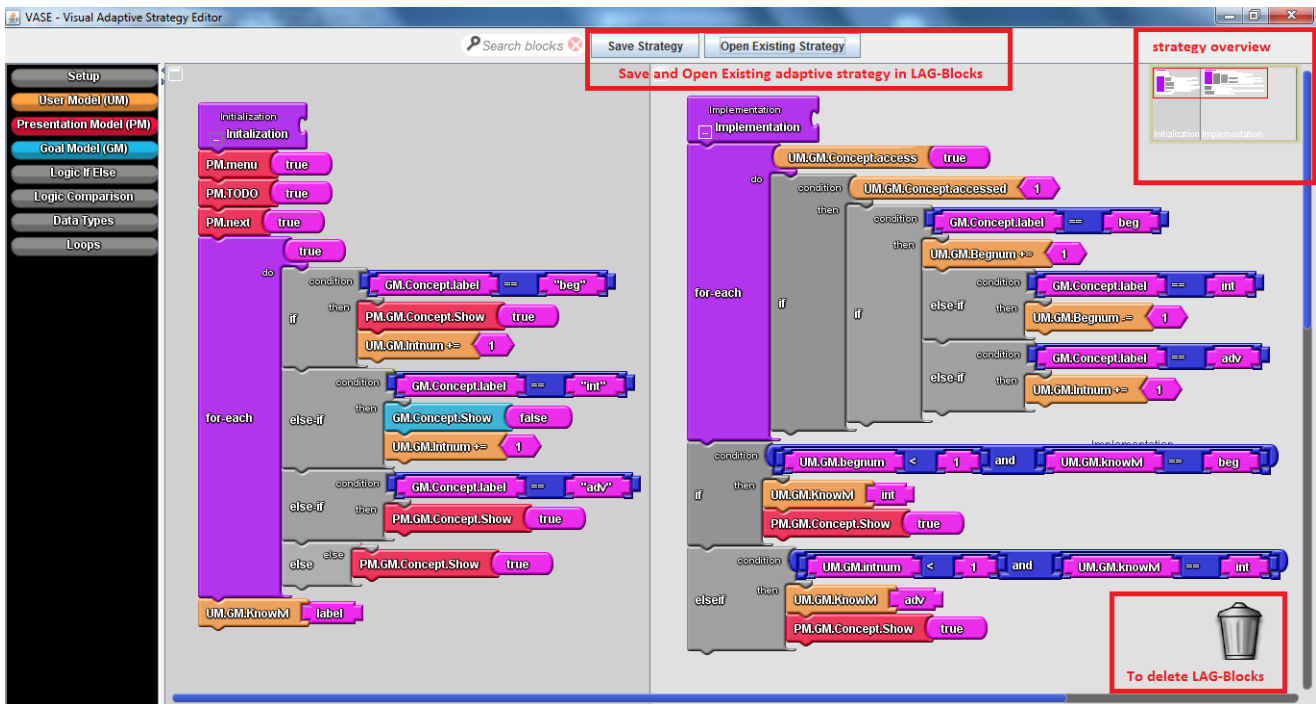


Figure 34: VASE 1.0 Interface – red boxes to represent various features

Figure 35 illustrates the compactness in VASE 1.0, where several conditions can be ‘tucked in’ into one ‘if’ (or ‘elseif’) construct.

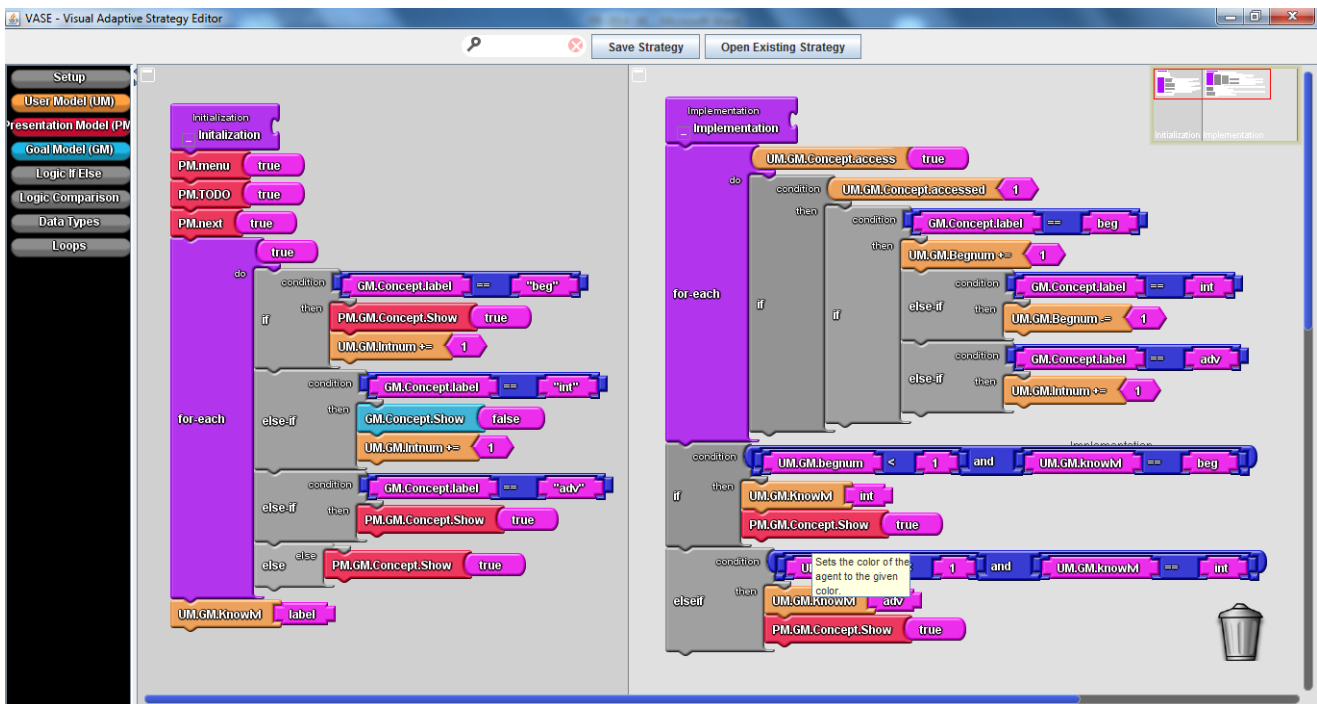


Figure 35: VASE 1.0: To show how compactness works

5.4.3 VASE 1.0 limitations

VASE 1.0 needed a plug-in to be installed to be able to run it. To make VASE interoperable (RQ 1.3), it would have needed to run in a browser without installing any additional plug-ins.

The other major limitation was that it was not possible at the time to create LAG code from the visual representation.

5.4.4 OpenBlocks's limitations uncovered by this research

The following issues were found with OpenBlocks, based on the implementation of VASE 1.0 in OpenBlocks:

- OpenBlocks [39] lacked one very important feature: it was not interoperable, it needed a Java plugin to run in a browser. In most academic settings, it may be difficult for authors to have permission to install an unknown and unsigned plugin on a local computer.
- In addition it wasn't compatible with all Internet browsers.

- The copy and paste functionality of LAGBlocks feature was requested during the user feedback session (see Table 3).
- It was also difficult to collect the usage data about the user interaction, so it was difficult to know how users were interacting with the authoring tool.

Therefore, a more suitable solution was needed, which would enable authors to use any internet browser or the new authoring tool on the local computer without any special permission.

5.5 VASE 2.0: The Extended Adaptive Authoring System

After new research in the area of block-based programming, based on the limitations (Chapter 5, Section 5.4.3) evident from the implementation, usage and experiment (Chapter 6, Section 6.5) of VASE 1.0, Google Blockly was chosen to have the desired characteristics for the framework, to create a visual environment for creating an adaptive specification visually.

Each of the LAGBlocks created in the OpenBlocks [39] framework using Java was recreated in the Blockly framework, which used the JavaScript programming language. This meant that the constructs created in Google Blockly would work on any internet browser, including mobile and tablet computers. The environment for manipulating LAGBlocks, VASE, was also developed by using HTML and JavaScript, so it can be interoperable, e.g., run on all browsers without using or installing any third part plugin. In this section, the updated system designed to be reflected upon the final version of VASE 2.0 is discussed. This section targets the refined set of requirements and features that have been explored in the previous chapters. It also presents an update on the framework and architecture (see Figure 36 and Figure 37) to include a set of authoring features addressed in the research questions (RQ1.3 and RQ1.4) listed in section 1.4.1.

VASE 2.0 was designed to address these limitations above (see Section 5.4.3) and the feedback received from the users (Chapter 6, Section 6.5.2). VASE 2.0 was designed in a completely new language and platform to address the limitation of VASE 1.0; it was designed on the HTTP protocol. Interoperability was a major factor in the redesign of VASE 2.0 and the HTTP protocol addresses this very well. A

LAGblock converter was also implemented in the redesign to be able to create LAG programming language code from the visual representation.

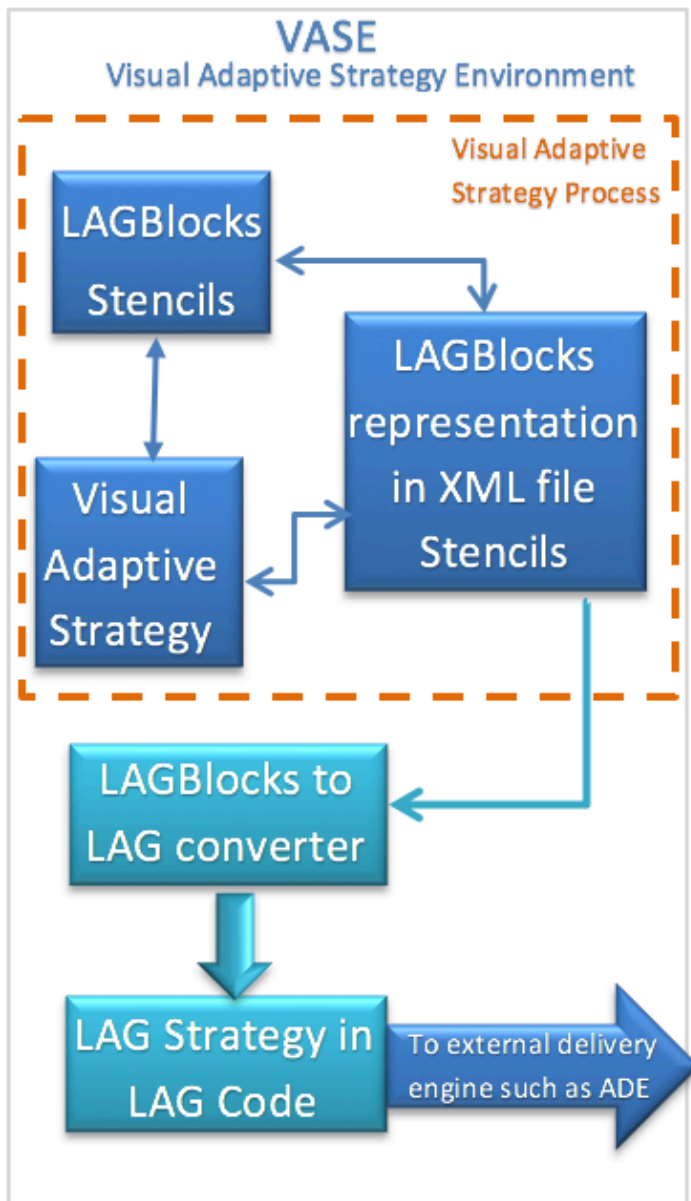


Figure 36: VASE 2.0 System Architecture

Figure 36 shows interaction between different VASE 2.0 internal components, it shows *adaptive authoring process* inner working and how it converts visual representation of adaptive strategy into LAG programming language code. The visual representation of adaptive strategy is converted into LAG programming code. It ensures that the visual representation adaptive strategy is interoperable with other

tools (ADE, MOT, etc) which require with LAG programming code. See chapter 5, Section 5.5.3 for more details on converting the visual representation into the LAG programming code.

5.5.1 Design of VASE 2.0

The VASE authoring tool was focused towards providing adaptive strategy authoring for non-programming savvy authors, and the features that help authors in adaptive strategy creation and reuse. The design of the system focused on producing an authoring tool which is acceptable for users in an academic setting as well as functional and usable. Figure 37 shows the new VASE 2.0 architecture diagram. User roles, as well as output, is depicted in the figure.

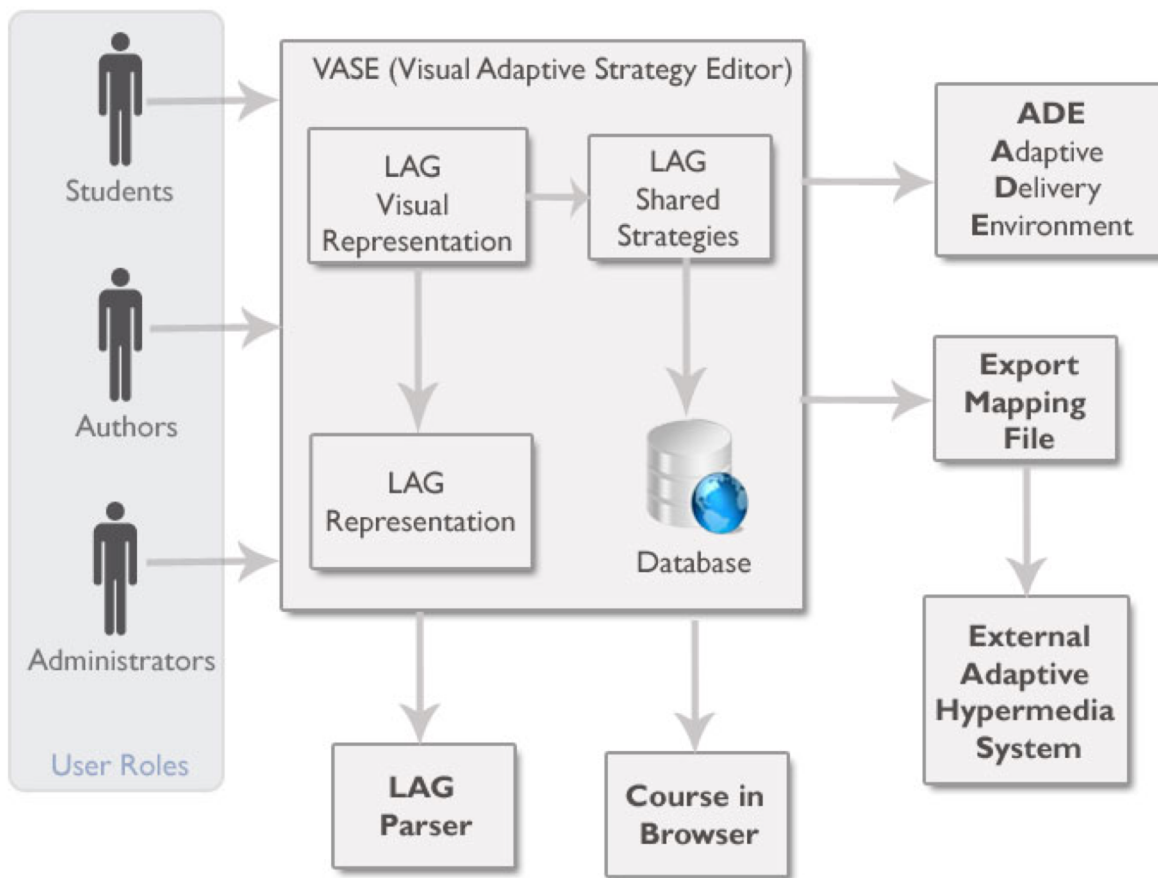


Figure 37: VASE 2.0 Architecture Diagram

In order to satisfy the above goal, the next iteration of VASE (2.0) authoring system implemented the following changes.

- The new design was based on browser technology, to avoid installation of any additional plugin to run the authoring system. JavaScript was chosen as the language for the development, because it is supported by all Internet browsers, including the mobile browsers.
- A new visual framework, which appeared to be very similar, with the same affordance, was chosen; it was created by Google and it was written in JavaScript. This seemed an ideal candidate for this purpose.
- A data collection web service was written, to store the events generated by users, to be able to track user behaviour.

5.5.2 User Interfaces

The resulting user interfaces, following a similar look and feel to the VASE 1.0 system, but with improved functionality, are illustrated by snapshots in Figure 38 to Figure 50.

Figure 38 shows an overall snapshot of the Visual Adaptive Editor proposed (VASE 2.0). On the right side, the author can choose between the different models, that are designed according to the LAOS framework (described earlier in Chapter 2 Section 2.4.3), such as User Model, Presentation Model and Goal Model.

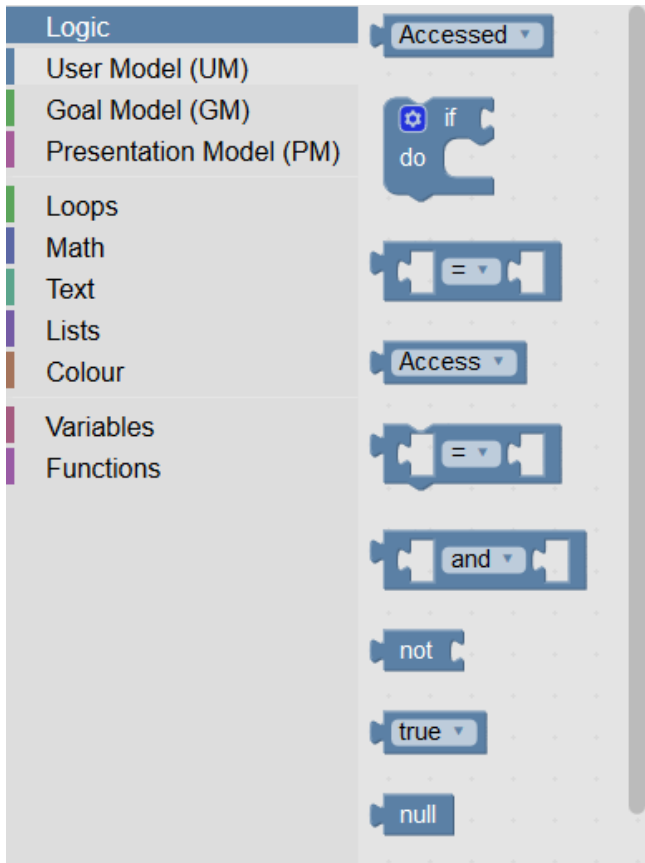


Figure 38: VASE 2.0 Logic Blocks

Moreover, additional blocks were created to facilitate adaptive strategy creation such as *Logic* blocks to construct LAG logic statements visually. All of the logic blocks had a notch which could be used to plug it in with any *if* Block. For creating if statement within if statement, additional blocks were created to combine two conditional statement into one. This way any complex *if* statement can be represented by using combination of conditional statement constructs, as depicted in Figure 39. It can be used to represent far more complicated conditions than have been shown in the Figure 39, additionally this approach is very flexibility and intuitive for the author, to create complex conditional statements using LAGBlocks.

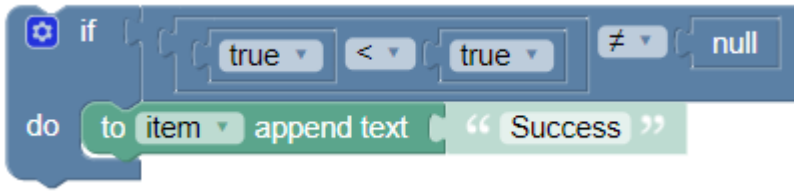


Figure 39: VASE 2.0 Logic Blocks complex usage

Next, in Figure 40, in the right panel, the view shows what can be accessed if the author clicks on the User Model option. Various User Model-related constructs are available to be selected, colour-coded in the same way as the User Model.

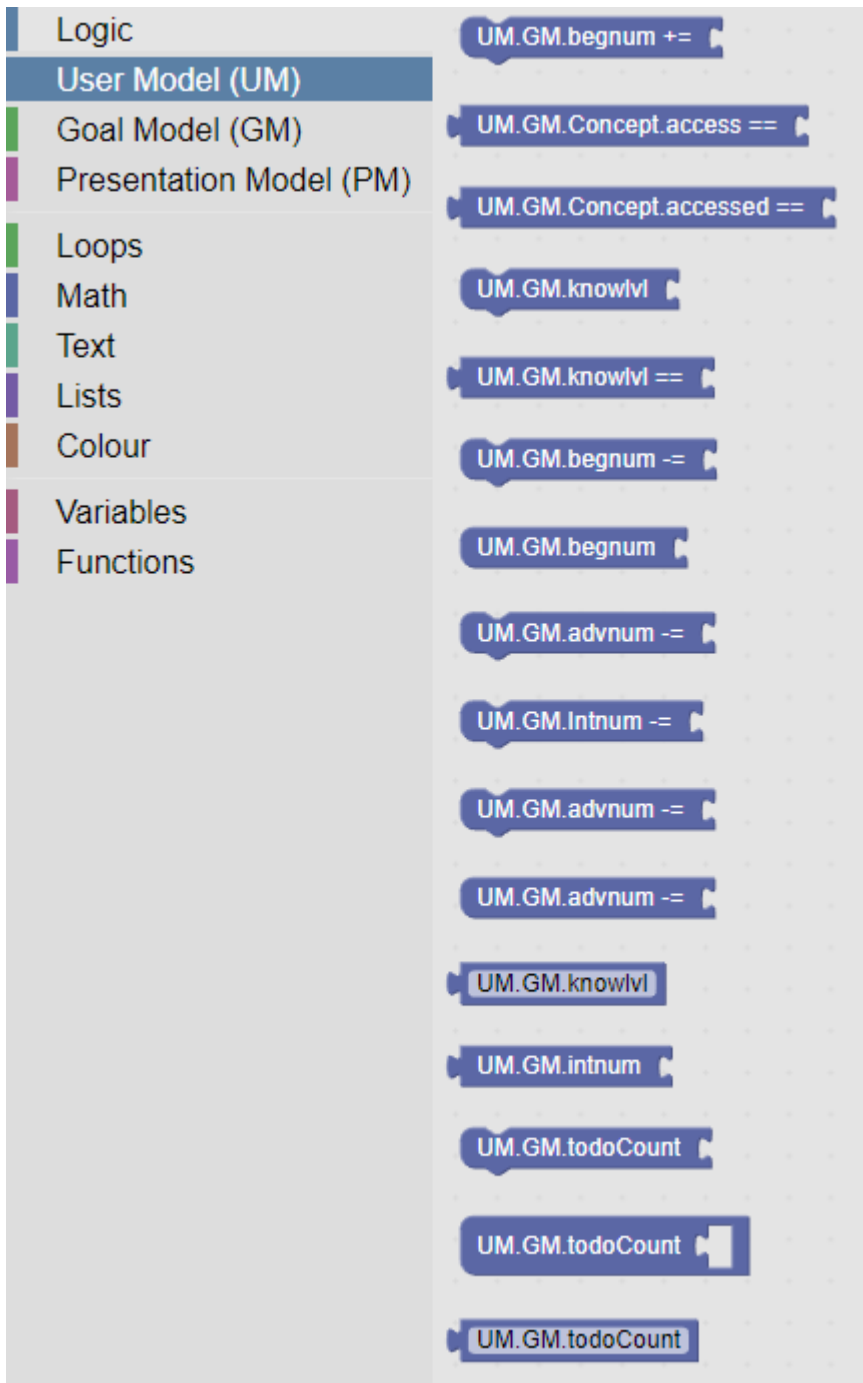


Figure 40: VASE 2.0 User Model Blocks

Figure 41 shows VASE 2.0, when the Goal Model option was clicked in the left panel. Thus, in the right panel, various Goal Model-related constructs are available to be selected, colour-coded in the same way as the Goal Model (green).

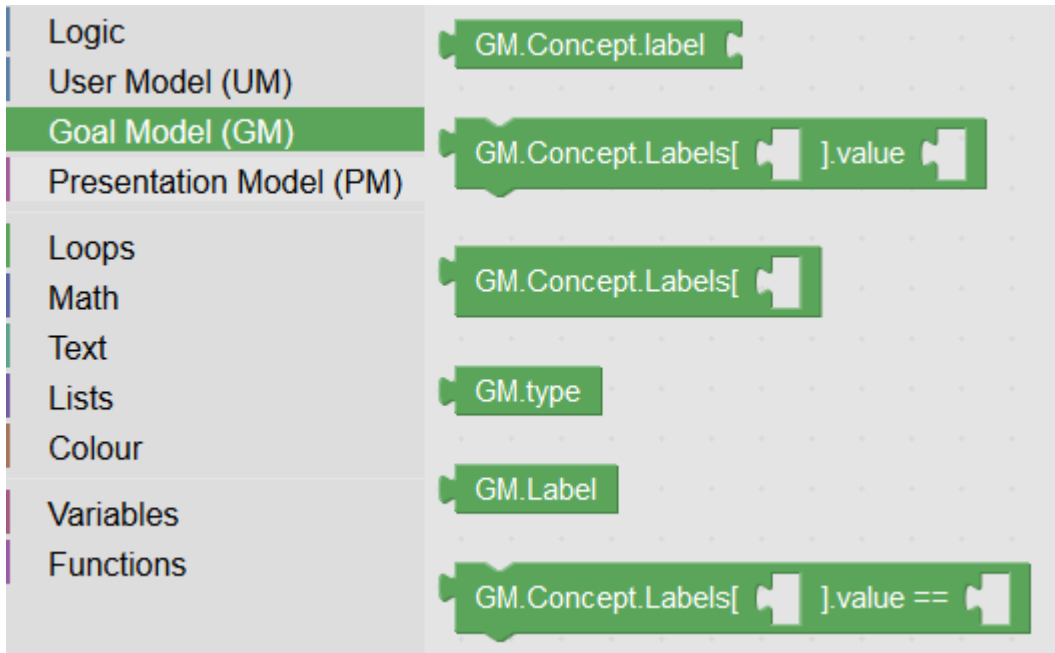


Figure 41: VASE 2.0 Goal Model Blocks

Next, Figure 42 and Figure 43 shows VASE 2.0, when the Presentation Model option was clicked in the left panel. In the right panel, various Presentation Model-related constructs are available to be selected, colour-coded in the same way as the Presentation Model (Purple). Presentation Model construct are used for presentation purposes. [59]

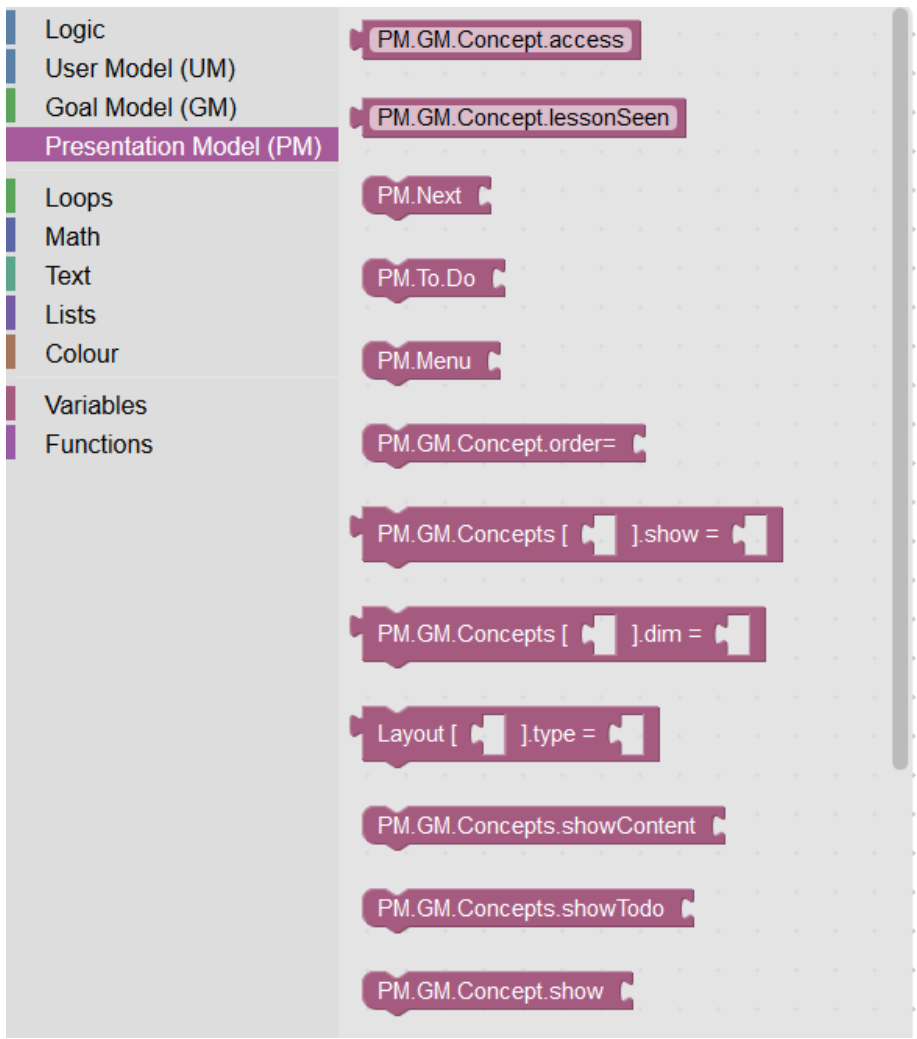


Figure 42: VASE 2.0 Presentation Model Blocks 1 of 2

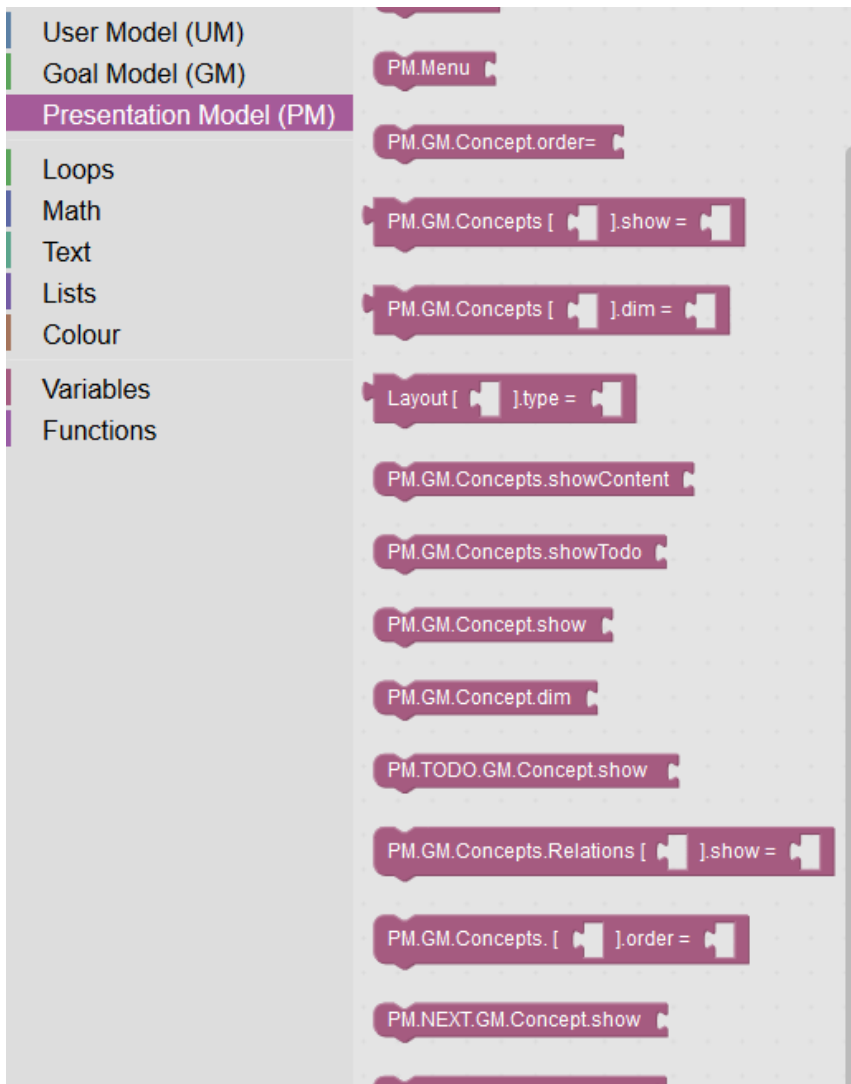


Figure 43: VASE 2.0 Presentation Blocks 2 of 2

Figure 44 shows VASE 2.0 first dynamic LAGBlock introduced in VASE 2.0 as VASE 1.0 did not offer dynamic LAGBlocks due to the limitation in the underlying framework. In VASE 1.0, this was not possible and such a condition was very cumbersome to represent. VASE 2.0 dynamic LAGBlock have the ability to change according to the user interaction with the available dynamic options.

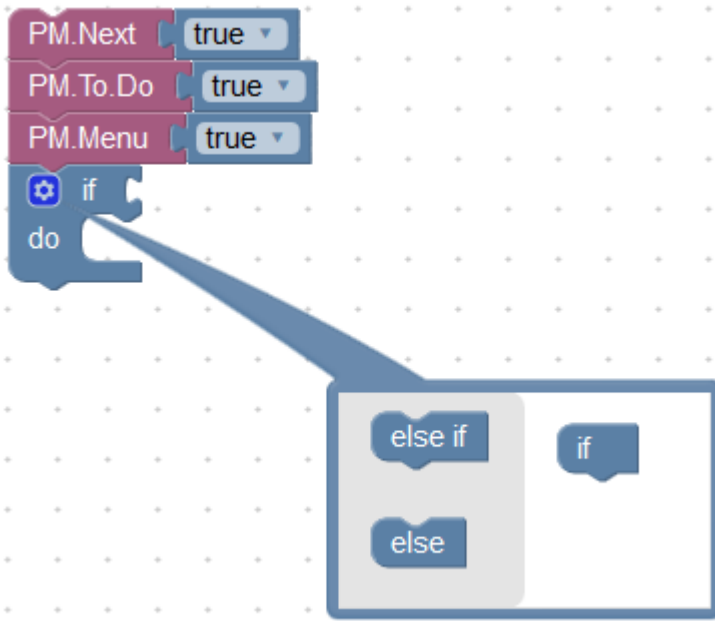


Figure 44: VASE 2.0 Dynamic LAGBlocks

Figure 45 shows that dynamic LAGBlock can have any number of dynamic options which could be selected by the user, making the LAGBlock change as per user choice.

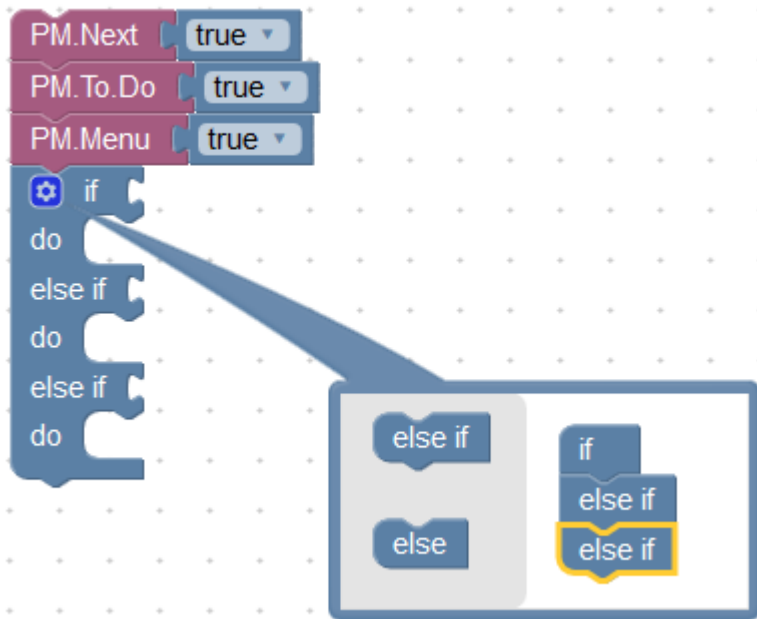


Figure 45: VASE 2.0 Dynamic LAGBlock addition

Figure 46 depicts additional LAGBlock options, for example any LAGBlock which contains other LAGBlocks could be copied and pasted where needed, its very useful to save time where previously used constructs are needed in an adaptive strategy. This feature was particularly important to the users as this was suggested by the users in the VASE 1.0 evaluation, see Table 4 for details.

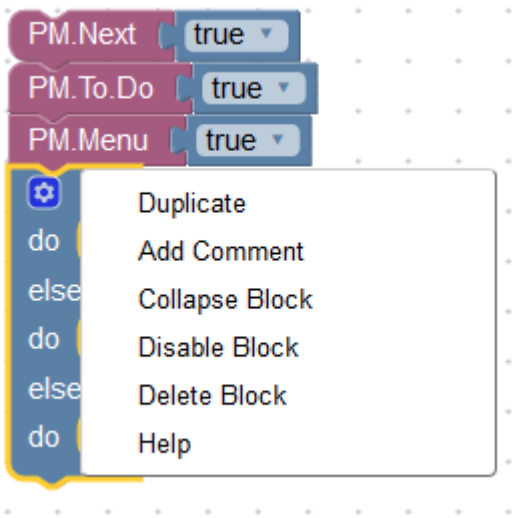


Figure 46: VASE 2.0 Additional LAGBlock options

Figure 47 shows another option to make adaptive strategy appear simple. For example, any LAGBlock could be *Collapsed* to hide the its content, making the strategy simpler to view, to help in working with large and unwieldy adaptive strategy. Comments can also be added to the block to attach extra meta-data with the block.

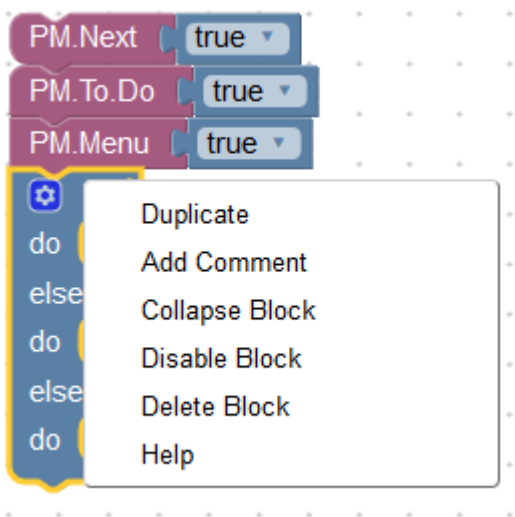


Figure 47: VASE 2.0 Collapse block option

Figure 48 shows the *collapsed* version of the LAGBlock shown in Figure 47. Once the LAGBlock has been collapsed, *Expand block* option will appear in the (right-click) menu. Once *expand block* is selected, the authoring environment will display the inner blocks as shown in Figure 47 . This feature was also suggested by the users to work with the large and complex adaptive strategies, please see Table 4 for details. Other LAGBlock options include disable and delete, as the name implies, *disable block* disables the block (greyed out) and the block code will not be executed, whereas the *delete option* deletes the block from the workspace.

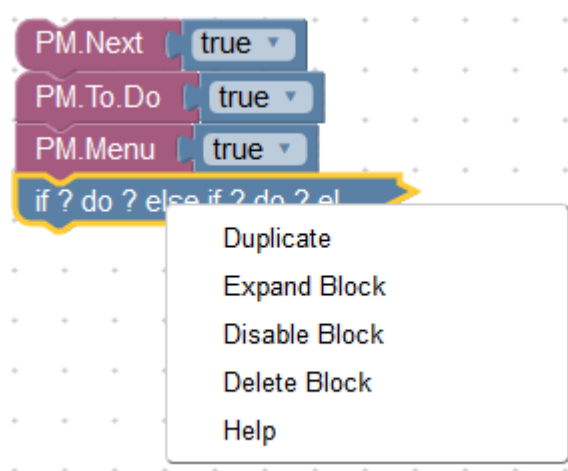


Figure 48: VASE 2.0 Expand Block option

Figure 49 shows a complete adaptive strategy in VASE 2.0, in this adaptive strategy the content within the course is labelled as 'beg', 'int', or 'adv' such that only the content labelled as 'beg' is shown to the beginners, 'beg'+ 'int' is shown to intermediates and 'beg'+ 'int'+ 'adv' is shown to advanced users. This is achieved by counting the number of 'beg' concepts and storing this in the 'begnum' User Module variable. This variable is decremented as the beginner content is accessed until all beginner concepts have been viewed. The intermediate content is displayed subsequently which has a variable called 'intnum' and the user is set to 'int'. All non 'beg'/'int'/'adv' labelled content is shown from the start, so a course would be displayed identically to the show All strategy, and those labels were not used.

```

PM.Next true
PM.To.Do true
PM.Menu true
For Each true
do
  if GM.Concept.label == "beg"
  do
    PM.GM.Concept.show true
    UM.GM.begnum += 1
  else if GM.Concept.label == "int"
  do
    PM.GM.Concept.show false
    UM.GM.begnum += 1
  else if GM.Concept.label == "adv"
  do
    PM.GM.Concept.show false
  else
    PM.GM.Concept.show true
  end
end
UM.GM.knowM "beg"

```

Figure 49: Beginner Intermediate and Advance strategy in VASE 2.0

Because every LAG strategy consists of two parts, *Initialization* and *Implementation*, new LAG constructs were created to be able to represent the *Initialization* and *Implementation* part of an adaptive strategy. This shows the usage of *Initialization* and *Implementation* LAGBlocks in an adaptive strategy.

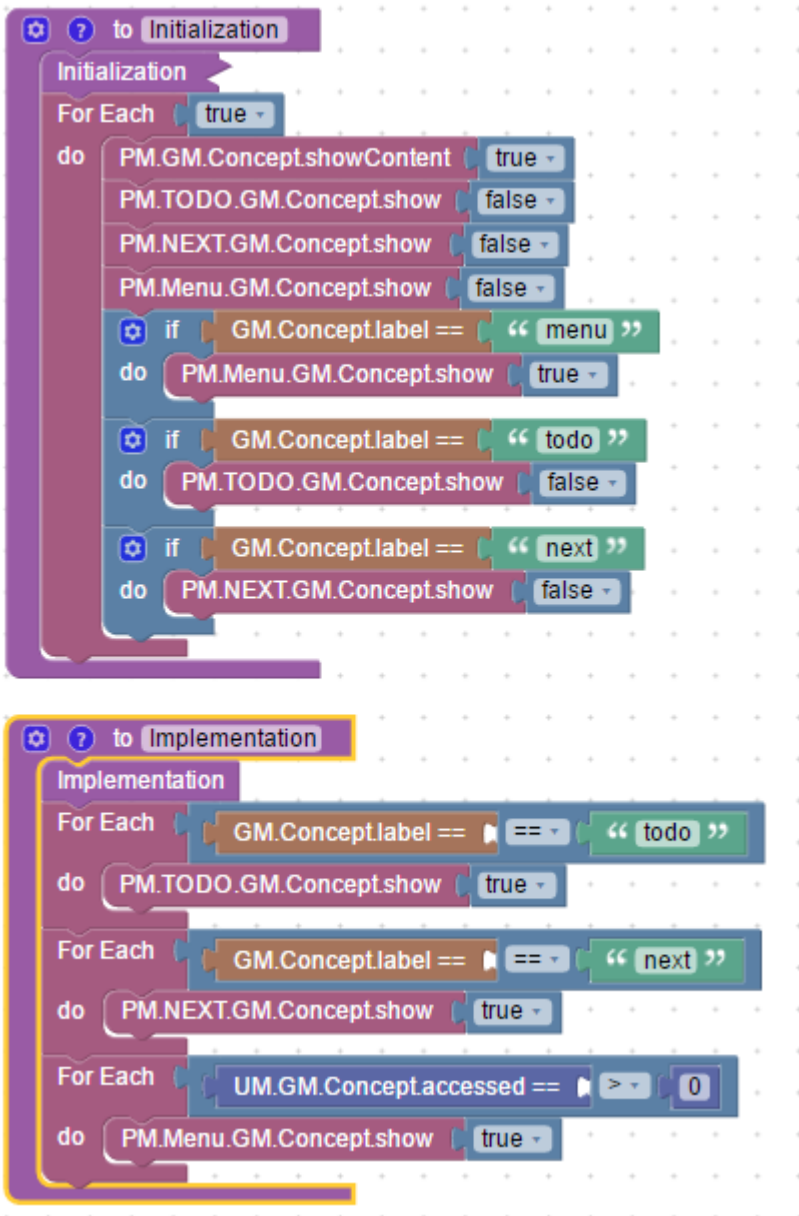


Figure 50: User interface screenshot of the working system (VASE 2.0)

As a result of the new design and implementation, VASE 2.0 should offer a substantial benefit over its predecessor (VASE 1.0), as it provides users with flexibility and access to the authoring environment, without installing any additional software. This has been further evaluated in Section 6.6.

5.5.3 LAG Converter: Converting Visual Representation to LAG syntax

Additionally to the above described implementations, to be able to run the code created in the visual environment, it was imperative to create a converter to a language which could be interpreted by an

adaptive hypermedia engine. The LAG language could be interpreted both by the AHA! Engine, as well as by ADE. Moreover, it had been used to inspire the LAGBlocks grammar. Thus, the conversion into the LAG language was selected and implemented. The place of the LAG converter in the overall VASE 2.0 system architecture can be seen in Figure 36.

From an implementational point of view, the conversion step traverses the graphical representation and generates the textual counterpart. This is achieved via parsing the graphical structure and using a lookup dictionary of language syntax. The visual framework follows an event-driven model and events are fired up at any changes to the visual workspace. This allows for triggering parse actions upon visual workspace changes, i.e., user adding new blocks or amending an existing block. The textual representation is further passed over to the LAG parser, for the generation of LAG strategies.

The next section draws the conclusions on the work presented in this chapter.

5.6 Conclusions

In this chapter, we have described the design and implementation of a tool aimed at weak programmers or non-programmers, to perform visual authoring of adaptive hypermedia. To render our visual extension more interoperable, the visual language definitions are stored in a XML file, which is then converted by our LAG converter, to create LAG code. This is to ensure that the VASE authoring tool can work with other tools which use LAG as the adaptation language, for instance, ADE (Adaptive Delivery Engine).

This research has proposed a visual programming paradigm for the LAG programming language, for creating adaptation specifications, by simply dragging and connecting visual elements. This allows the non-programming savvy author to create adaptation specifications, with little knowledge of LAG or any other programming language. This research has demonstrated that a visual extension for the LAG programming language is feasible. In the following chapter, we will show how we have continuously evaluated the output of this research, starting with the design phase, and then the two versions of the visual authoring system, VASE 1,0 and 2.0.

6 Evaluation of VASE: the Visual Adaptive Strategy Environment

6.1 Introduction

This Chapter evaluates the products of the research in a structured way, in order to address the research questions and objectives proposed in Chapter 1. It also tests a set of hypotheses in relation to the research questions. Overall, four case studies have been conducted, to attain further evidence on the indicated results gathered from the previous chapters. In particular, the following objectives are addressed in this chapter:

Objective 1 – *Conduct extensive theoretical background research into the area, to find the problems faced by the authors in adaptive strategy creation and to propose a theoretical framework to improve adaptive authoring for authors with different needs.*

The first part of this objective has been addressed in chapter 2 which summarises how authoring remains a bottleneck in adaptive hypermedia and the main problems faced by the author. In this chapter, the second part of this objective is addressed, analysing the features needed to introduce a theoretical framework to help the author with different needs.

Objective 2 - *Get feedback from users for their preferred way of authoring an adaptive strategy.*

Previously developed authoring tools for creating adaptive specifications were mostly aimed at the programming-savvy author; this research attempts to create an adaptive authoring tool which is suitable for non-programming-savvy authors as well, i.e., for any author without extensive programming language experience or knowledge. It is believed that by minimising the problems faced by authors in this context, it would encourage authors to create adaptive strategies. This would increase the uptake of adaptive authoring amongst the authors who do not have extensive programming knowledge.

Objective 5 - *Conduct a series of experiments that investigate the appropriate approach and features to design adaptive authoring system, and to test the practical development of newly added features in an adaptive authoring system, addressing the acceptance of visual form of adaptive authoring*

in the evaluations.

Outcome: the evaluations conducted in this chapter are based on the design and usage of the adaptive authoring system VASE, with its two iterations (VASE 1.0 and VASE 2.0). Importantly, evaluations were performed with all stakeholders: non-programmers, academic teachers and domain specialists, adaptation programming experts (LAG experts) and the general public.

6.2 Hypotheses

The main hypotheses that are researched in this thesis are listed below, together with their respective research questions they attempt to answer.

RQ 1.1 How can an adaptive authoring system render adaptive strategy creation easy for authors with different needs e.g. knowledge and experience?

H1. User roles can be used to show appropriate authoring options suited for an author to avoid confusion and *cognitive overload* (RQ1.1).

RQ 1.2 - Are there benefits in using visual programming techniques to create adaptive specifications and if so, what would these benefits be?

H2. Novice authors (non-programming-savvy authors) will find it easier to create adaptation strategies by manipulating visual elements as opposed to writing adaptive programming language syntax (RQ1.2).

RQ 1.3 How can an adaptive authoring system assist authors in adaptive specification's correctness and completeness, as well as interoperability?

H3. Designing *adaptive specifications* visually will provide better adaptive specification correctness and completeness. Creating export functionality into generic adaptation languages (such as LAG) can help in delivering the adaptive course on various systems (e.g., AHA!, ADE). (RQ1.3).

RQ 1.4. How can the visual programming techniques and technologies be implemented, in order to enhance the adaptive authoring system and adaptation language, and thus provide a high level of effectiveness, efficiency and satisfaction amongst authors?

H4. Visual programming techniques can be implemented to enhance adaptive authoring systems and adaptation languages:

- a. By representing an adaptation language in a visual form to improve user-interaction;
- b. By lowering the programming barrier to create adaptive specifications when compared to text-based techniques;
- c. By increasing user satisfaction;
- d. By increasing adaptive specification reuse amongst authors in different circumstances.

6.3 Non-professional Programmers' Case Study

The first case study was conducted on **SurveyMonkey** with the help of 47 participants, who volunteered for the experiment. No explanation was given about any of the programming concepts or subject area. An initial evaluation was conducted, to find out if users preferred a visual way of describing adaptive specifications or if they preferred to write strategies in plain text, to find out if such a visual authoring tool is worthwhile developing.

An online questionnaire was created, to guide all participants through the evaluation process. To ensure that the participants would volunteer for this evaluation, the time spent performing it was deliberately limited (to around 10-15 minutes). This questionnaire consisted of category type and ordinal category type; this questionnaire and its results have been added in

APPENDIX E: Non-professional programmer evaluation results. The results show that most users, almost 60%, preferred the adaptive strategy to be visual, by using a visual programming technique, as opposed to the plain text LAG syntax. Users were asked to select their profession; some of the users were company workers, who did not have much programming language experience. These users also preferred the visual way of creating an adaptive strategy; this shows that the non-professional programmers preferred the visual way to create an adaptive strategy. In fact, the majority of the users had no authoring experience (over 60%), thus conforming to the target users for this research in this thesis. To increase the uptake of Adaptive Specification authoring, it was very important to establish if the authors without much knowledge of programming languages would prefer to create *Adaptive Specification* visually, as opposed to writing adaptive strategies in a text-based environment.

For the initial system design of VASE, participants were asked to select a preferred paradigm. The data collected from the study were used in the first iteration of the system implementation, VASE 1.0, as described in section 5.4. The main features implemented in the first iteration of VASE, as a result of the *non-professional programmer experiment* were:

- User-roles and functionality matrix (as also theoretically researched in Chapter 4), to show relevant authoring options to each author;
- Using block programming framework, to enable non-programming savvy authors to create adaptive specifications based on visual programming techniques;
- Representing LAG adaptation language grammar in OpenBlocks [39] framework to create visual blocks for the LAG language called LAGBlocks (as theoretically explored in Chapter 2, Section 2.9.1).

6.4 LAG Expert Group Case Study

The second case study was performed with the help of a small number of experts in the language selected as basis for this research: the LAG language. Thus, in order to understand the deeper research issues, two face-to-face sessions were arranged with LAG experts, one with the MOT 3.0 [14] (see also Chapter 2,

Section 0) creator and another one with the ADE [15] creator, who have been involved in Adaptive Hypermedia research for over 3 years at the time of the interview.

Feedback was primarily qualitative data, during the face-to-face session with the researcher who was working on the LAG adaptation language. Specific improvements to the LAG language were suggested (listed in APPENDIX C: LAG Extension), comments were noted regarding the types of useful and meaningful improvements which could be made to the VASE authoring tool from a technical perspective.

Below is the list of important points highlighted during these sessions.

6.4.1 Differences between GAT and MOT

GAT [33] (see also Chapter 2, Section 2.5.6) and MOT (together with PEAL, see also Chapter 2, Section 0) still represent the most advanced authoring tools for adaptive hypermedia. Thus, any new authoring tool would need to take into account their good and bad points, in order to ensure, at a minimum, the same level of expressivity, ease of use, etc., but, potentially, deal with any of their limitations found by previous research. The following differences were extracted from the discussion.

- One of the main differences is how properties are represented in each system. Properties of the course are kept within the goal model in MOT, whereas they are spread between domain model and strategy in GAT.
- In GAT, resources are linked to different web pages, with a template like isMoon etc. These templates contain GALE code for advanced functionality. This spreads the adaptation strategies into different files and at different levels. It should be available for very advanced users only.
- In GAT, the order in which the domain is shown can be introduced directly in the domain model, making domain reuse very difficult, if not impossible.
- Constraints exist in GAT (in principle) but are not implemented.
- User roles should be fine-tuned, different author roles are useful, both in terms of specialty (e.g., domain specialty versus pedagogical specialty) and in terms of difficulty (e.g., beginner, intermediate, advanced, expert).
- Sockets in GAT are represented as the pedagogical labels in the goal model in MOT.

Specifically, the following problems have been identified with GAT (Graphical Authoring Toolkit).

- The course creation process is really difficult for authors.

- GAT is very textual for a visual tool, it would be better to have icons for the different pedagogical relationships, which are easy to identify.
- To provide advanced functionality, users write GALE code with the XHTML files.
- Everything (concepts, relations and prerequisites) looks the same: it should be displayed differently, based on the type.
- It is not possible to set a User Model variable in GAT.
- Visualising pedagogic relations could be improved.
- Drag and drop is needed, to provide better interaction.
- Also not good in GAT is that adaptation based on content and adaptation based on visual aspects is in the same place (e.g., formatting PRTs and prerequisites are together).

The following advantages have been identified in GAT:

- XML and XSLT based output may be preferred by the programming savvy authors.
- Meta information level can be exchanged via CAM strategies.
- User roles exist, to control what's being displayed.
- Strategy List with description appears on mouse hover.
- Possibility to filter what is seen (e.g., show or not show relations between sockets; show or not show certain type of pedagogical relations),

As a result of analysing the above advantages and disadvantages in GAT, the following ideas for the new authoring tool emerged, in terms of the following desirable features.

- Drag and Drop LAG visual editor with intuitive interface.
- Visualising the strategies to give clues to the user.
- Limiting the complexity shown to the author according to their user roles.
- Strategy list to be selected easily.
- Not only prerequisites, but different types of pedagogical relations (e.g., beginner-intermediate-advanced is such a relation as well) need their own representations.
- The properties need to be kept separately, like a goal level property should be separate from a domain level property, so the strategies can be reused easily.
- Meta-information should be easily added via the interface.
- Goal level variables should be separate from user level attributes and at run-time should combine all of these variables, to generate the adaptable material.
- Different authors should be able to work on different parts of the course, to promote separation-of-concern.

- A better way to visualise relations and concepts is needed, which is more intuitive to the user.
- Ideally, a wizard component would be useful, to visually arrange LAG objects on the screen.

The ideas from these discussions, as well as the feedback from the first case study (Section 6.3) were used, together with the theoretical considerations in Chapter 2, and based on the implementation methodology, as described in Chapter 3, Section 3.2, were used to create the first version of the visual block authoring system for adaptive hypermedia (further described in Chapter 5).

6.5 VASE 1.0 evaluation with University lecturers with eLearning experience

In order to better understand the university lecturer's take on the authoring tool implemented, a third case study was performed. The target was to investigate:

- How do professional lecturers feel about the two different paradigms (visual versus textual) of an adaptive authoring system?
- What improvements can be made from the authoring perspective?
- How does an adaptive authoring system compare with the current method they employed in course creation?

6.5.1 Experimental Setting

Face-to-face interviews were conducted as a part of this evaluation. Participants were lecturers at UK Universities, with over 5 years of teaching experience and with course creation experience. Prior appointments were made, to conduct the face-to-face interviews at convenient locations for the participants.

A short introduction was given, to explain how the adaptive course was to be built, i.e. how the static part and the dynamic part of adaptive course work together.

Each participant was shown 3 LAG adaptive strategies in the PEAL and VASE 1.0 authoring systems (listed below). Participants selected the system they would like to use first and author association with either of the systems was not disclosed, to avoid any preferential bias.

Each participant was asked to do the following in each authoring system (PEAL and VASE 1.0):

- **View** the '*Beginner-intermediate-advance*' adaptive strategy in each authoring system;

- **Create** an ‘*End-of-course-message*’ adaptive strategy in PEAL and VASE;
- **Modify** the ‘*Dimming*’ Strategy in VASE and PEAL.

After performing the above tasks, each participant was asked a list of questions in a semi-structured interview, to explore their feelings and preferences about each authoring tool and about course creation in adaptive hypermedia systems. Their notes were recorded on paper, after conducting the open discussion. As the discussions were focused towards adaptive authoring to create adaptive courses, the main areas to collect information were:

- *Usability*
- *Flexibility*
- *Completeness*
- *Correctness*
- *Compactness*

The way these keywords appeared in the discussion is shown in Table 3 below, as the types of suggestions or improvements collected from the participants. All the raw data had been independently verified by the researcher, no raw data had been stored, the suggestions in the table had been compiled during the interviews. The suggestions in the table had been further interpreted by the researcher, in order to generate the desired features for the VASE 2.0 authoring system.

Table 3: Raw Data Collected from Users and their interpretations

| Users suggested following types based on the raw data | Interpretations of Suggestion |
|--|---|
| Users suggested having <i>copy</i> and <i>paste</i> function blocks. | The researcher proposed a copy and paste feature to be added to the right-click of each block, to copy the selected block along with all the children blocks within that block. <i>Flexibility</i> |

| | |
|---|--|
| Users also suggested having <i>collapse</i> function for blocks. | The researcher proposed a collapse and un-collapse feature to be added to the right-click of each block and its child blocks. <i>Compactness</i> |
| Users suggested having <i>toggle user-friendly labels</i> function. | The researcher proposed a button to show user-friendly labels instead of PM.GM it should something more meaningful according to the user-role. <i>Usability</i> |
| Users suggested having auto-complete feature for blocks. | The researcher proposed an auto-complete to automatically fill-in the required blocks. <i>Completeness</i> |
| Users suggested having no two different connectors shape shouldn't connect. | The researcher proposed type checking before any block is connected together. <i>Correctness</i> |

The qualitative results presented above were transformed into a new set of requirements; these requirements were added to the list of requirements highlighted during the LAG expert's session (Section 6.4).

The table below (Table 4) summarises the final list of requirements. The requirements were based on the information collected from the users and the interpretation of the data, these were categorised into functionality and usability requirements. The requirements that are listed in Table 4 were to be included within the next development iteration of the VASE authoring tool (VASE 2.0).

6.5.2 Results and refined requirement list

The focal point of the experiment was to generate qualitative data, to be interpreted by the researcher. Thus, the aim of this evaluation was to retrieve real user feedback to do a like-for-like comparison with existing authoring tools (for this PEAL was chosen, as the authoring system for comparison, as previously

explained). This evaluation concentrated on the functionality and usability characteristics the PEAL and VASE authoring systems. Below is a list of most important requirements to be included in the development of the new authoring tool, VASE 2.0.

Table 4: Authoring System Requirements

| <i>No</i> | Feature Details |
|------------|---|
| <i>R1</i> | An adaptive authoring tool is to be able to implement the creation of adaptive behaviour. However, this represents the minimal feature and an authoring tool has to perform this feature in order to be called adaptive authoring tool. |
| <i>R2</i> | To be able to create LAG strategy by manipulating visual elements. Drag and Drop LAG constructs with snap-in functionality for intuitive user-interface (this will require minimum user training). |
| <i>R3</i> | Authoring tool should support user-roles and System [8]; user-roles are not supported by the current authoring tools such as PEAL. |
| <i>R4</i> | Validate LAG specification, i.e. error like PM.PM should be picked up during validation. |
| <i>R5</i> | Programmers or advanced users should be given access to a programming language editor and novice users could be given a visual code designer. |
| <i>R6</i> | Different authors should be able to work on different parts of the course to promote separation-of-concern. |
| <i>R7</i> | Authoring system should be able to work without installing any additional piece of software or plugin. |
| <i>R8</i> | Authoring system should be able to copy and paste blocks with its children (all the blocks which are connected under the block). |
| <i>R9</i> | Authoring system should offer a way to make the large and complex authoring strategy appear simple by collapsing the block with its children. |
| <i>R10</i> | Authoring system should prevent users from creating invalid adaptive specification. |
| <i>R11</i> | Authoring system should offer auto-complete, where applicable to automate tasks for the author. |

6.5.3 Discussions and Conclusions on the VASE 1.0 Experiment

The initial feedback was that nearly all of the users preferred the visual approach. The initial results revealed that the system did offer benefits in adaptive creation and reuse. However, there were serious limitations additional to the suggestions made by the users in the interviews. For example, as VASE 1.0 was developed in the Java programming language, it ran VASE in a Java class file, which meant that a plugin needed to be installed, on the computer, to use it in a browser. This could cause problems in an academic setting, where lecturers may not have the access rights on the university provided computer to install new software. There were several other limitations which were discovered by the researcher following the lecturers' evaluation.

The qualitative outcome of the experiment resulted in a detailed requirement list, to be used in the second iteration of the VASE authoring system, focusing on a more systematic way on extracting a generic set of features for adaptive authoring, to be used by professional lectures.

6.6 The Large Scale Final Experiment: Evaluation of VASE 2.0

After taking into account all of the above feedback and implementing them in a tool, VASE 2.0 was ready to be tested on large numbers of users. This was performed with the help of Amazon Mechanical Turk.

A number of online crowdsourcing services have been developed, which connect individuals willing to perform online tasks with other people willing to pay for the work performed. One of the popular systems is Amazon's Mechanical Turk (AMT). AMT is useful for researchers, because it handles recruitment and payment in an automatic way. Essentially, there is a huge number of people who use AMT, making it an excellent way to circulate studies. There have been a number of reviews about using AMT for researchers [59]. Amazon Mechanical Turk service has been proven to be an effective tool for conducting survey research [60], [61].

In summary, the AMT service allows requesters to post *human intelligence tasks* (HITs) that workers can complete for a small amount of money. A HIT is a small unit of work that is usually completed in the

worker's Internet browser. Most common HITs include providing audio translation or feedback about a website. These tasks usually require some sort of human intelligence.

In the case of this research, I designed an HIT to participate and complete an adaptive authoring task. Amazon provides web-based tools for creating several different kinds of HITs. However, none of these was suitable for authoring adaptive courses. Alternatively, Amazon also provides a way for tasks to be completed on external web servers. The requester (in this case, researcher) hosted the task on an external website, capable of running two adaptive authoring systems side-by-side to conduct adaptive authoring tasks. Once the external website has been developed, it was used to interface with Amazon's service, so that workers who accept and complete tasks can get paid.

The total number of workers who participated in the task was 380, which is just very slightly under the ideal sample size of 383, regarding sample size calculation, please see Chapter 3, Section 3.4.6 for more details.

The first stage of the experiment was to ensure that good quality workers were chosen to conduct the experiment. Thus, the worker's requirement was set to be Master (*Amazon Mechanical Turk Masters*) to do the specified HIT. Amazon Mechanical Turk has a built-in technology which analyses worker performance, identifies high performing workers, and monitors their performance over time. Workers who have demonstrated excellence across a wide range of HITs are awarded the Masters Qualification. Masters must continue to pass Amazon's statistical monitoring to retain the Mechanical Turk Masters Qualifications.

In the second stage, the workers were asked:

- to use three LAG adaptive strategies listed below, available on <http://adaptive.dcs.warwick.ac.uk>; the contents of this URL is also included in APPENDIX G: ADE 0.2 and LAG 4.0 Demonstration Courses;
- to create two adaptive courses, the content of the course was also available on the URL.

The adaptive course contained all elements of an adaptive course. For example, the workers were able to view the contents to be used in a course and the adaptive strategy used to apply adaptive behaviour in a course. Workers could choose which authoring system to use. A mix of complex and simple adaptive strategies was used for this experiment:

- To view the ‘*Beginner-intermediate-advance*’ strategy in the PEAL & VASE authoring system, respectively;
- To create an ‘*End-of-course-message*’ strategy in PEAL and VASE;
- To modify the ‘*Dimming*’ Strategy in VASE and PEAL.

In the third and the final stage, workers were asked to answer a questionnaire, to assess their experience with the authoring systems. The main aim of the VASE authoring system is to address the incomplete and difficult process of adaptive strategy creations, in terms of (perceived) *incompleteness* and *difficulty* present during adaptive strategy creation.

One of the main areas to focus on was if the programming barrier was indeed lowered for authors, by allowing non-programming savvy author to create or build adaptive strategies by using visual puzzle pieces. VASE 2.0, the system they used, was the second version of the novel authoring software that has been used to conduct evaluations. It was based on the research findings and frameworks in the Adaptive Hypermedia literature, as well on the research evaluations conducted previously in this research, described earlier in this Chapter. The detailed design and development of the authoring system and its various iterations has been described in Chapter 5.

The questions presented in the questionnaire (see Table 5) have an exploratory nature; the connection with the research questions is shown in the table.

Table 5: Questionnaire mapping onto the research questions

| Questions and answer range | Related research question |
|----------------------------|---------------------------|
|----------------------------|---------------------------|

| | |
|--|-------|
| Q1: <i>I think the system displayed relevant options according to my experience with the system</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.1 |
| Q2: <i>I would prefer to select the options I would like to see in the authoring system</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.1 |
| Q3: <i>I think the system displayed too many authoring options</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.1 |
| Q4: <i>I was satisfied with the authoring system features presented to me</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q5: <i>I think, the authoring system is easy to learn to use.</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q6: <i>Imagine you had created your own domain content and you wanted to reuse somebody else's adaptive strategy. Would you prefer use the authoring system?</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q7: <i>I think the authoring system can be used to create adaptive strategies without much knowledge about LAG programming language.</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q8: <i>I think the authoring system provided a fun way to create adaptive strategy.</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q9: <i>I think the authoring system provided a useful environment to create new adaptive strategies.</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q10: <i>I think the authoring system provided a better way to create adaptive strategies.</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.3 |
| Q11: <i>I think the authoring system provided a simpler way to view adaptive strategies.</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q12: <i>I found the authoring system to be highly effective in authoring of adaptive strategies.</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q13: <i>I think the authoring system encouraged the reuse existing adaptive strategies with other users.</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q14: <i>I would like to use the authoring system again</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| Q15: <i>I would like to spend more time with the authoring system</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.4 |

| | |
|---|-------|
| <i>Q16: I would imagine that most people will learn to use the authoring system quickly</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.4 |
| <i>Q17: I found the authoring system not very complex to use</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.4 |
| <i>Q18: I think the authoring system provides a quicker way to create adaptive strategies</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.4 |
| <i>Q19: Regarding the overall authoring experience, I was satisfied with the authoring system</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.4 |
| <i>Q20: I would imagine that most people will prefer to interact with the authoring system</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.4 |
| <i>Q21: I found the authoring system easy to share adaptive strategy with other users</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.1 |
| <i>Q22: I would prefer to use the authoring system to create large and complex adaptive programs</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.2 |
| <i>Q23: I think the authoring system helps to minimise errors in adaptive strategies</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.3 |
| <i>Q24: I think the authoring system helps to create complete adaptive strategies</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.3 |
| <i>Q25: I found the authoring system prevented me to create incorrect/invalid adaptive strategies</i> (Strongly disagree, disagree, neither agree nor disagree, agree, strongly agree) | RQ1.3 |

The final large-scale experiment conducted on AMT produced mainly quantitative results. The results from the questionnaire are discussed, alongside the hypotheses in the next section.

6.6.1 Quantitative Results: VASE 2.0 versus PEAL

The questionnaire was focused towards uncovering the issues relating to the adaptive authoring process for comparing the two systems, VASE 2.0 and PEAL. The questionnaire had a total of 25 questions, which used answers on a Likert scale, from 1 to 5 (1 = strongly disagree, 2 = disagree, 3 = Neither agree nor disagree, 4 = agree and 5 = strongly agree). The questions were divided into the following groups.

- *Usability Questions*: The questions focused towards guidelines for designing usability related concepts; for instance, usage in context, user interface and user-interaction.
- *Usefulness Questions*: These questions were aimed at measuring authoring system features in terms of correctness, completeness and other functional features, such as user roles etc.

In order to establish which significance test to use, a Pearson-Chi squared test was performed for each question for each of the two systems, VASE 2.0 and PEAL. Table 6 shows that neither the answers to questions on VASE 2.0, nor the answers to questions on PEAL were normally distributed. Thus, the T-test may not be enough to establish the significance of the answers. For this reason, I have computed significances for all the questions and categories also with the Wilcoxon signed-rank test.

Table 6: Normality of Answers to Questionnaire for VASE 2.0 and PEAL

| Question | VASE 2.0 | PEAL |
|--|-------------------|-------------------|
| <i>Q1: I think the system displayed relevant options according to my experience with the system</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q2: I would prefer to select the options I would like to see in the authoring system</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q3: I think the system displayed too many authoring options</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q4: I was satisfied with the authoring system features presented to me</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q5: I think, the authoring system is easy to learn to use.</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q6: Imagine you had created your own domain content and you wanted to reuse somebody else's adaptive strategy. Would you prefer use the authoring system?</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q7: I think the authoring system can be used to create adaptive strategies without much knowledge about LAG programming language.</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q8: I think the authoring system provided a fun way to create adaptive strategy.</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q9: I think the authoring system provided a useful environment to create new adaptive strategies.</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |

| | | |
|--|-------------------|-------------------|
| <i>Q10: I think the authoring system provided a better way to create adaptive strategies.</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q11: I think the authoring system provided a simpler way to view adaptive strategies.</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q12: I found the authoring system to be highly effective in authoring of adaptive strategies.</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q13: I think the authoring system encouraged the reuse existing adaptive strategies with other users.</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q14: I would like to use the authoring system again</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q15: I would like to spend more time with the authoring system</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q16: I would imagine that most people will learn to use the authoring system quickly</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q17: I found the authoring system not very complex to use</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q18: I think the authoring system provides a quicker way to create adaptive strategies</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q19: Regarding the overall authoring experience, I was satisfied with the authoring system</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q20: I would imagine that most people will prefer to interact with the authoring system</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q21: I found the authoring system easy to share adaptive strategy with other users</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q22: I would prefer to use the authoring system to create large and complex adaptive programs</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q23: I think the authoring system helps to minimise errors in adaptive strategies</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q24: I think the authoring system helps to create complete adaptive strategies</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |
| <i>Q25: I found the authoring system prevented me to create incorrect/invalid adaptive strategies</i> | p-value < 2.2e-16 | p-value < 2.2e-16 |

Due to the fact that I have computed significance for 25 questions, Bonferroni corrections needed applied to these. Thus, instead of using the usually applied threshold of $p < 0.5$ to establish significance of a comparison, $p' = 0.5/25 = 0.002$ is the threshold that needs to be applied. For individual categories, the

threshold is somewhat higher, indirectly proportional to the number of questions used to address that category (see also below in Tables Table 7 to Table 18 how questions map to categories), as follows:

- User role: $p_u=0.05/4 = 0.0125$
- Benefits: $p_b=0.05/9 = 0.0056$
- Usability: $p_{us}=0.05/9 = 0.0056$
- Quality: $p_q=0.05/3 = 0.0167$

Further, I analyse each hypothesis one by one.

Hypothesis 1 states “*User roles* can be used to show relevant authoring options specific to the author’s expectations with the authoring system to avoid *confusion* and *cognitive overload*”. This hypothesis addresses the users’ perceived usefulness of user-roles. It investigates if the use of user-roles will lead to users getting less confused, as the options displayed to the user should be adapted according to their knowledge and experience and will not lead to cognitive overload for users. This represents a more customised and personalised view of the system for the user. These options would change over time, since a system administrator could change the *user role* after certain criteria are met by the user, e.g., when a user has conducted a certain task successfully or spent a certain time within the system, e.g., created an adaptive strategy or an used existing adaptive strategy. The system can then use this new information about the user and update the user profile, to display relevant options to the user. Currently, this change is not automatic and system administrators will update the *user role* manually by using a functionality matrix (see Chapter 4). One set of results for hypothesis H1 with respect to the *user role* are shown in Table 7. The table shows the questions related to H1, and, for each question, the mean

answer, the mode, the standard deviation (SD), and the results of the t-test and Wilcoxon test³ in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_u = 0.0125$), when compared to the ‘indifferent’ user, who would answer 3 for each question (on a scale from 1 to 5).

Table 7: VASE 2.0 Results for questions related to *user role* for H1

| No | Hypothesis 1, N=190 | Mean | Mode | SD | t-test | Wilcoxon Rank sum |
|----|--|--------------|----------|--------------|--------------|-------------------|
| 1 | <i>I think the system displayed relevant options according to my experience with the system.</i> | 4 | 4 | 0.609 | 0.000 | 0.000 |
| 2 | <i>I would prefer to select the options I would like to see in the authoring system.</i> | 3.915 | 4 | 0.611 | 0.000 | 0.000 |
| 3 | <i>I think the system display too many authoring options.</i> | 3.084 | 4 | 1.187 | 0.000 | 0.000 |
| 4 | <i>I was satisfied with the authoring system features presented to me.</i> | 3.731 | 4 | 0.702 | 0.000 | 0.000 |
| | Overall | 3.682 | 4 | 0.777 | 0.000 | 0.000 |

Users indicated that the VASE authoring system displayed relevant options, with a mean value of (3.9 ± 0.609) with a mode of 4, the mode indicating the most frequent answer being positive.

³ The Wilcoxon test suffices, as it is stricter than the t-test; the results were provided for both in this table, for this category, and all other categories, for comparison only.

Users preferred to choose the options they would like to see in the authoring system, with a mean value of (3.9 ± 0.611) with mode of 4. This was the highest mean amongst the questions regarding the user roles. This could mean that the user preferred to be able to control the options displayed to them. Users did agree with the statement that the system display too many authoring options, with a mean value of (3.084 ± 1.187) with mode of 4. For the same question, the average value for PEAL is 2.379 with mode of 2, a difference of 0.705 between VASE and PEAL (see Table 8). This meant that users may have agreed with the number of options available in the VASE implementation for their respective *user role*.

Largely, the users were satisfied with the features that were available to them. Whilst not by a huge majority, they did answer positively, with a mean value of (3.73 ± 0.702) . For the questions related to the *user roles*, the total average value for PEAL is 3.04, in comparison, the total average value for VASE is 3.616, a difference of 0.576 between VASE and PEAL. The total average value for VASE is believed to be credibly positive. From a statistical significance perspective, the probability value p for each of the question is much lower than 0.05 (based on the t-test and Wilcoxon test as well). All the results are thus statically significant.

Table 8 shows one set of results on the questions related to the user role and H1 for PEAL. The table shows the questions related to H1, and, for each question, the mean answer, the mode, the standard deviation (SD), and the results of the t-test and Wilcoxon test⁴ in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_u = 0.0125$), when compared to the ‘indifferent’ user, who would answer 3 for each question (on a scale from 1 to 5).

⁴ The Wilcoxon test suffices, as it is stricter than the t-test; the results were provided for both in this table, for this category, and all other categories, for comparison only.

Table 8: PEAL Results for questions related to *user role* for H1

| No | Hypothesis 1, N=190 | Mean | Mode | SD | Wilcoxon Rank sum | t-test |
|----|---|--------------|------------|--------------|-------------------|--------|
| 1 | <i>I think the system displayed relevant options according to my experience with the system</i> | 2.326 | 2 | 0.975 | 0.000 | 0.000 |
| 2 | <i>I would prefer to select the options I would like to see in the authoring system</i> | 3.336 | 3 | 1.009 | 0.000 | 0.000 |
| 3 | <i>I think the system display too many authoring options</i> | 2.379 | 2 | 0.993 | 0.000 | 0.000 |
| 4 | <i>I was satisfied with the authoring system features presented to me</i> | 2.878 | 3 | 1.380 | 0.000 | 0.000 |
| | Overall | 2.730 | 2.5 | 1.089 | 0.000 | 0.000 |

The results from the Table 8 above show that for features implemented in PEAL which were related to the *user roles*, the users expressed less positive feedback, possibly because *user roles* were not implemented, to show relevant options to the users, according to the *user role*. All results are statistically significant, even at the (Bonferroni-) corrected $p < p_u = 0.0125$. In fact, PEAL did not have any notion of a functionality matrix, thus the total score is under 3. Table 8 shows that the majority of the users thought that the PEAL authoring system displayed too many options, as the opposite question was rated as $2.379 < 3$. In contrast to VASE, the same question in Table 7, has been rated higher by the users (rated as 3.084), which meant that users preferred the implementation of the *user role* in VASE in comparison with the PEAL authoring system.

Users further indicated that the PEAL authoring system did not display relevant options, with a mean value of (2.326 ± 0.975) with a mode of 2, the mode indicating the most frequent answer as being negative. Users preferred to choose the options they would like to see in the authoring system, with a mean value of (3.336 ± 1.009) with mode of 3. In contrast, the users were not satisfied with the features which were displayed to them, with a mean value of (2.878 ± 1.38) .

The overall results about questions relating to the *user role* as shown in Table 9. The total average value for PEAL is 2.729, in comparison with the total average value for VASE is 3.682, a difference of 0.953 between VASE and PEAL. This difference is statistically significant, at the (Bonferroni-)corrected $p < p_u = 0.0125$. This meant that the majority of the users preferred *user role* implementation of VASE instead of PEAL authoring system.

Table 9: PEAL versus VASE overall results for the questions related to *user roles* (H1)

| Questions related to <i>user roles</i> | PEAL average | VASE average | Wilcoxon Rank sum |
|---|-----------------|-----------------|----------------------|
| <i>Q1: I think the system displayed relevant options according to my experience with the system</i> | 2.326 | 4 | 0.000 |
| <i>Q2: I would prefer to select the options I would like to see in the authoring system</i> | 3.337 | 3.915 | 0.000 |
| <i>Q3: I think the system didn't display too many authoring options</i> | 2.379 | 3.084 | 0.000 |
| <i>Q4: I was satisfied with the authoring system features presented to me</i> | 2.879 | 3.731 | 0.000 |
| Total | 2.730 | 3.682 | 0.000 |

Hypothesis 2 targets the *benefit* of using visual programming techniques to create adaptive specifications. Here, I concretely translated this in terms of *shallow learning curve*, *higher user satisfaction* and *less error-proneness*, as follows:

- *Shallower learning curve* - By lowering programming barriers to create Adaptive specifications when compare to text-based techniques;
- *Higher user satisfaction* - By representing adaptation languages visually to improve user-interaction and tracking;
- *Less error-prone* - users will make fewer errors in adaptive specification creation and reuse.

One set of results to the questions related to the *benefits* of the authoring paradigm used in VASE 2.0 are listed in **Table 10**. The table shows the questions related to H2, and, for each question, the mean answer, the mode, the standard deviation (SD), and the results of the t-test and Wilcoxon test⁵ in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_b = 0.0056$), when compared to the ‘indifferent’ user, who would answer 3 for each question (on a scale from 1 to 5). The difference is significant.

Table 10: VASE 2.0 Results for *Benefits* questions for H2

| | Hypothesis 2, N=190 | Mean | Mode | SD | t-test | Wilcoxon Rank sum |
|---|--|-------------|-------------|-----------|---------------|--------------------------|
| 1 | <i>I think the authoring system is easy to learn to use.</i> | 3.663 | 4 | 1.024 | 0.000 | 0.000 |
| 2 | <i>Imagine you had created your own domain content and you wanted to reuse somebody else's adaptive strategy. Would you prefer to use this authoring system?</i> | 2.710 | 3 | 0.984 | 0.000 | 0.000 |
| 3 | <i>I think the authoring system can be used to create adaptive strategies without much knowledge about LAG programming language.</i> | 3.821 | 4 | 0.712 | 0.000 | 0.000 |
| 4 | <i>I think the authoring system provided a fun way to create adaptive strategy.</i> | 4.221 | 4 | 0.700 | 0.000 | 0.000 |
| 5 | <i>I think the authoring system provided a useful environment to create new adaptive strategies.</i> | 3.721 | 3 | 0.756 | 0.000 | 0.000 |
| 6 | <i>I think the authoring system provided a better way to create adaptive strategies.</i> | 3.773 | 3 | 0.746 | 0.000 | 0.000 |
| 7 | <i>I think the authoring system provided a simpler way to view adaptive strategies.</i> | 3.926 | 4 | 0.670 | 0.000 | 0.000 |
| 8 | <i>I found the authoring system to be highly effective in authoring of adaptive strategies.</i> | 3.710 | 3 | 0.716 | 0.000 | 0.000 |

⁵ The Wilcoxon test suffices, as it is stricter than the t-test; the results were provided for both in this table, for this category, and all other categories, for comparison only.

| | | | | | | |
|---|---|--------------|-------------|--------------|-------------|-------------|
| 9 | <i>I think the authoring system encouraged the reuse existing adaptive strategies with other users.</i> | 3.805 | 3 | 0.769 | 0.000 | 0.000 |
| | Overall | 3.705 | 3.44 | 0.786 | 0.00 | 0.00 |

Most users thought that the VASE 2.0 authoring system was easy to learn to use, with a mean value of (3.66 ± 1.02) with mode of 4. This meant that the majority of the users thought that it was easy to learn how the system works and were not confused; this would be one of the benefits of the VASE 2.0 authoring system from the user's viewpoint.

One of the questions that scored low (average below 3) compared to the other questions in Table 10, was the question 2, to use someone's adaptive program with their own domain content, with a mean value of (2.71 ± 0.984) with mode of 3. One of the possible explanations of such a low score could be that the users found it difficult to imagine the authoring process, for instance, how to use their domain content with someone's adaptive program. It might be because users were not clear on how to use these features in the VASE 2.0 authoring system. One of the ways this could be improved is by providing a wizard or step-by-step instructions to guide the users to use someone's adaptive strategy with their contents. It would be a good idea to explore more ways to make these features clearer for the users, as this would encourage users to reuse their domain content with an adaptive strategy written by someone else. This has to be handled with caution, since some adaptive strategies only work with domain content correctly labelled and may not work if the required labels are not present in the domain content. This may not be an easy concept for an author with basic knowledge of adaptive hypermedia systems.

A popular view amongst the users was that the VASE authoring system can be used to create adaptive strategies without much knowledge of the LAG programming language, with a mean value of (3.821 ± 0.712) with mode of 4. This was very important from this research perspective, because this was one of the reasons to introduce a visual programming paradigm for adaptive strategy authoring, to make adaptive strategy authoring accessible to users without much knowledge of adaptation programming languages. This has the added benefit that the adaptation language can be changed in the future, as it addresses the

separation of concern concept. The majority of the users responded positive to using a visual programming paradigm. VASE can be used to create/build adaptive strategies without much knowledge about the LAG adaptation language.

The aspect regarding the *higher user satisfaction* and being *less error-prone* as reflected in Questions 4,5,6,7,8 and 9 with mean values of (4.22 ± 0.7) , (3.721 ± 0.756) , (3.773 ± 0.746) , (3.926 ± 0.670) , (3.71 ± 0.716) and (3.805 ± 0.769) , respectively. The largest mean value is generated for the questions asking if the authoring system provided a fun way to create adaptive strategy and if the authoring system provided a simple way to create adaptive strategies. Overall, the results indicate the *benefits* of using visual programming techniques in adaptive strategy creation. The overall benefit mean value is (3.705 ± 0.786) .

To strengthen the validation of the proposed hypothesis, statistical tests were conducted for significance. Please see Chapter 3 for more details on the test that were conducted.

Table 11 reflects one set of results for the *Benefits* question for hypothesis H2 for the PEAL system. The table shows the questions related to H2, and, for each question, the mean answer, the mode, the standard deviation (SD), and the results of the t-test and Wilcoxon test⁶ in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_b = 0.0056$), when compared to the ‘indifferent’ user, who would answer 3 for each question (on a scale from 1 to 5). Most differences are significant, with two exceptions, discussed below.

⁶ The Wilcoxon test suffices, as it is stricter than the t-test; the results were provided for both in this table, for this category, and all other categories, for comparison only.

Table 11: PEAL Results for *Benefits* questions for H2

| No | Hypothesis 2, N=190 | Mean | Mode | SD | Wilcoxon Rank sum | t-test |
|----|---|---------------|--------------|----------------|-------------------|--------|
| 1 | <i>I think the authoring system is easy to learn to use.</i> | 2.878 | 2 | 1.305 | 0.000 | 0.000 |
| 2 | <i>Imagine you had created your own domain content and you wanted to reuse somebody else's adaptive strategy Would you prefer to use this authoring system?</i> | 2.957 | 3 | 1.364 | 0.454 | 0.000 |
| 3 | <i>I think the authoring system can be used to create adaptive strategies without much knowledge about LAG programming language.</i> | 2.852 | 1 | 1.436 | 0.000 | 0.000 |
| 4 | <i>I think the authoring system provided a fun way to create adaptive programs.</i> | 2.063 | 2 | 0.605 | 0.000 | 0.000 |
| 5 | <i>I think the authoring system provided an environment to try to create new adaptive strategies.</i> | 3.031 | 2 | 1.417 | 0.079 | 0.000 |
| 6 | <i>I think the authoring system provided a better way to create adaptive strategies.</i> | 2.742 | 3 | 1.273 | 0.000 | 0.000 |
| 7 | <i>I think the authoring system provided a simpler way to view adaptive strategies.</i> | 2.942 | 1 | 1.437 | 0.000 | 0.000 |
| 8 | <i>I found the authoring system to be highly effective in authoring of adaptive strategies.</i> | 2.731 | 4 | 1.184 | 0.000 | 0.000 |
| 9 | <i>I think the authoring system encouraged the reuse existing adaptive programs with other users.</i> | 3.068421 | 2 | 1.391795 | 0.000 | 0.000 |
| | Overall | 2.8076 | 2.222 | 1.26842 | 0.000 | 0.000 |

In contrast to VASE 2.0, most users thought that PEAL was not that easy to learn, with a mean value of (2.878± 1.305) with mode of 2. In Table 11, question 4 scored the lowest, with a mean value of (2.063 ± 0.605) with mode of 2. The majority of the users thought that PEAL is not an authoring system that provides a fun way of creating adaptive specification; perhaps this is why earlier PEAL evaluations reported that while using PEAL, the majority of the users were intimidated, which led to confusion.

Making an authoring environment fun for users will provide an enjoyable way for users to create adaptive strategies.

Question 2 in **Table 11** is the only question that scored higher in comparison with results in **Table 10**, with a mean value of (2.95 ± 1.3640) with mode of 3, (2.71 ± 0.984) with mode of 3 respectively. Although the results are not significant, the scores are not vastly different between VASE and PEAL for this question, and this was scored for both negatively (below the 'indifferent' 3). Thus, it would be a good idea to look into the details of how VASE could improve adaptive strategy sharing options, by making it easy to understand and follow; additionally, guidance must be provided to the user.

Aspects regarding higher *user satisfaction* and *less error-prone* were reflected in Questions 4,5,6,7,8 and 9 with mean values of (2.063 ± 0.605) , (3.031 ± 1.417) , (2.742 ± 1.273) , (2.942 ± 1.437) , (2.73 ± 1.184) and (3.06 ± 1.391) . Overall, the results indicate that PEAL was perceived to provide fewer *benefits* in adaptive strategy creation and reuse, in comparison with the VASE authoring system. In fact, the only one of the two questions answered positively for PEAL (above 3) is Question 5, which is not significant. The overall *benefit* mean value is (2.807 ± 1.26) , which is below 3.

Again, the results from all the significance tests match the results collected before. As said, all the results are statically significant, with the exception of Questions 2 and 5, which have probability value larger than 0.05 and than $p_b=0.05/9 = 0.0056$ (0.454 and 0.079, respectively). The problem raised by the answers to Question 2, with its highest frequency of the answers within the agreeing range, is that it indicates that users may have struggled to imagine how adaptive specification created by other authors can be used with content created by the user. From a statistical significance angle. For Question 5, the probability value (0.079) is also far from the acceptable threshold of $p_b=0.05/9 = 0.0056$. However, it's clear that some users felt that the PEAL authoring system did not provide a quick way to create new adaptive strategy.

The results for hypothesis H2 with respect to the *benefits* are shown in Table 12. The table shows the questions related to H2, and, for each question, the mean answer, the mode, the standard deviation (SD), and the results of the t-test and Wilcoxon test in terms of the value of the probability p (significant if $p < 0.05$ as

well as at $p < p_b = 0.05/9 = 0.0056$), when compared to the ‘indifferent’ user, who would answer 3 for each question (on a scale from 1 to 5). Overall, all scores on the questions on benefits are significantly lower for PEAL, with the exception of Question 5 (where PEAL is still lower, but not significantly so), and Question 2, which was extensively discussed above. It is however worth mentioning that overall, the benefits category, whilst lower in average for PEAL, is not significant with the Bonferroni correction (but would be significant at the 0.5 level). Clearly, Question 2 especially has an impact on the overall significance of the results, and would need further exploration.

Table 12: PEAL versus VASE Results for the questions related to *benefits* (H2)

| Questions related to <i>benefits</i> | PEAL MEAN | VASE MEAN | Wilcoxon Rank sum |
|--|--------------|--------------|----------------------|
| <i>Q1: I think, the authoring system is easy to learn to use.</i> | 2.879 | 3.663 | 0.000 |
| <i>Q2: Imagine you had created your own domain content and you wanted to reuse somebody else's adaptive strategy. Would you prefer to use this authoring system?</i> | 2.958 | 2.710 | 0.454 |
| <i>Q3: I think the authoring system can be used to create adaptive strategies without much knowledge about LAG programming language.</i> | 2.841 | 3.821 | 0.000 |
| <i>Q4: I think the authoring system provided a fun way to create adaptive strategy.</i> | 2.063 | 4.221 | 0.000 |
| <i>Q5: I think the authoring system provided a useful environment to create new adaptive strategies.</i> | 3.032 | 3.721 | 0.070 |
| <i>Q6: I think the authoring system provided a better way to create adaptive strategies.</i> | 2.742 | 3.773 | 0.000 |
| <i>Q7: I think the authoring system provided a simpler way to view adaptive strategies.</i> | 2.942 | 3.926 | 0.000 |

| | | | |
|---|-------------|-------------|-------------|
| <i>Q8: I found the authoring system to be highly effective in authoring of adaptive strategies.</i> | 2.732 | 3.710 | 0.000 |
| <i>Q9: I think the authoring system encouraged the reuse existing adaptive strategies with other users.</i> | 3.068 | 3.805 | 0.000 |
| Overall | 2.80 | 3.70 | 0.04 |

Hypothesis 3 states that designing Adaptive Specifications visually will provide better adaptive specification *correctness (A)* and *completeness (B)*.

The results from **Table 13** show the questions related to usability for VASE 2.0. The table shows the questions related to H3, and, for each question, the mean answer, the mode, the standard deviation (SD), and the results of the t-test and Wilcoxon test⁷ in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_{us} = 0.0056$), when compared to the ‘indifferent’ user, who would answer 3 for each question (on a scale from 1 to 5). These results are statistically significant, for both parametric and non-parametric tests. Overall, users reported that VASE 2.0 was usable, with a mean value of (3.418 ± 1.04) with mode of 3.

Table 13: VASE 2.0 Results for *usability* questions for H3

| No | Hypothesis 3, N=190 | Mean | Mode | SD | Wilcoxon | t-test |
|----|---|-------|------|-------|----------|--------|
| 1 | <i>I would like to use the authoring system again</i> | 3.663 | 4 | 0.619 | 0.000 | 0.000 |

⁷ The Wilcoxon test suffices, as it is stricter than the t-test; the results were provided for both in this table, for this category, and all other categories, for comparison only.

| | | | | | | |
|---|---|----------------|----------|---------------|-------|-------|
| 2 | <i>I would like to spend more time with the authoring system</i> | 3.294 | 3 | 0.990 | 0.000 | 0.000 |
| 3 | <i>I would imagine that most people will learn to use the authoring system quickly</i> | 3.484 | 3 | 1.016 | 0.000 | 0.000 |
| 4 | <i>I found the authoring system not very complex to use</i> | 3.984 | 4 | 0.612 | 0.000 | 0.000 |
| 5 | <i>I think the authoring system provides a quicker way to create adaptive strategies</i> | 2.894 | 1 | 1.454 | 0.000 | 0.000 |
| 6 | <i>Regarding the overall authoring experience, I was satisfied with the authoring system</i> | 3.768 | 4 | 0.958 | 0.000 | 0.000 |
| 7 | <i>I would imagine that most people will prefer to interact with the authoring system</i> | 3.273 | 2 | 1.172 | 0.000 | 0.000 |
| 8 | <i>I found the authoring system easy to share adaptive strategy with other users</i> | 2.931 | 2 | 1.436 | 0.000 | 0.000 |
| 9 | <i>I would prefer to use the authoring system to create large and complex adaptive programs</i> | 3.468421 | 4 | 1.148522 | 0.000 | 0.000 |
| | Overall | 3.41819 | 3 | 1.0456 | 0.000 | 0.000 |

This is implying that the majority of participants preferred the block programming paradigm employed by VASE 2.0 to create adaptive strategies.

The results in Table 14 reflect the answers to the questions related to the *usability* of the PEAL authoring system when compared to the same ‘indifferent’ user, who would answer 3 for each question (on a scale from 1 to 5). Specifically, they show the mean answer, the mode, the standard deviation (SD), and the results of the t-test and Wilcoxon test in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_{us} = 0.05/9 = 0.0056$).

Overall, users reported that PEAL was perceived to have lower usability, with mean value of (2.856 ± 1.367) mode of 2.4. The results from all the questions indicate statistical significance, as all the probability values p for all tests have a value starting with 0.00 ($p < p_{us} = 0.0056$), even for PEAL’s overall usability value (for the t-test and the Wilcoxon test). This is an indication that the difference to the indifferent user is statistically significant, for both parametric and non-parametric tests. With the exception of Questions 2 and 4, this means a negative result for PEAL. The overall usability result is also (significantly) below average.

Table 14: PEAL Results for *usability* questions for H3

| No | Hypothesis 3, N=180 | Mean | Mode | SD | Wilcoxon Rank sum | t-test |
|----|---|--------------|--------------|--------------|-------------------|--------------|
| 1 | <i>I would like to use the authoring system again</i> | 2.678 | 2 | 1.435 | 0.000 | 0.000 |
| 2 | <i>I would like to spend more time with the authoring system</i> | 3.084 | 4 | 1.411 | 0.000 | 0.000 |
| 3 | <i>I would imagine that most people will learn to use the authoring system quickly</i> | 2.831 | 1 | 1.422 | 0.000 | 0.000 |
| 4 | <i>I found the authoring system not very complex to use</i> | 3.021 | 3 | 1.383 | 0.000 | 0.000 |
| 5 | <i>I think the authoring system provides a quicker way to create adaptive strategies</i> | 2.963 | 1 | 1.463 | 0.000 | 0.000 |
| 6 | <i>Regarding the overall authoring experience, I was satisfied with the authoring system</i> | 2.894 | 4 | 1.341 | 0.000 | 0.000 |
| 7 | <i>I would imagine that most people will prefer to interact with the authoring system</i> | 2.631 | 2 | 1.372 | 0.000 | 0.000 |
| 8 | <i>I found the authoring system easy to share adaptive strategy with other users</i> | 2.868 | 4 | 1.144 | 0.000 | 0.000 |
| 9 | <i>I would prefer to use the authoring system to create large and complex adaptive programs</i> | 2.736 | 1 | 1.335 | 0.000 | 0.000 |
| | Overall | 2.856 | 2.444 | 1.367 | 0.000 | 0.000 |

The overall results for hypothesis H3 with respect to the *usability* are shown in Table 15. The table shows the comparison between the mean for PEAL and VASE, and its significance. For all questions on usability, with the exception of Question 5, PEAL has a lower score. Also for the majority, this score is significantly lower ($p < p_{us} = 0.0056$). For the exception Question 5, whilst PEAL is preferred in average, this is not significant. Questions 2 and 8 show a vague preference for VASE 2.0, but this difference is not significant. Clearly, the speed of authoring, and the sharing of strategies needs further exploration.

Table 15: PEAL versus VASE Results for the questions related to *usability* (H3)

| Questions related to <i>usability</i> | PEAL MEAN | VASE MEAN | Wilcoxon Rank sum |
|---|--------------|--------------|----------------------|
| <i>Q1: I would like to use the authoring system again</i> | 2.679 | 3.663 | 0.000 |
| <i>Q2: I would like to spend more time with the authoring system</i> | 3.084 | 3.294 | 0.454 |
| <i>Q3: I would imagine that most people will learn to use the authoring system quickly</i> | 2.832 | 3.484 | 0.000 |
| <i>Q4: I found the authoring system not very complex to use</i> | 3.021 | 3.984 | 0.000 |
| <i>Q5: I think the authoring system provides a quicker way to create adaptive strategies</i> | 2.963 | 2.894 | 0.794 |
| <i>Q6: Regarding the overall authoring experience, I was satisfied with the authoring system</i> | 2.895 | 3.768 | 0.000 |
| <i>Q7: I would imagine that most people will prefer to interact with the authoring system</i> | 2.632 | 3.273 | 0.000 |
| <i>Q8: I found the authoring system easy to share adaptive strategy with other users</i> | 2.868 | 2.931 | 0.8122 |
| <i>Q9: I would prefer to use the authoring system to create large and complex adaptive programs</i> | 2.737 | 3.468 | 0.000 |
| Overall | 2.85 | 3.4 | 0.000 |

Hypothesis 4 states that visual programming techniques can be implemented to enhance adaptive authoring systems and adaptation languages.

The results from Table 16 below show the questions related to the *Quality* of VASE 2.0, when compared to the ‘indifferent’ user, who would answer 3 for each question (on a scale from 1 to 5). The table shows the questions related to H4, and, for each question, the mean answer, the mode, the standard deviation

(SD), and the results of the t-test and Wilcoxon test⁸ in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_q = 0.0167$). *Quality* relates to the code quality of adaptive strategies created with minimum number of errors. Overall, users felt that VASE produced adaptive specifications with better code quality, with a mean value of (3.856 ± 0.956) and mode of 4. The results from all the questions indicate statistical significance, as all the probability values p for all tests have a value starting with 0.00 ($p < p_q = 0.0167$), even for VASE's quality average value (for the t-test and the Wilcoxon test). This is an indication that the answers to all questions are statistically significant, for both parametric and non-parametric tests.

Table 16: VASE 2.0 Results for *Quality* questions for H4

| No | Hypothesis 4, N=180 | Mean | Mode | SD | Wilcoxon Rank sum | t-test |
|----|---|--------------|----------|--------------|-------------------|--------------|
| 1 | <i>I think the authoring system helps to minimise errors in adaptive strategies.</i> | 3.878 | 4 | 1.018 | 0.000 | 0.000 |
| 2 | <i>I think the authoring system helps to create complete adaptive strategies.</i> | 3.878 | 4 | 0.903 | 0.000 | 0.000 |
| 3 | <i>I found the authoring system prevented me to create incorrect/invalid adaptive strategies.</i> | 3.810 | 4 | 0.973 | 0.000 | 0.000 |
| | Overall | 3.856 | 4 | 0.965 | 0.000 | 0.000 |

The results from Table 17 show the questions related to the *quality* of PEAL when compared to the 'indifferent' user, who would answer 3 for each question (on a scale from 1 to 5). The table shows the

⁸ The Wilcoxon test suffices, as it is stricter than the t-test; the results were provided for both in this table, for this category, and all other categories, for comparison only.

questions related to H4, and, for each question, the mean answer, the mode, the standard deviation (SD), and the results of the t-test and Wilcoxon test⁹ in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_q = 0.0167$). The results from all the questions indicate statistical significance, even for the PEAL overall *quality* average value (for the t-test and the Wilcoxon test). Overall, users reported that PEAL produced code that of a lower quality, with mean value of (2.854 ± 1.359) and mode of 2. This is possibly due to the fact that PEAL did not prevent users to connect/join incompatible constructs together, resulting in adaptive strategies with more errors in comparison with the VASE 2.0 authoring system.

Table 17: PEAL Results for *Quality* questions for H4

| No | Hypothesis 4, N=180 | Mean | Mode | SD | Wilcoxon Rank sum | t-test |
|----|---|--------------|----------|--------------|-------------------|--------|
| 1 | <i>I think the authoring system helps to minimise errors in adaptive strategies.</i> | 3.052 | 2 | 1.401 | 0.000 | 0.000 |
| 2 | <i>I think the authoring system helps to create complete adaptive strategies.</i> | 2.873 | 2 | 1.331 | 0.000 | 0.000 |
| 3 | <i>I found the authoring system prevented me to create incorrect/invalid adaptive strategies.</i> | 2.636 | 2 | 1.345 | 0.000 | 0.000 |
| | Overall | 2.854 | 2 | 1.359 | 0.000 | 0.000 |

⁹ The Wilcoxon test suffices, as it is stricter than the t-test; the results were provided for both in this table, for this category, and all other categories, for comparison only.

The overall results for hypothesis H4 with respect to the *quality* are shown in Table 18. The table shows the mean values for PEAL and VASE, respectively, and the Wilcoxon test, evaluating the significance of the difference between the two means, in terms of the value of the probability p (significant if $p < 0.05$ as well as at $p < p_q = 0.05/3 = 0.0167$). The results here are significantly lower for each question for PEAL when compared to VASE 2.0.

Table 18: PEAL versus VASE Results for questions related to *Quality* (H4)

| Questions | PEAL Mean | VASE Mean | Wilcoxon Rank sum |
|--|--------------|--------------|----------------------|
| <i>Q1: I think the authoring system helps to minimise errors in adaptive strategies</i> | 3.053 | 3.878 | 0.000 |
| <i>Q2: I think the authoring system helps to create complete adaptive strategies</i> | 2.874 | 3.878 | 0.000 |
| <i>Q3: I found the authoring system prevented me to create incorrect/invalid adaptive strategies</i> | 2.637 | 3.810 | 0.000 |
| Overall | 2.854 | 3.855 | 0.000 |

Summarising, the results in Table 19 indicates that the mean for VASE 2.0 was much higher than the mean for PEAL for all four categories, suggesting that users perceived VASE 2.0 to be better than PEAL in terms of *user role, benefits, usability* and *quality*. Please note that all these differences are statistically significant even with the Bonferroni corrections.

Table 19: Summary of PEAL versus VASE Results for question related to each category

| No | Category | PEAL Mean | VASE Mean | Wilcoxon Rank sum |
|----|-------------------------------------|--------------|--------------|----------------------|
| 1 | <i>User roles</i> related questions | 2.73 | 3.682 | 0.000 |

| | | | | |
|---|------------------------------------|------|-------|-------|
| 2 | <i>Benefits</i> related questions | 2.80 | 3.705 | 0.000 |
| 3 | <i>Usability</i> related questions | 2.85 | 3.418 | 0.000 |
| 4 | <i>Quality</i> related question | 2.85 | 3.856 | 0.000 |

6.7 Discussions and Hypotheses Validation

The research experiments discussed in this chapter focused towards addressing the research questions and hypotheses. For the final and largest experiment, described in Section 6.6, the sample size was a total of 380 for a quantitative experiment conducted on *Amazon Mechanical Turk*. The number is just below the recommended sample size to argue universal significance, which is 385, the limitations of the sample size having been discussed in Chapter 3 section 3.4.6. However, this research has conducted overall four case studies, two qualitative and two quantitative. Two of the quantitative ones bring the size up to $380 + 47 = 427$ for the quantitative experiments. A sample size of this number is acceptable in the context of this research. All references within this chapter when discussing the results, represent significance with respect to the sample, which was calculated with standard statistical methods, as described in Chapter 3.

Another point that can be mentioned here is that the demographics of the sample size was completely random, as it was chosen by the *Amazon Mechanical Turk* service; the only parameters which were chosen by the researcher were the qualification of the workers, to ensure that good quality workers only conduct the *human intelligent task* (HIT) as shown in the Figure 51.

Worker requirements

Require that Workers be Masters to do your HITs (Who are Mechanical Turk Masters?)
 Yes No

Specify any additional qualifications Workers must meet to work on your HITs:
 (up to 5)

Project contains adult content (See details)
 This project may contain potentially explicit or offensive content, for example, nudity.

HIT Visibility (What is HIT visibility?)
 Public - All Workers can see and preview my HITs
 Private - All Workers can see my HITs, but only Workers that meet all Qualification requirements can preview my HITs
 Hidden - Only Workers that meet my HIT Qualification requirements can see and preview my HITs

Figure 51: Amazon workers’ requirement to do HIT

Hypothesis 1: “*User roles* can be used to show relevant authoring options specific to the author’s expectation with the authoring system to avoid *confusion* and *cognitive overload*”.

Confirmation: Hypothesis 1 is supported, based on the statistically significant findings and statistical tests reported in Section 6.6.1, which suggest that the use of the *user role*, in conjunction with the *functionality matrix* leads to *user interface (UI)* options which are more relevant to the user, according to the user’s knowledge and skills, provided that it is controlled via a *functionality matrix*, as described in Chapter 4. The majority of the users confirmed that using user roles to display relevant options leads to higher user satisfaction. This indicates that using personalisation and customisation in an adaptive authoring system will ultimately lead to higher *user satisfaction*. This is accomplished because users found the *user roles* to be useful in displaying relevant *UI* options to the user. In other words, displaying too many irrelevant *UI* options to the users, not in-line with the user’s knowledge, will lead to confusion and cognitive overload, as the user gets overwhelmed and confused by the options presented.

Hypothesis 2: Having a personalised adaptive authoring platform, based on a visual programming paradigm, to create adaptive specifications, will offer *benefits*. Hypothesis 2 aimed at exploring the

different *benefits* of visual programming techniques, in the terms of *shallow learning curve (A)*, *higher user satisfaction (B)* and *less error-prone (C)*.

- *Shallower learning curve:*
 - By lowering programming barriers to create Adaptive specifications when compared to text-based techniques;
 - Non-programming-savvy authors will find it easier to create adaptation strategies by manipulating visual elements as opposed to writing adaptive programming language syntax.
- *Higher user satisfaction:*
 - By representing adaptation language visually to improve user-interaction and tracking;
 - By making authoring fun, users may enjoy spending more time in creating adaptive strategies.
- *Less error-prone:*
 - Users will make fewer errors in adaptive specification creation and encourage reuse.

Confirmation: Hypothesis 2

The hypothesis examined different benefits of visual programming techniques, some of which have been discussed in Chapter 2, Section 2.9 in this thesis. The hypothesis is supported based on the statistically significant findings and statistical tests in Section 6.6.1. The different features examined achieved statistical significance, suggesting that creating adaptive strategies visually offers several benefits to the users. These benefits include *shallow learning curve (A)*, *higher user satisfaction (B)* and *less error-prone (C)*. This was confirmed by the answers provided for the questions which related to the (A), (B) and (C), responses to the questions listed in **Table 10** for VASE and **Table 11** for PEAL authoring system.

The hypothesis was further confirmed by the interviews conducted, users clearly preferred a visual way of creating adaptive strategies in comparison to creating adaptive strategies written using an Adaptation Language. The results also suggested that more users thought that the VASE 2.0 authoring system was fun to use and would not mind playing with the system.

Hypothesis 3: Designing Adaptive Specifications visually will provide better adaptive specification *correctness (A)* and *completeness (B)*. Creating export functionality into generic adaptation languages (such as LAG) can help in delivering the adaptive course on various systems (e.g., ADE, MOT).

Confirmation: The latter part, about the export functionality, as already been addressed in Chapter 5, Section 5.5.3. In order to assess adaptive specification correctness and completeness, the definition of each was analysed in the context of an adaptive authoring system.

Correctness: *Correctness* of a program is claimed when the program is correct with respect to its specifications. *Functional correctness* relates to the input and output behaviour of the program (i.e., for every input it produces an expected output) [62]. A difference between *total correctness* and *partial correctness* is that the *total correctness* requires that the program terminates while *partial correctness* requires that if an answer is returned it will be correct.

In light of the above definition, a program is said to be *totally correct* if it follows the language specification and terminates successfully. However, there is no formal definition of Adaptive Strategy correctness in the literature. The proposed authoring tool helps the author to create adaptive strategies, which have fewer errors, compared to the text-based authoring tools because invalid constructs will not connect; only valid LAGBlocks can be connected together, according to the language grammar, thus providing an authoring environment which is forgiving to the author. This was confirmed by the majority of the users who agreed that VASE provides adaptive strategy correctness to minimise the errors produced.

Completeness: In logics, a formal system is termed *complete* with regard to a specific property if each formula having the property can be obtained using that system, else the system is said to be *incomplete* [63]. Different kinds of completeness have been presented in Section 6.7. According to the definition of completeness of a program, VASE provides completeness in the context of adaptive strategy authoring.

There are several forms of completeness:

Expressive completeness – A formal language is expressively complete if it can express the subject matter for which it is intended [63].

Functional Completeness – A set of logical connectives associated with a formal system is functionally complete if it can express all propositional functions [63].

Semantic completeness – A formal system is complete with respect to tautologousness or “semantically complete” when all its tautologies are theorems, whereas a formal system is “sound” when all theorems are tautologies (that is, they are semantically valid formulas: formulas that are true under every interpretation of the language of the system that is consistent with the rules of the system) [63].

Structurally complete - a logic is structurally complete if every admissible rule is derivable. [63].

According to the definitions of completeness above, the VASE 2.0 authoring system provides completeness in the context of adaptive strategy authoring, for instance, adaptive strategies created using the VASE 2.0 authoring system offer expressive completeness, functional completeness, semantic completeness and are structurally complete.

The adaptive strategies created using the VASE 2.0 authoring system offer *expressive completeness*, in the sense that, any LAG statement can be expressed, which the authoring system was intended for. It is true in the case of *functional completeness* because it can be used to express all of the LAG functions.

Semantic completeness is also provided, under every interpretation of the language of the system that is consistent with the rules of the system. The adaptive strategies are said to be *structurally complete* because every rule of the adaptation language is derivable. In the analysis of the above definitions of completeness in the context of adaptive strategies authoring, it is confirmed that VASE 2.0 allows authoring of adaptive strategies with a higher degree of correctness and completeness.

Hypothesis 4: Visual programming techniques can be implemented to enhance adaptive authoring systems and adaptation languages.

Confirmation: The hypothesis is related to the implementation of a visual authoring framework for adaptation language and an authoring environment to create adaptive specifications. It is possible to create a visual authoring environment to implement the requirements listed in the section 6.4.1 and refined requirements listed in 6.5.2.

A visual programming framework (LAGBlocks) to represent the LAG adaptation language and an adaptive authoring system (VASE 1.0) were created by using the block programming framework

Table 20: Comparison of adaptive hypermedia authoring systems and VASE 1.0 and VASE 2.0

| System | Adaptive behaviour | Ease of use | Completeness | Correctness | Visualisation | Interoperability | Separate authoring roles |
|---|--------------------|-------------|--------------|-------------|---------------|------------------|--------------------------|
| Interbook | Y | Y | Y | Y | N | N | N |
| AHA!: AMT | Y | Y | N | N | N | N | Y |
| AHA!: Domain Model Tool | Y | Y | N | N | Y | N | Y |
| AHA!: Graph Author Tool | Y | Y | N | N | Y | N | Y |
| AHA!: Concept Editor | Y | Y | N | N | N | N | Y |
| AHA!: strategy editor | Y | N | N | N | N | N | Y |
| AHA!: form editor | Y | N | Y | Y | Y | N | Y |
| ACCT: Subject Concept Space Editor (SMCS) | N | Y | Y | Y | N | N | Y |
| ACCT: Narrative Model Builder | Y | Y | Y | Y | N | N | Y |

| | | | | | | | |
|-------------------------------|---|---|---|---|---|---|---|
| APeLS | Y | N | N | N | N | N | Y |
| ACTSim | Y | Y | N | N | Y | N | N |
| GRAPPLE GAT: DM | Y | Y | Y | Y | Y | Y | Y |
| GRAPPLE GAT: PRT editor | Y | N | N | N | Y | Y | Y |
| GRAPPLE GAT: CAM editor | Y | Y | Y | Y | Y | Y | Y |
| MOT | Y | Y | N | N | Y | Y | Y |
| PEAL | Y | N | N | N | Y | Y | N |
| VASE 1.0 | N | Y | Y | Y | Y | N | Y |
| VASE 2.0 | Y | Y | Y | Y | Y | Y | Y |

called OpenBlocks [39]. To address the limitation of VASE 1.0 described in section 5.4.3, the visual extension and the authoring system (VASE 2.0) were redesigned, using a different yet powerful block programming framework called Blockly. To confirm the hypothesis, two iterations of the VASE authoring system were designed and implemented, using various open technologies to address the issues and the desired features highlighted by the evaluations.

6.8 Revisiting the Comparison of Authoring Systems for Adaptive Hypermedia

In Section 2.6, we have analysed the current state of the art in terms of the most well-known existing adaptive hypermedia authoring systems. Here, we revisit this comparison, also adding the two systems which I have contributed with, VASE 1.0 and VASE 2.0 (see Table 2). This feature comparison is based on features state of the art Adaptive Hypermedia Authoring Systems of VASE 1.0 and 2.0 as described in Chapter 5, Sections 5.4 and 5.5. The first column checks if the particular tool expresses adaptive behaviour,

as this is the main target for this research, and so on. All features are of a binary nature (Y for yes and N for no). As can be seen, VASE 2.0 encompasses all desired features, which are not available in the previous systems.

6.9 Conclusions

Concluding, via this Chapter, it was shown that a visual authoring system for adaptive hypermedia can lower the threshold for knowledge about programming. It was also shown that non-programming savvy authors can quickly learn to use the proposed LAGBlocks visual extension, to create and edit existing strategies without having to type LAG adaptation language syntax. The more detailed conclusions follow.

All of the hypotheses suggested throughout the chapter have been examined extensively. This chapter has proposed a highly refined set of hypotheses, which address all aspects of creating adaptive specification for adaptive hypermedia related to this research.

Quantitative evaluations were conducted over two experiments, a small scale experiment, with 47 users, and a large scale with 380 users. The first experiment was conducted on SurveyMonkey and the second evaluation was conducted using Amazon Mechanical Turk (AMT). The issues associated with the first experiment were that some adaptive authoring and user modelling aspects were not explored in detail or not explored at all, through the initial questionnaire answered by the users. Therefore, the follow-up experiment (explained in Section 6.6.1) was conducted. The large evaluation generated results that were matched to the recommended sample size. This experiment had statistical significance for most of the questions, showing that results were conclusive within the users questioned.

Hypothesis 1 focused towards using user roles to display relevant authoring options specific to author expectations with the authoring system to avoid confusion and cognitive overload.

Users indicated that the VASE authoring system displayed relevant options, with a mean value of 3.9 ± 0.609 with a mode of 4, the mode indicating the most frequent answer of positive. Users preferred to choose the

options they would like to see in the authoring system, with a mean value of 3.9 ± 0.611 with mode of 4. This was the highest mean amongst the questions regarding the user roles. It means that the users preferred to be able to control the options displayed to them. Largely the users were satisfied with the features which were displayed to them, not by a huge majority but they did answer it in positive, with a mean value of 3.73 ± 0.702 . The results indicate that considering features related to the *user roles*, the users expressed positive feedback during the experiment. Any average value equal or greater than 3 is deemed to be positive. The value of 3.616 is believed to be credibly positive.

Hypothesis 2 aimed at the benefits of using visual programming techniques to create adaptive specifications in the terms of *shallow learning curve*, *higher user satisfaction* and *less error-prone*. The results to the questions related to the *benefits* of authoring paradigm used in VASE are listed in **Table 10**. The majority of the users thought that VASE was easy to learn to use and were not confused, this would be one of the benefits of the authoring system from the user's viewpoint. A popular view amongst the users was that, VASE authoring system can be used to create adaptive strategies without much knowledge of the LAG adaptation language, with a mean value of (3.821 ± 0.712) and mode of 4. Aspects regarding the *higher user satisfaction* and *less error-prone* are reflected in Questions 4,5,6,7,8 and 9 with mean values of 4.22 ± 0.7 , 3.721 ± 0.756 , 3.773 ± 0.746 , 3.926 ± 0.670 , 3.71 ± 0.716 and 3.805 ± 0.769 , respectively. The largest mean value is generated for the questions asking if the authoring system provided a fun way to create adaptive strategies and if the authoring system provided a simple way to create adaptive strategies. Overall, the results indicate the *benefits* of using visual programming techniques in adaptive strategy creation. The overall benefit mean value is 3.705 ± 0.786 .

For Hypothesis 3, *correctness* of a program is claimed when the program is correct with respect its specifications. *Functional correctness* relates to the input and output behaviour of the program (i.e., for every input it produces an expected output) [62]. A difference between *total correctness* and *partial correctness* is that the *total correctness* requires that the program terminates while *partial correctness* requires that if an

answer is returned it will be correct. According to this definition *correctness*, a program is said to be *totally correct* if it follows the language specification and terminates successfully. The VASE authoring tool helps authors to create adaptive strategies which have fewer errors compared to the text based authoring tools because invalid constructs cannot be formed and will not connect in the authoring environment, only valid LAGBlocks can be connected together according to the language grammar, thus providing an authoring environment which is forgiving to the author. This was confirmed by the majority of the users who agreed that VASE provides adaptive strategy correctness to minimise the errors produced. In logic, a formal system is termed *complete* with regard to a specific property if each formula having the property can be obtained using that system, else the system is said to be *incomplete* [63]. Different kinds of completeness have been presented in Section 6.7. According to the definition of completeness of a program, VASE provides completeness in the context of adaptive strategy authoring.

Adaptive strategies created using the VASE authoring system offer *expressive completeness*, *functional completeness*, *semantic completeness* and *structural completeness*. The adaptive strategies created using the VASE authoring system offer *expressive completeness*. For instance, any LAG statement can be expressed which the authoring system was intended for. It is true in the case of *functional completeness* because it can be used to express all of the LAG functions. *Semantic completeness* is also provided as every interpretation of the language of the system is consistent with the rules of the authoring system. The adaptive strategies are said to be *structurally complete* because every rule of the adaptation language is derivable.

However, there is no formal definition of adaptive specification correctness and completeness in the literature, this research has taken this opportunity to introduce the following definition of adaptive correctness: *An adaptive specification is deemed correct if its execution produces the desired objectives and there are no syntax errors according to the grammar of the adaptation language.*

In the analysis of the above definitions of completeness and correctness in the context of adaptive strategies authoring, it is confirmed that VASE allows authoring of adaptive strategies with higher degrees of *correctness* and *completeness*.

Hypothesis 4 concentrated on the implementation of adaptation language and adaptive authoring tool by utilising visual programming techniques. A visual programming framework (LAGBlocks) to represent LAG adaptation language and an adaptive authoring system (VASE 1.0) was created by using the block programming framework called OpenBlocks [39]. To address the limitation of VASE 1.0 described in Section 5.4.3, the visual extension and the authoring system (VASE 2.0) was re-designed using a different block programming framework called Blockly. To confirm the hypothesis, two iterations of the VASE authoring system were designed and implemented using various open technologies, to address the issues and the desired features highlighted by the evaluations.

These requirements were implemented by using various web technologies based on W3C standards. To follow W3C standards served two purposes, firstly, it was highlighted in the authoring imperative (section 2.7) in the literature, and secondly, it meant that any browser which implements the W3C standard will automatically be able to run the VASE 2.0 authoring system. It also meant that to run VASE 2.0 will not require any third-party plugin to run the authoring system to create adaptive specification rules.

These requirements were implemented by using various web technologies based on W3C standards such as HTML5. To follow W3C standards served two purposes, firstly, it was highlighted in the authoring imperative (section 2.7) in the literature and secondly, it meant that any browser which implements the W3C standard will automatically be able to run VASE 2.0 authoring system. It also meant that to run VASE 2.0 will not require any third-party plug-in to run the authoring system to create adaptive specification rules.

The current chapter has achieved all the objectives stated earlier in Section 1.5 & 6.2 and helped in answering the set of research questions **RQ1:** What can be done to simplify and improve adaptive specification authoring for adaptive hypermedia?, **RQ 1.1:** How can an adaptive authoring system make adaptive strategy creation easy for authors with different needs, e.g. knowledge and experience?, **RQ 1.2:**

Are there benefits in using visual programming techniques to create adaptive specifications and if so, what would these benefits be?, **RQ 1.3:** How can an adaptive authoring system assist authors in adaptive specification's correctness and completeness?, **RQ 1.4:** How can the visual programming techniques and technologies be implemented, in order to enhance the adaptive authoring system and adaptation language, and thus provide a high level of effectiveness, efficiency and satisfaction amongst authors?

7 Conclusions and Recommendations for Future Work

All research presented in this thesis is focused on addressing the main research question discussed in the beginning of the work, which is:

RQ1 - What can be done to simplify and improve the adaptive specification authoring for adaptive hypermedia?

The outcome of the research has ultimately guided the exploration and creation of a novel research approach to adaptive specification authoring, using a technological method that has not been investigated before.

In the following, we will show how we have addressed each of the sub-questions deriving from the main question above.

7.1 Answers to Research Questions

This research reflected on the problem of adaptive specification creation being difficult and often incomplete, and on the fact that this process needs to be improved, as it represents a bottleneck in Adaptive Hypermedia.

In the following the sub-research questions are re-examined, and the answer, derived from this research is presented.

RQ 1.1 How can an adaptive authoring system render adaptive strategy creation easy for authors with different needs, e.g. knowledge and experience?

Answer: Adaptive Hypermedia advances eLearning systems by providing a personalised model of learning for each learner; however this flexibility comes at a cost of rendering the process of authoring of adaptive specification really difficult for an author, who is often pressed by time and may lack the skills needed to write adaptive specifications in an adaptation (programming) language, such as LAG. This

research attempts to address the problems faced by authors and suggests a new way to creating adaptive specifications, which is shown to be intuitive and requiring minimal training (see Chapter 6, Section 6.6.1).

The overall answer resulting from the different evaluations and experiments is that the creation and the usage of adaptive specifications can be simplified, by using several techniques. Answering this question required 4 different experiments. The first experiment was exploratory in nature, aiming to find out if a chosen visual framework to design adaptive specification visually will be appropriate for non-programming savvy users, as discussed in the Section 6.3. The next point of research was to retrieve qualitative feedback from the LAG experts, to uncover deeper functional issues with regards to the authoring of adaptation specifications from technical perspectives; e.g., tokenizer, parser, language grammar and compatibility (see Section 6.4). The final qualitative evaluation, to receive feedback on adaptive authoring, was conducted with the university lecturers with eLearning experience (see Section 6.5). This produced a list of features for an adaptive authoring system; the final requirements list for an authoring system has been presented in Section 5.5.

RQ 1.2 Are there benefits in using visual programming techniques to create adaptive specifications and if so, what would these benefits be?

The research used an adaptive authoring tool. It has been developed during the research for this thesis (see Chapter 5), based on the framework which was also designed and developed for this thesis (see Chapter 4).

The research used an adaptive authoring visual framework, an adaptive authoring tool based on the issues surrounding the creation and reuse of adaptive specification and the problems faced by the authors in adaptive specification creation and reuse.

The research has thus shown that, yes, there are benefits in using visual programming techniques when creating adaptive specifications. The benefits are that users (here, authors) clearly prefer these tools to other, more programming-based interfaces, both in theory (as shown during the initial experiments - see Section 6.3) and in practice (see Sections 6.5, 6.6).

RQ1.3 How can an adaptive authoring system assist authors in adaptive specification's correctness and completeness, as well as interoperability?

Answer: Correctness - In theoretical computer science, correctness of a program is asserted when it is said that the program is correct with respect to a specification. Functional correctness refers to the input-output behaviour of the program (i.e., for each input it produces the expected output). A program is said to be *totally correct* if it follows the language specification and terminates successfully [64]. The proposed authoring tool helps the author to create an adaptive strategy, which has fewer errors when compared to text-based authoring tools, because invalid constructs will not connect. This provides a high level of correctness in adaptive strategy authoring, as only valid LAGBlocks can be formed, according to the adaptation language grammar, please see Chapter 2, section 2.9.2, for more details on how blocks only allow valid or compatible blocks to be connected together. This was confirmed by the majority of the users who agreed that VASE provides adaptive strategy correctness, to minimise the errors produced (see Sections 6.5, 6.6).

Completeness: The VASE authoring system provides completeness in the context of adaptive strategy authoring [65]. For instance, adaptive strategies created using the VASE authoring system offer *expressive completeness, functional completeness, semantic completeness* and *structural completeness*. The adaptive strategies created using the VASE authoring system offer *expressive completeness*, as any LAG statement can be expressed, for which the authoring system was intended for. This also holds for the case of *functional completeness*, due to the fact that it can be used to express all of the LAG functions. *Semantic completeness* is also provided, as every interpretation of the language of the system is consistent with the rules of the system. The adaptive strategies are said to be *structurally complete*, as every rule of the adaptation language is derivable. In the analysis of the above definitions of completeness in the context of adaptive strategies authoring, it is confirmed that VASE allows authoring of adaptive strategies with higher degrees of correctness and completeness.

However, there is no formal definition of adaptive specification correctness and completeness in the literature. Thus, this research further contributes by taking this opportunity to introduce the following definition of adaptive correctness:

Definition: *An adaptive specification be deemed correct if its execution produces the desired objectives and there are no syntax errors according to the grammar of the adaptation language.*

Definition: *An adaptive specification be deemed complete if its execution offers functional completeness, semantic completeness and structural completeness and its consistent with the rules of the adaptation language.*

RQ1.4. How can the visual programming techniques and technologies be implemented, in order to enhance the adaptive authoring system and adaptation language, and thus provide a high level of effectiveness, efficiency and satisfaction amongst authors?

Answer: A visual programming framework (LAGBlocks) to represent the LAG adaptation language and an adaptive authoring system (VASE 1.0) were created, by using a block programming framework called OpenBlocks [39]. To address the limitation of VASE 1.0 described in Section 5.4.3, the visual extension and the authoring system (VASE 2.0; see Section 5.5) was re-designed, using a different block programming framework called Blockly. To answer this research question and confirm the related hypothesis, as said, two iterations of the VASE authoring system were designed and implemented using various open technologies to address the issues and the desired features highlighted by the evaluations. These requirements were implemented by using various web technologies based on W3C standards. To follow W3C standards served two purposes; firstly, this was required by the authoring imperatives (see Section 2.7) in the literature and secondly, it meant that any browser which implements the W3C standard will automatically be able to run the VASE 2.0 authoring system. It also meant that to run VASE 2.0 will not require any third-party plug-in to run the authoring system to create adaptive specification rules.

One of the key findings in the literature review which is related to this research was reported by the LAG-XLS [66] creators, who have stated that “the major requirements for the ideal language are: *reuse*, *flexibility*, *high level semantics*, and *ease of use*”. The research presented in this thesis has attempted to address every aspect of these requirements. For instance, *high level semantics* has been created (LAGBlocks) to hide the programming syntax from the author, to lower the barrier of entry, in order to create adaptive specifications. *Ease of use* of the proposed visual framework has been demonstrated by allowing to create adaptive specifications visually (as confirmed by the users in Sections 6.5 and 6.6). The *reuse* is addressed by using the visual representation, which can be further converted into the LAG programming language behind the scenes, able thus to be used by other adaptive tools which use LAG as the adaptation language. In addition, the tools have been found to be *easy to use* by the users during the evaluations conducted by this research (see Sections 6.5 and 6.6).

7.2 Addressing the Research Objectives

In order to answer the research questions, they have been mapped onto five different Objectives (as per Chapter 1 Section 1.5). Below, the way each of these objectives was implemented in this thesis is explained.

Objective 1 – *Conduct an extensive theoretical background research into the area, to find the problems faced by the authors in adaptive strategy creation and to propose a theoretical framework to improve adaptive authoring for authors with different needs.*

This objective has been addressed mainly in the early phase of the research. However, it remained as a continuous process, as the need for further research emerged with each new aspect in the research. Chapter 2 presents the detailed relevant theoretical background research and related work. User roles have been implemented on a rudimentary level in adaptive hypermedia and visual programming techniques with regards to the authoring adaptive strategies have not been well explored in adaptive hypermedia. Moreover, no adaptive authoring platform has utilised a block programming framework to create adaptive

specifications. The limitations and gaps found in the theoretical background and related work provided a starting point for the further work in this thesis.

Objective 2 - *Get feedback from users for their preferred way of authoring an adaptive strategy. Conduct a survey which reflects the users' preferred way to create adaptive programs.*

An initial evaluation was conducted, to find out if users preferred a visual way of describing adaptive specifications or if they preferred to write strategies in plain text, to find out if such a tool is worthwhile developing (see Section 6.3). An online questionnaire was created, to guide all participants through the evaluation process. To ensure that the participants will volunteer for this evaluation, the time spent performing it was deliberately limited (to around 10-15 minutes). This questionnaire consisted of category type and ordinal category types responses. This questionnaire showed that the most users, more than 60%, preferred adaptive strategies to be visual, by using a visual programming technique, as opposed to the plain text LAG syntax. Users were asked to select their profession; some of the users were company workers, who did not have much programming language experience. These users also preferred the visual way of creating an adaptive strategies; this shows that the non-professional programmers preferred the visual way to create adaptive strategies. To increase the uptake of adaptive specification authoring, it was very important to ensure that the authors without much knowledge of programming languages would be catered to - and they clearly showed a preference to create *adaptive specifications* visually, as opposed to writing adaptive strategies in a text-based environment. Based on the above, research Objective 2 has been achieved, in order to answer the research question RQ1.2.

Objective 3: - *Propose a visual programming framework for implementation to provide adaptive specifications' completeness and correctness.*

This research objective has been thoroughly explored via two different approaches. The first approach was through the extensive study and work on the relevant literature, to understand the different visual programming frameworks. This included research into visual programming techniques, to investigate if the LAG [9] language grammar can be represented visually. There has not been a block programming framework used for authoring adaptive specifications before. Therefore, the visual framework proposed

was derived from the concepts and theories found in the domain of visual programming, to design new visual constructs, to represent the entire LAG programming language.

In summary, the proposed visual programming framework was successfully implemented to:

- Create LAG visual *adaptive specifications* by using visual programming techniques;
- Represent the entire LAG language grammar via visual programming techniques;
- Enable authors to create adaptive specifications in a puzzle-building manner, with a higher degree of *completeness* and *correctness*.

Based on the above, research Objective 3 has been implemented, in order to answer the research questions RQ1.3.

Objective 4 – *Implement a visual programming framework to enhance adaptive authoring systems and adaptation languages. To propose a suitable framework for adaptive authoring, to allow authors to create visual adaptive specification, by manipulating visual elements to create adaptive specification, thereby lowering the programming threshold for authors. {RQ1.4}*

This research objective was achieved through the design and implementation of VASE (see Chapter 5), serving as an adaptive authoring system, to create and reuse adaptive specifications for non-programming savvy users. The implemented system enables users to author adaptive strategies.

The VASE authoring system provides:

- Support in creating, editing and sharing adaptive specifications with other authors;
- Assistance in creating adaptive specification by manipulating visual blocks, for the author with little or no knowledge of adaptation programming language;
- Interface with another Adaptive Delivery Engine (ADE [15]), which uses the LAG adaptation language to apply adaptations. This means that adaptive specifications created in VASE will be able to execute in ADE.

Based on the above, research Objective 4 has been implemented, in order to answer research question RQ1.4.

Objective 5 - Conduct a series of experiments that investigate the appropriate approach and features to design an adaptive authoring system, and to test the practical development of the newly added features in an adaptive authoring system, addressing the acceptance of the visual form of adaptive authoring in the evaluations.

This research objective has been implemented as an on-going process, since the beginning of the design phase in Chapter 5, to the evaluations in Chapter 6. This outcomes of this objective culminated with comparing the visual authoring tool (VASE) with non-visual authoring tools (PEAL) representing the previous state of the art.

This research has conducted four experiments; the second experiment was with LAG experts who had advance knowledge and experience of adaptive hypermedia and teaching (see Section 6.4). This provided qualitative feedback with regards to the technical issues relating to the LAG language and authoring tool, as described in Chapter 6 Section 6.4. The third experiment was the implementation of VASE 1.0 and provided qualitative feedback from the university lecturers described in Chapter 6 Section 6.5.

The fourth experiment was the implementation of VASE 2.0 and provided quantitative feedback from the users on a large scale, which allowed for results showing statistical significance on questions related to usability and usefulness (see Section 6.6). The users evaluated the authoring system for authoring adaptive specifications. The results from the experiment showed which features were considered better than text-based authoring tools.

Based on the above, research Objective 5 has been implemented, in order to answer the research question RQ1.4.

7.3 Original Contributions

The major contribution of this research is to answer the research question **What can be done to simplify and improve adaptive specification authoring for adaptive hypermedia?** The research has extensively

investigated theories, frameworks, techniques and technologies to provide both a theoretical and a technical answer for the question.

The work presented in this thesis introduces a novel visual platform for collaborative adaptive specification authoring in a visual environment. This overcomes traditional text-based authoring environments, by also focusing on multiple groups of users. The case studies presented cater for users with varying expertise of adaptive authoring and attempt to extract and highlight user perceptions towards visual authoring. The impact of user roles and sharing is discussed in detail. Furthermore, the ambiguities around the definition of completeness and corrections of an adaptive strategy are discussed, with a notion to achieve an agreeable definition. Authoring issues and user perceptions have been statistically evaluated and corresponding results have been derived accordingly. The work also highlights minimalistic requirements for a visual adaptive strategy authoring environment, considering the graphical elements and their impact on strategy-author cognition.

7.4 Recommendations for Future Work

The approach described in this research can be applied to create a visual blocks framework for other programming languages as well, for instance, for the GRAPPLE GALE programming language (see Section 2.5.6). This could have the benefit that the GALE programming syntax could be created just by manipulating visual elements, as shown in this thesis for the LAG programming language. In addition, the author would not need to know the GALE programming syntax to create adaptive specifications in GALE, thus potentially lowering the programming threshold.

The visual programming framework proposed by this research can also further be extended to represent other programming languages. This fact was suggested in the qualitative interviews as well. One of the points mentioned by one participant was that if a similar framework would be developed for Microsoft Excel to write Macros, this would be really useful in saving time and it would enable users without programming know-how to be able to create macros without much knowledge of the Excel programming (VBA).

Other technologies which could benefit from the visual programming approach presented in this thesis, to create a visual representation for the LAG programming language grammar, can be used to [67] represent the *Structured Query Language* (SQL), used extensively in commercial databases, visually, by representing SQL constructs in visual blocks, to create SQL queries [67]. This could enable business users to create SQL queries without extensive or expert knowledge of SQL. This approach could also ensure that the resultant queries generated visually would be correct and complete, due to the fact that the invalid visual objects would not connect together, in order to prevent incomplete and incorrect SQL queries.

8 References

- [1] We are Social, [Online]. Available: <https://wearesocial.com/blog/2018/01/global-digital-report-2018>. [Accessed 03 03 2018].
- [2] “Research and Markets,” [Online]. Available: https://www.researchandmarkets.com/research/qgq5vf/global_elearning. [Accessed 3 03 2018].
- [3] “Blackboard,” Blackboard, [Online]. Available: <http://www.blackboard.com/higher-education/index.aspx>. [Accessed 04 03 2016].
- [4] “Sakai,” Sakai, [Online]. Available: <https://www.sakaiproject.org/sakai-cle>. [Accessed 03 04 2016].
- [5] N. Stash, A. I. Cristea and P. D. Bra, “Explicit intelligence in adaptive hypermedia: Generic adaptation languages for learning preferences and styles.,” in *HT'05 CIAH Workshop*, Salzburg, 2005.
- [6] A. I. Cristea, “Authoring of Adaptive Educational Hypermedia.,” in *International Conference on Advanced Learning Technologies (ICALT 2007)*, Niigata, Japan, 2007.
- [7] A. I. Cristea, “Automatic Authoring in the LAOS AHS Authoring Model,” in *ACM Conference on Hypertext and Hypermedia (HT03) (AH Workshop)*, Nottingham, UK, 2003.
- [8] M. Hendrix and A. I. Cristea, “A spiral model for adding automatic adaptive authoring to adaptive hypermedia,” *Journal of Universal Computer Science (JUCS)*, vol. 14, p. 17, 28 September 2008.
- [9] A. I. Cristea, D. Smits, J. Bevan and M. Hendrix, “LAG 2.0: Refining a REusable Adaptation Language and Improving on Its Authoring,” in *Learning in the Synergy of Multiple Disciplines*, Springer, 2009, pp. 7-21.

- [10] A. I. Cristea and C. Stewart, "Automatic authoring of adaptive educational hypermedia," *Web-based intelligent e-learning systems: Technologies and application*, pp. 24-55, 2006.
- [11] A. I. Cristea, D. Smits, J. Bevan and M. Hendrix, "LAG 2.0: Refining a reusable Adaptation Language and Improving on its Authoring," in *European Conference on Technology Enhanced Learning*, Berlin Heidelberg, 2009.
- [12] J. D. Scotton and A. I. Cristea, "Resuing adaptive strategies in adaptive educational hypermedia systems," in *The 10th IEEE International Conference on Advanced Learning Technologies*, Sousse, Tunisia, pp. 528-532, 2010.
- [13] A. Cristea and L. Calvi, "The Three Layers of Adaptation Granularity," in *International Conference on User Modeling (UM'03)*, Johnstown, PA, USA, 2003.
- [14] J. G. K. Foss, *Manual and automatic authoring for adaptive hypermedia*, University of Warwick, 2012.
- [15] J. Scotton, *Supporting Delivery of Adaptive Hypermedia*, University Of Warwick, 2013.
- [16] P. De Bra and J. P. Ruiter, "AHA! Adaptive Hypermedia for All," in *Proceedings of WebNet 2001-World Cenference on the WWW and the Internet*, Orlando, FL, USA, pp. 262-268, Oct 2001.
- [17] J. D. Bothma and A. I. Cristea, "Augmenting Authoring of Adaptation Languages via Visual Environments," *Preface vii Development and Experimental Comparison of Exact and Heuristic Algorithms for the Maximum-Leaf Spanning Tree Problem*, pp. 24-26, 43, 2010.
- [18] M. Hendrix, *Supporting Authoring of Adaptive Hypermedia*, University of Warwick: PhD Thesis, Department of Computer Science, 2010.

- [19] N. Stash, A. I. Cristea and P. De Bra, "Adaptation languages as vehicles of explicit intelligence in adaptive hypermedia," *International Journal of Continuing Engineering Education and Lifelong Learning*, vol. 17, no. 4-5, pp. 319-336, 2007.
- [20] A. Cristea and A. de Mooij, "LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators.," in *The 12th Int. WWW'03 Conference. Alternate Track on Education*, Budapest, Hungary, 20-24 May 2003.
- [21] H. Wu, G. Houben and P. De Bra, "AHAM: A reference model to support," in *Proceedings of the Conference on*, Antwerp, Belgium, pp. 77-88, 12-16 Jul 1998.
- [22] P. De Bra, G. Houben and H. Wu, "AHAM: a Dexter-based reference model for adaptive hypermedia," in *Proceedings of the ACM conference on Hypertext and Hypemedia (Hypertext)*, 21-25 Feb, Darmstadt, Germany, 1999.
- [23] F. Halasz and M. Schwartz, "The Dexter hypertext reference model," *Communications of the ACM*, vol. 37, no. 2, pp. 30-39, Feb 1994.
- [24] P. Vrieze, P. Bommel and T. Weide, "A Generic Adaptivity Model in Adaptive Hypermedia," in *Adaptive Hypermedia and Adaptive Web-Based Systems*, Berlin Heidelberg, Springer , 2004, pp. 344-347.
- [25] A. Cristea and M. Arnout, "LAOS: Layered WWW AHS authoring model and their corresponding algebraic operators," in *Proc. of 12th www Conference*, 2003.
- [26] A. Cristea and A. de Mooij, "LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators.," in *The 12th Int. WWW'03 Conference. Alternate Track on Education*, Budapest, Hungary, 2003.

- [27] P. Brusilovsky, E. Schwarz and G. Weber, "ELM-ART : An Intelligent Tutoring System on World Wide Web," in *Intelligent Tutoring Systems, Third International Conference*, Montréal, 1996.
- [28] J. Eklund and P. Brusilovsky, "InterBook: An Adaptive Tutoring System," *UniServe Science News*, vol. 12, no. 3, pp. 8-13, 1999.
- [29] P. De Bra, D. Smits and N. Stash, "The Design of AHA!," in *In Proceeding of the ACM Hypertext Conference*, Odense, Denmark, 2006.
- [30] D. Dagger, V. Wade and O. Conlan, "A Framework for Developing Adaptive Personalized eLearning," in *Knowledge and Data Engineering Group*, Ireland.
- [31] O. Conlan and V. Wade, "Evaluation of APeLS - An Adaptive eLearning Service based on the Multi-model, Metadata-driven Approach," in *Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Eindhoven, The Netherlands, 2004.
- [32] C. Gaffney, D. Dagger and V. Wade, "Evaluation of ACTSim: a Composition Tool for Authoring Adaptive Soft Skill Simulations,," in *Fifth International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, Hannover, Germany, 2008, pp. 113-122.
- [33] "GRAPPLE," Grapple, [Online]. Available: <http://www.grapple-project.org/summary>. [Accessed 5th May 2011].
- [34] D. Smits, P. De Bra , K. v. d. Sluijs, A. I. Cristea, J. Foss, C. Glahn and C. M. Steiner, "GRAPPLE: Learning Management Systems Meet Adaptive Learning Environments," in *Intelligent and Adaptive Educational-Learning Systems*, 2012.

- [35] M. Hendrix, P. De Bra, M. Pechenizkiy, D. Smits and A. Cristea, "Defining Adaptation in a Generic Multi Layer Model: CAM: The GRAPPLE Conceptual Adaptation Model," in *Times of Convergence. Technologies Across Learning Contexts*, vol. 5192, Springer, 2008, pp. 132-143.
- [36] P. De Bra, D. Smith, K. v. d. Sluijs, A. I. Cristea and M. Hendrix, "GRAPPLE: Personalization and Adaptation in Learning Management Systems," in *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA)*, Toronto, 2010.
- [37] J. Foss and A. Cristea, "The Next Generation of Authoring Adaptive Hypermedia Platform: Using and Evaluating the MOT3.0 and PEAL tools," in *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, New York, NY, USA, 2010.
- [38] A. I. Cristea, D. Smits and P. De Bra, "Towards a generic adaptive hypermedia platform: a conversion case study," *Journal of Digital Information*, vol. 8, no. 3, 2007.
- [39] R. V. Roque, OpenBlocks: an extendable framework for graphical block programming systems, Doctoral dissertation, Massachusetts Institute of Technology, 2007.
- [40] "Google for Education," Google, [Online]. Available: <https://developers.google.com/blockly/guides/get-started/web>. [Accessed 2015 July 02].
- [41] S. I. Irny and A. A. Rose, "Designing a Strategic Information Systems Planning Methodology for Malaysian Institutes of Higher Learning Issues in Information System," 2005.
- [42] C. Quintana, J. Krajcik and E. Soloway, "Exploring a Structured Definition for Learner-Centered Design," in *Fourth International Conference of the Learning Sciences*, 2013.
- [43] E. Soloway, S. L. Jackson, J. Klein, C. Quintana, J. Reed, J. Spitulnik, S. J. Stratford, S. Studer, S. Jul, J. Eng and N. Scala, "Learning theory in practice: Case studies of learner-centered design," in *ACM Press*, New York, 1996.

- [44] S. Kujala, "User involvement: a review of the benefits and challenges," in *Behaviour & information*, HUT, Finland, 2003.
- [45] D. Fox, J. Sillito and F. Maurer, "Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry," in *Agile, 2008. AGILE '08. Conference*, Toronto, ON, 2008.
- [46] Usability.gov, "Usability," U.S. Department of Health and Human Services, [Online]. Available: <https://www.usability.gov/what-and-why/user-centered-design.html>. [Accessed 20 01 2018].
- [47] "GLGToolkit," GenLogic, [Online]. Available: <http://www.genlogic.com/products.html#Toolkit>. [Accessed 01 02 2018].
- [48] "Oryx," Oryx, [Online]. Available: <http://oryx-project.org/Oryx>. [Accessed 01 02 2018].
- [49] "HTML5.2," World Wide Web Consortium, [Online]. Available: <https://www.w3.org/TR/html52/>. [Accessed 01 02 2018].
- [50] J. Sauro, "Measuring Usability with the System Usability Scale," *Measuring Usability*, 2 Feb 2011. [Online]. Available: <http://www.measuringusability.com/sus.php>. [Accessed 25 Sep 2012].
- [51] G. D. Israel, *Determining sample size*. University of Florida Cooperative Extension Service, Institute of Food and Agriculture Sciences, EDIS, 1992.
- [52] HESA, "Staff in Higher Education," Higher Education Statistics Agency, 25 02 2015. [Online]. Available: https://hesa.ac.uk/index.php?option=com_pubs&Itemid=&task=show_year&pubId=1717&versionId=27&yearId=326. [Accessed 24 03 2016].

- [53] D. A. Qudah, A framework for adaptive personalised e-advertisements, Coventry: University of Warwick, 2016.
- [54] H.-F. Hsieh and S. Shannon, "Three approaches to qualitative content analysis," in *Qualitative Health Research*, 2005.
- [55] R. Likert, "A Technique for the Measurements of Attitudes," *Archives of Psychology*, vol. 22, no. 140, pp. 1-55, 1932.
- [56] F. L. Coolidge, Statistics: A gentle introduction. 2012:, Sage Publication, 2015.
- [57] A. Cristea, C. Stewart, T. Brailsford and P. Cristea, "Evaluation of Interoperability of Adaptive Hypermedia Systems: tesing the MOT to WHURLE Conversation in a classroom setting," in *International Workshop on Authoring of Adaptive Adaptable Educational Hypermedia*, Amsterdam, The Netherlands, Jul 2005.
- [58] G. Fawaz and A. Cristea, "MOT 2.0: A Case Study on the usefulness of social Modelling for Personalized E-Learning Systems," in *AIED - Artificial Intelligence in Education*, 2009.
- [59] M. J. C. Crump, J. V. McDonnell and T. M. Gureckis, "Evaluating Amazon's Mechanical Turk as a tool for experimental behavioral research," in *PloS ONE*, 2013.
- [60] S. D. Gosling, S. Vazire, S. Srivastava and O. P. John, "Should we trust web-based studies? A comparative analysis of six preconceptions about Internet questionnaires," in *American Psychologist*, 2004.
- [61] K. T. G. S. Buhrmester M., "Amazon's Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data?," in *Perspectives on psychological science*, 2011.
- [62] "Wikipedia," [Online]. Available: <http://www.wikipedia.org/>. [Accessed 22 Sep 2012].

- [63] “Completeness,” Wikipedia, [Online]. Available: [https://en.wikipedia.org/wiki/Completeness_\(logic\)#Forms_of_completeness](https://en.wikipedia.org/wiki/Completeness_(logic)#Forms_of_completeness). [Accessed 2016 01 02].
- [64] “Correctness,” Wikipedia, [Online]. Available: [https://en.wikipedia.org/wiki/Correctness_\(computer_science\)](https://en.wikipedia.org/wiki/Correctness_(computer_science)). [Accessed 02 03 2016].
- [65] “Turing Completeness,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Turing_completeness. [Accessed 05 03 2016].
- [66] N. Stash, A. Cristea and P. De Bra, “Learning Sytles Adaptation Language for Adaptive Hypermeida,” in *Proceedings of the Adaptive Hypermedia (AH)*, Dublin, Ireland, pp.323-327, 21-23 June 2006.
- [67] “VBA,” [Online]. Available: https://en.wikipedia.org/wiki/Visual_Basic_for_Applications. [Accessed 01 02 2018].
- [68] S. Faulkner, A. Eicholz, T. Leithead, A. Danilo and S. Moon, “World Wide Web Consortium,” W3C Consortium, [Online]. Available: <https://www.w3.org/TR/html52/>. [Accessed 20 12 2017].
- [69] A. I. Cristea, J. A. Khan and C. Stewart, “Adaptive Authoring of Adaptive Hypermedia towards, Role-based, Adaptive Authoring,” in *Computers and Advanced Technology in Education - 2011*, Cambridge, 2011.
- [70] J. Khan, A. I. Cristea and C. Stewart, “Adaptive Authoring of Adaptive Hypermedia towards, Role-based, Adaptive Authoring,” in *Computers and Advanced Technology in Education - CATE*, Cambridge, UK , 2011.
- [71] “GALE: A highly eXtensible adaptive hypermedia engine,” in *Hypertext 11, Proceedings of the 22nd*, Eindhoven, 2011.

- [72] J. Khan, "Node Classification in Hypermedia Systems," in *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, (pp. 1464-1469). Chesapeake, VA, 2007.

APPENDIX A: LAG Grammar

PROG
DESCRIPTION // “text”
VARIABLES // “text”
INITIALIZATION initialization (STATEMENT)
IMPLEMENTATION implementation (STATEMENT)
STATEMENT IFSTAT | WHILESTAT | FORSTAT |
BREAKSTAT | GENSTAT |
SPECSTAT | (STATEMENT)*
STATEMENT | ACTION
IFSTAT if CONDITION then (STATEMENT)+ |
if CONDITION then (STATEMENT) + else (STATEMENT)+
WHILESTAT while CONDITION
(STATEMENT)+ [TARGETLABEL]
FORSTAT for RANGE do (STATEMENT)
[TARGETLABEL]
BREAKSTAT break SOURCELABEL
GENSTAT generalize((CONDITION)*)
SPECSTAT specialize((CONDITION)*)
ACTION ATTRIBUTE OP VALUE
CONDITION enough((CONDITION)+, VALUE) |
PREREQ
RANGE “integer”
PREREQ ATTRIBUTE COMPARE VALUE
TARGETLABEL “text” | “”
SOURCELABEL “text” | “”
ATTRIBUTE GENCONCEPTATTR |
SPECCONCEPTATTR
GENCONCEPT ATTR LAOSCONCEPTMAP.CONCEPT.ATTR |
LAOSCONCEPTMAP.CONCEPT.ATTR.ATTRATTR |
LAOSCM.ATTR |
LAOSCM.LAOSCM.ATTRATTR |
LAOSCM.LAOSCM.CONCEPT.ATTR.ATTRATTR
SPECCONCEPTATTR ‘\SPECCONMAP\SPECCON\SPECATTR\ATTR’.ATTRATTR
LAOSCM, LAOSCONCEPTMAP DM | GM | UM | PM | CM
CONCEPT Concept | “text”
ATTR Attribute | title | keywords | text | introduction | conclusion |
exercise | child | parent | Relatedness | ATTR.ATTR | CONCEPT.ATTR |
label | weight | “text”
PREDEFATTR
PREDEFATTRATTR
ATTRATTR type | order | next | ToDo | menu | show | access | visited | “text”
SPECCONMAP “text”
SPECCON “text”
SPECATTR “text”
OP = | += | -= | .=
COMPARE == | < | > | in
VALUE true | false | “text”

APPENDIX B: LAG Grammar Semantics

PROG: A LAG strategy or procedure, containing a set of instructions (programming constructs) defining the user and presentation adaptation in an adaptive hypermedia environment.

DESCRIPTION: The description of PROG; contains a natural language description of the behaviour of the adaptive strategy; it serves as the label (meta-description) for the whole strategy. It is important, as laic (nonprogrammer) authors should be able to extract from it the necessary elements to make a decision about using this adaptation or not.

VARIABLES: The variables of PROG; contains the list of variables that are used in the adaptive strategy. This information can be used by a laic (non-programmer) author to decide what attributes of the GM (goal and constraints model) should be filled-in for this strategy.

INITIALIZATION: The static initialization part of PROG; in this part, the initial experience of the user, when entering the adaptive environment, is described. This is useful so that a user doesn't enter a void environment. Here, all the default decisions are set. Adaptive environments which are adaptable but not adaptive can only render this part.

IMPLEMENTATION: The dynamic implementation part of PROG; in this part, the interactivity between the adaptive environment and the user is described (for instance, the effect of user clicks).

STATEMENT: The LAG language is a simple language built of a number of programming constructs, or statements, as follows:

IFSTAT: This statement is similar to IF statements in other programming languages, and is used for condition-action rules; the exact syntax is given in the grammar. This is the basic building block of the adaptation language. Any other (higher level) building block is translatable to it, as all adaptive hypermedia environments use this as the basis of adaptation.

WHILESTAT: This statement is similar to WHILE statements in other programming languages, and is used for loops; the exact syntax is given in the grammar.

FORSTAT: This statement is similar to WHILE statements in other programming languages, and is used for loops; the exact syntax is given in the grammar.

BREAKSTAT: This statement is similar to BREAK statements in other programming languages; the exact syntax is given in the grammar. It is used to exit a FOR or a WHILE loop. It is currently not available for the **MOT2AHA conversion**.

GENSTAT: This statement uses the hierarchical structure in the DM (domain model) and GM (goal and constraints model) for adaptive navigation. It specifies that more general concepts, higher in the hierarchy than the current concept, will be displayed to the user, given that the condition(s) is (are) fulfilled. It is currently not available for the **MOT2AHA conversion**; instead, the child-parent relation can be used.

SPECSTAT: This statement uses the hierarchical structure in the DM (domain model) and GM (goal and constraints model) for adaptive navigation. It specifies that more specific concepts, lower in the hierarchy than the current concept, will be displayed to the user, given that the condition(s) is (are) fulfilled. It is currently not available for the **MOT2AHA conversion**; instead, the child-parent relation can be used.

ACTION: This is part of the basic building block of condition-actions. It can be used by itself, as if the condition attached to it would be set to TRUE. This statement is the only one that allows specification of updates and changes of visible (such as the current screen) or invisible (such as the user knowledge) variables.

CONDITION: this part of a statement can appear in various constructs, for ultimately triggering condition action rules. A condition can be a change in a variable, but also an event, such as the user access to a certain concept. The condition is specified by a prerequisite, or a set of prerequisites, of which enough of them should be fulfilled.

Enough: this is a part of a condition statement; it specifies that a number of conditions should be fulfilled. How many of these conditions are enough is specified by the VALUE.

ATTRIBUTE: can appear in a condition or an action; can be a generic attribute of a DM (domain map), GM (goal and constraints map), UM (user map) or PM (presentation map) (e.g., DM.Concept.knowledge); or can be a specific attribute (e.g., '\Neural Networks Map\Learning\Introduction\Weight'.show). Generic attributes are used in adaptive strategies that can be applied on (almost) any DM and GM. When specific attributes are used, the whole strategy becomes only applicable on a specific, given DM and/or GM (such as in our example, only on the “Neural Networks Map” domain model).

ATTR: the variables in LAG follow the naming in LAOS (as shown by the syntax). There are a series of standard, predefined names, such as: title | keywords | text | introduction | conclusion | exercise | Relatedness | type (inherited from the DM – domain model); child | parent (inherited from both DM and GM – goal and constraints model); label| weight | order (inherited from the GM); next | ToDo | menu | show | access | visited (inherited from the PM – presentation model).

Title: title of a concept.

Keywords: keywords list of a concept.

Text: main body of a concept.

Introduction: introductory (hyper-)text of a concept.

Conclusion: concluding (hyper-)text of a concept.

Exercise: an exercise referring to the contents of a concept.

Relatedness: a (typed) relation of a concept with another concept, other than the hierarchical relation. Can be used to present all related concepts of the current concept.

Type: an attribute of a concept attribute (e.g., the attribute Title is of type ‘Title’). Can be used to present only attributes of a given type.

Child: any sub-concept of the given concept in the concept hierarchy (concepts below in this hierarchy), as well as all attributes of a group concept.

Parent: the super-concept of the given concept in the concept hierarchy (concept above in this hierarchy)

Label: the textual label used in the GM for giving extra presentational information about a concept (such as pedagogical information – e.g., this concept is for beginners, thus it is labeled ‘beg’)

Weight: the numerical label used in the GM for giving extra presentational information about a concept (such as pedagogical information – e.g., this concept is for readers of the short version of 30% of the material, thus it is labeled ‘30’); it can be used in combination with Label to give more detailed extra presentational behaviour.

Next: (currently,) it can be used (only) in the Initialization part to turn the “Next” button on or off. PM.GM.next = false turns it off (default is true). (for beginners, it might be enough to have a next button without other confusing menus or lists)

ToDo: (currently,) it can be used (only) in the Initialization part to make the “ToDo” list of concepts still to visit visible or not. PM.GM.ToDo = false turns it off (default is visible).

Menu: (currently,) it can be used (only) in the Initialization part to make the “Menu” hierarchical list of concepts visible or not. PM.GM.menu = false turns it off (default is visible). (for global learners, for instance, it is advisable to have a menu)

Show: this is a reserved variable (from the PM – presentation model) used to show a concept to the user, when true. By default, all concepts’ show variable is set to false, so they are invisible. If no concepts’ show variable is turned to true in the initialization part of the adaptive strategy, the user will enter a blank screen.

Access: this is a reserved variable (from the PM – presentation model) used to specify that a concept is currently accessed by the user.

Visited: this is a reserved variable (from the PM – presentation model) used to specify that a concept has been accessed by the user in the past.

APPENDIX C: LAG Extension

Aim: To extend the LAG language to enhance functionality, re-usability and compatibility of an Adaptive Specification authoring systems.

Triggers respond to Events - If certain conditions are met in the User Model or Goal Model then to display a notification to inform the user. Inspired by the computer games, where when a player finishes a level or achieves a high score, the game lets the user know about the change in the User Model.

LAG Validator - Similar to XML validator which validates a document against a definition file to check for invalid expression such as PM.PM.

Events - Related to user model, knowledge model or domain model, events can be used to update GM, Domain Model or User Model.

Time dependent tags - The users should get a chance to undo any adaptations, a time-delayed message for undo to give users the option for undo, like a message on the screen for undo action and then that message will disappear after a delay. This delay variable could be set in the user preferences.

Control variable list - a list of all the control variables within an adaptive strategy to provide a simple view of the dynamic element of a strategy.

Strategy searching for re-uses - To search for existing strategies (meta data, pedagogical listing) to select a strategy suitable for their required adaptation.

Simulate - LAG Strategy should have keywords for SIMULATE or VIEW to simulate how this strategy will be applied to different delivery parameters such as network condition, user groups, for instance, how this strategy will be viewed by beginners and advance users and what potential problem might occur. To make sure that all target users have access to the information and navigational elements being offered

APPENDIX D: Novice Group Evaluation

VASE (Visual Adaptive Strategy Environment) Evaluation

* 1. What is your current job?

- Student
- Company Worker
- Teacher

Other (please specify)

* 2. how much experience do you have in computer programming?

- None
- less than a year
- 1-3 year
- 4-6 years
- over 6 years

Other (please specify)

3. What is your age?

- Under 18 years
- 18-25 Years
- 26-35 years
- 36-50 years
- Over 50 years

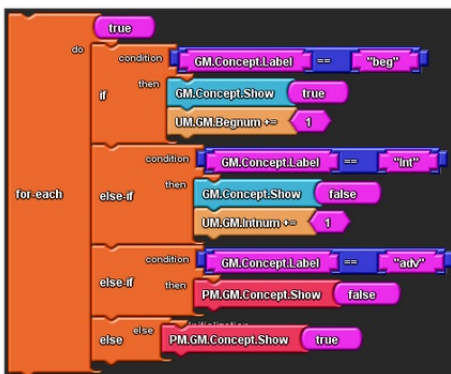
Other (please specify)

4. How much authoring experience do you have?

- None
- less than a year
- 1-2 years
- 3-6 years
- 7-10 years
- over 11 years

5. Here are two version of the same adaptive program, which one do you prefer?

(A)



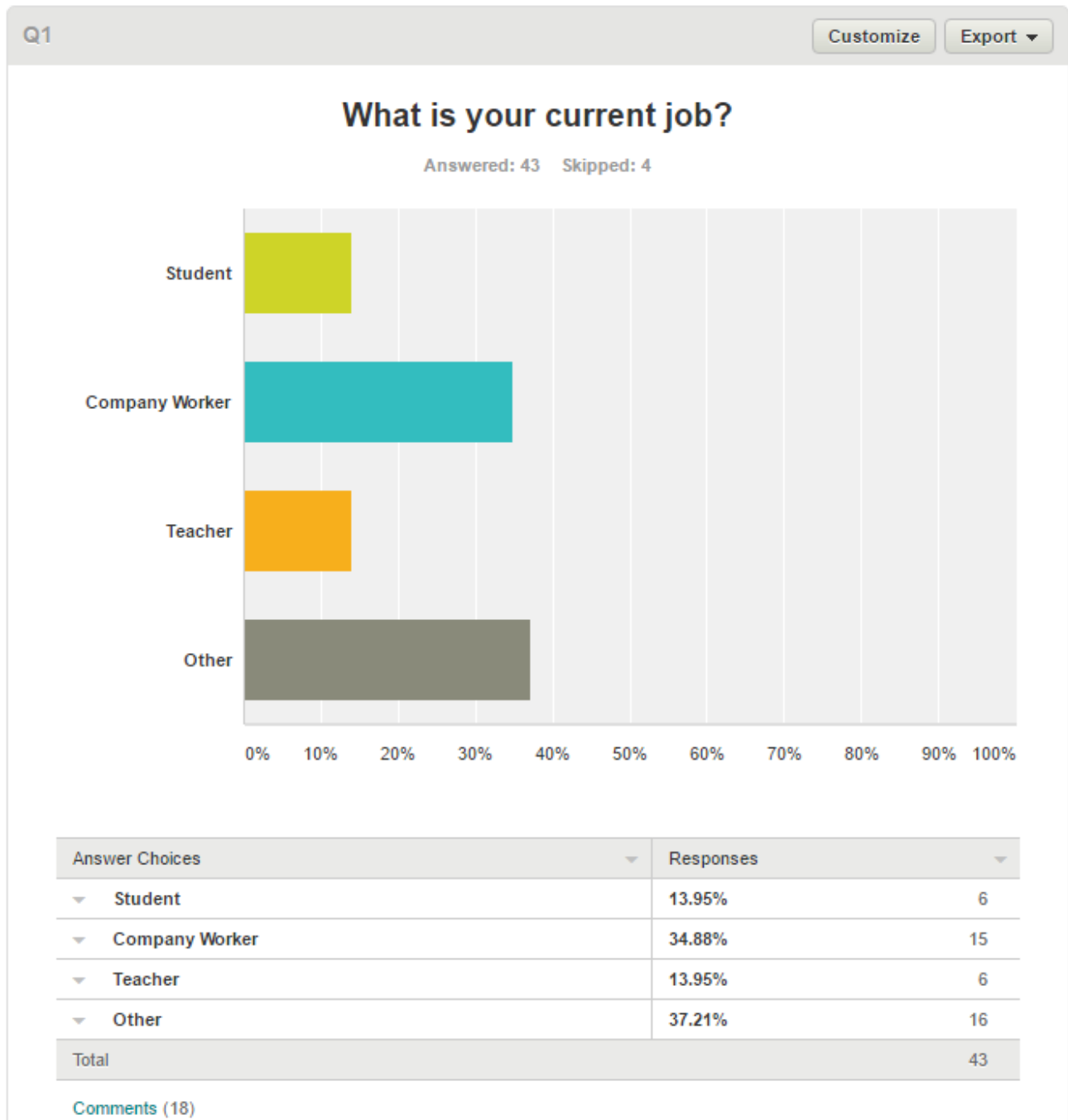
(B)

```

for-each (true) (
  if ( GM.Concept.label == "beg") then (
    PM.GM.Concept.show = true
    UM.GM.begnum += 1
  ) elseif (GM.Concept.label == "int") then (
    PM.GM.Concept.show = false
    UM.GM.intnum += 1
  ) elseif (GM.Concept.label == "adv") then (
    PM.GM.Concept.show = false
  ) else (
    PM.GM.Concept.show = true
  )
)
    
```

- Prefer A
- Prefer B
- Neither

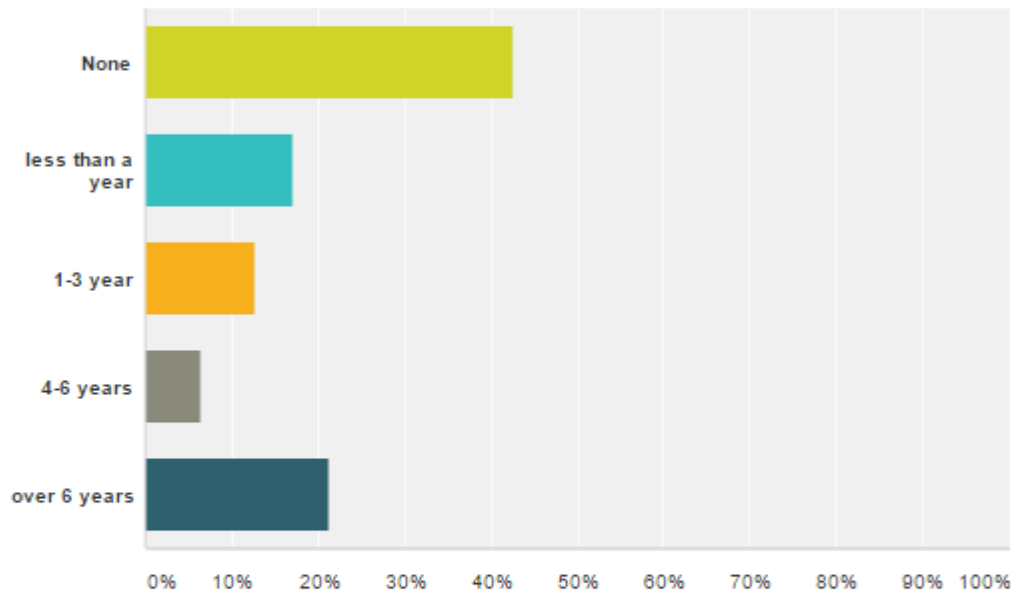
APPENDIX E: Non-professional programmer evaluation results





how much experience do you have in computer programming?

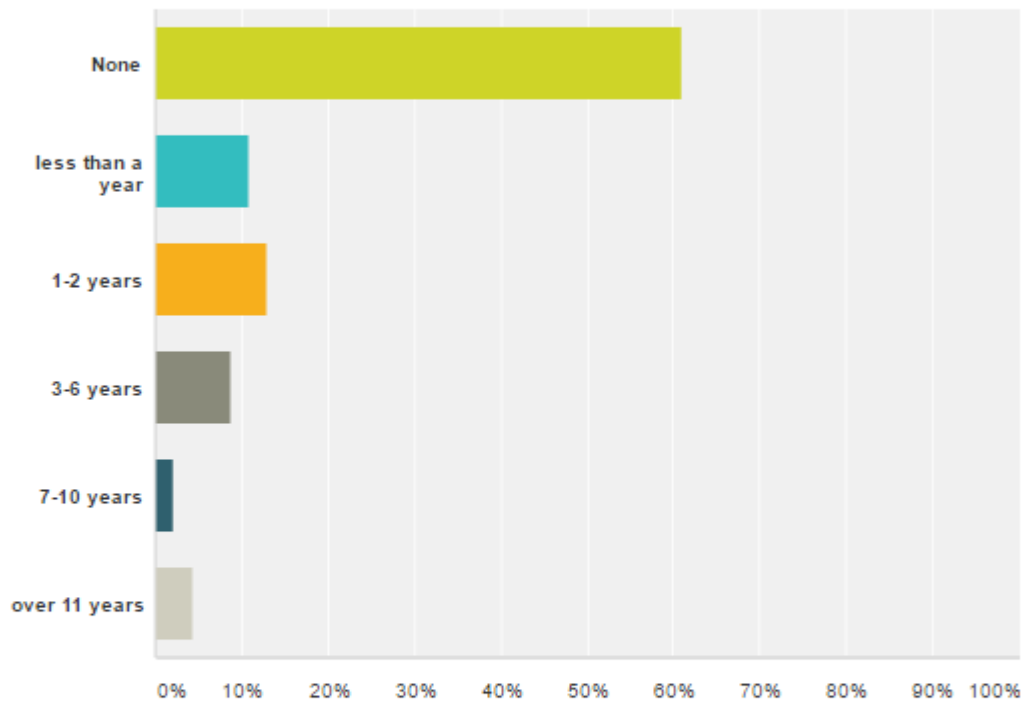
Answered: 47 Skipped: 0



| Answer Choices | Responses |
|------------------|-----------|
| None | 42.55% 20 |
| less than a year | 17.02% 8 |
| 1-3 year | 12.77% 6 |
| 4-6 years | 6.38% 3 |
| over 6 years | 21.28% 10 |
| Total | 47 |

How much authoring experience do you have?

Answered: 46 Skipped: 1

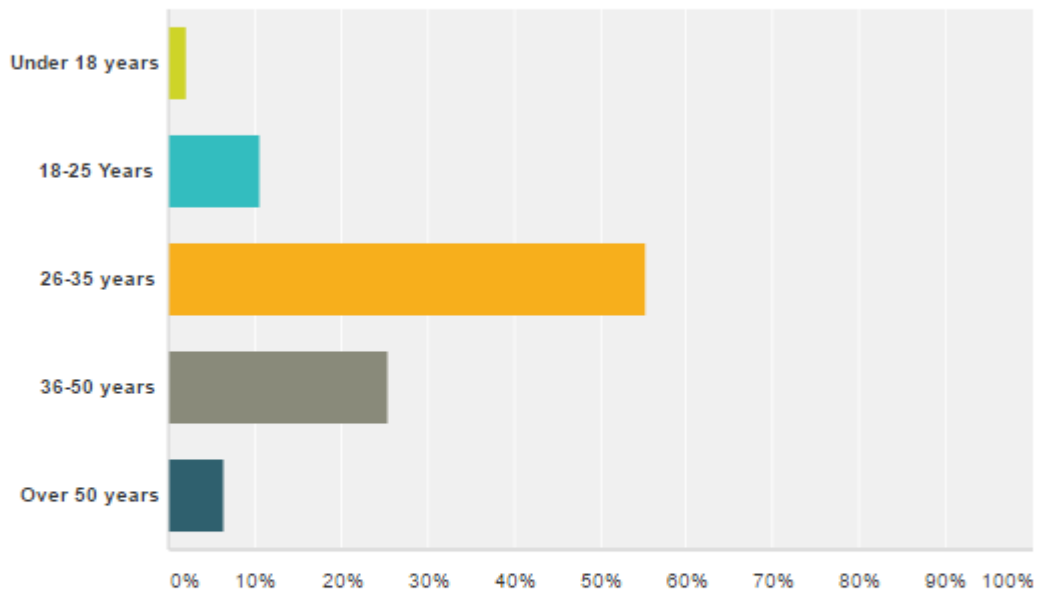


| Answer Choices | Responses |
|------------------|-----------|
| None | 60.87% 28 |
| less than a year | 10.87% 5 |
| 1-2 years | 13.04% 6 |
| 3-6 years | 8.70% 4 |
| 7-10 years | 2.17% 1 |
| over 11 years | 4.35% 2 |
| Total | 46 |



What is your age?

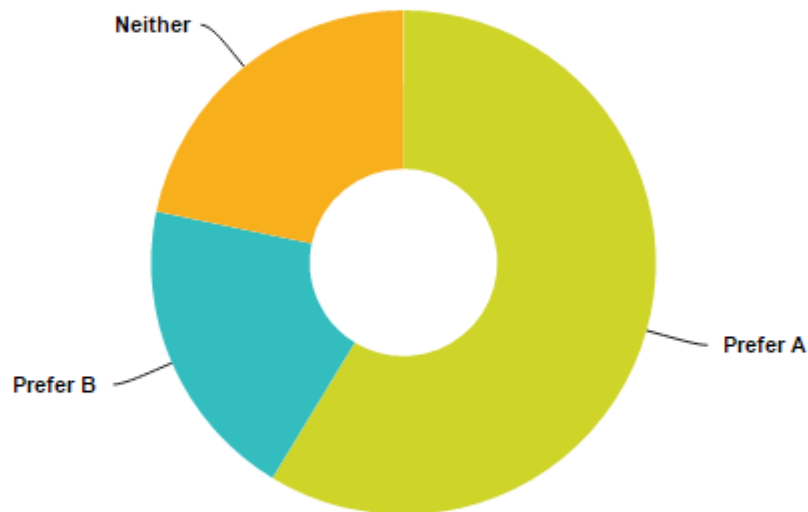
Answered: 47 Skipped: 0



| Answer Choices | Responses |
|------------------|-----------|
| ▼ Under 18 years | 2.13% 1 |
| ▼ 18-25 Years | 10.64% 5 |
| ▼ 26-35 years | 55.32% 26 |
| ▼ 36-50 years | 25.53% 12 |
| ▼ Over 50 years | 6.38% 3 |
| Total | 47 |

Here are two version of the same adaptive program, which one do you prefer?

Answered: 46 Skipped: 1



| Answer Choices | Responses |
|----------------|-----------|
| ▼ Prefer A | 58.70% 27 |
| ▼ Prefer B | 19.57% 9 |
| ▼ Neither | 21.74% 10 |
| Total | 46 |

APPENDIX F: The CAF Language Format

<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT CAF (domainmodel?, goalmodel?)>

<!ELEMENT domainmodel (concept+)>

<!ELEMENT concept (name, attribute*, concept*)>

<!ELEMENT attribute (name, contents)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT contents (#PCDATA)>

<!ATTLIST contents

weight CDATA ""

label CDATA ""

>

<!ELEMENT goalmodel (lesson)>

<!ELEMENT lesson (contents*, lesson*)>

APPENDIX G: ADE 0.2 and LAG 4.0 Demonstration Courses

Hide After Multiple Attempts

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(CAF\)](#)

This strategy initializes the content by displaying everything. When each concept is accessed for a second time, the concept is hidden from the MENU, TODO and NEXT UI areas.

The reason for specifying the MENU, TODO and NEXT areas and not using just PM.GM.Concept.show = false is because the LAG implementation loop is applied before the content is displayed to the user, so the user would just be shown a blank page. Hiding the concept from the three navigational areas and not the CONTENT UI area as well allows the user to still view the content but not reaccess it later from the menus.

This strategy would work with any course content that it is applied to as it is completely generic.

Positions

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(CAF\)](#)

This course hides Goal Model concepts from all navigational menus at the start and then displays them in the navigational elements based on their labels.

Once they have been accessed once, the strategy moves them to the main tree menu.

This strategy will only work with content files that are labelled with 'todo', 'next' or 'menu' as unlabelled content would never be displayed in the navigational menus.

Sorting

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(LAF\)](#)

The sorting strategy initializes a course by setting the ‘order’ Presentation Model variable of each concept to the ‘sort’ label weight if it exists for that concept.

This results in those concepts being displayed in sorted alphanumeric order by the ‘order’ variable when the lesson they are part of is viewed.

After any concept has been accessed more than once, the ‘order’ variable is set to 99, displaying the content in the unsorted order again.

This strategy would work with any course content that has ‘sort’ labels applied to some of the content. On any other content, it will be equivalent to the [Show All](#) strategy.

View and Move

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(CAF\)](#)

This strategy hides the Next menu UI section and only shows content in the Todo list. As each concept is accessed, it is added to the hierarchical tree menu.

The strategy works with all course content as it is a generic strategy.

Relations

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(LAF\)](#)

The Relations strategy shows all content except the Goal Model Concepts that link to Domain Model Attributes that are children of a ‘preReqOf’ relationship. In short, if content has been enriched with prerequisite relationships, then content will not be shown until it’s prerequisite concept has been accessed.

This is identical to the [Show All](#) strategy if no concepts have ‘preReqOf’ relationships.

Dimming

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(LAF\)](#)

The dimming strategy initializes a course by checking concepts in the Goal Model to see if the label ‘dim’ is set to 1, setting those concepts ‘dim’ variable in the Presentation Model to true.

This results in those concepts being displayed as dimmed text when the lesson they are part of is viewed.

After any concept has been accessed more than once, the ‘dim’ variable is set to false, displaying the content as normal text again.

This strategy would work with any course content that has ‘dim’ labels applied to some of the content. On any other content, it will be equivalent to the [Show All](#) strategy.

Beginner-Intermediate-Advanced

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(CAF\)](#)

A User Model variable called ‘knowlvl’ is used to track the knowledge level of the user in this strategy. It can be set to ‘beg’, ‘int’ or ‘adv’. The idea is that content within the course is labelled as ‘beg’, ‘int’ or ‘adv’ and then only ‘beg’ is displayed to beginners, ‘beg’+‘int’ shown to intermediates and ‘beg’+‘int’+‘adv’ shown to advanced users.

This is done by counting the number of ‘beg’ concepts and storing this in the ‘begnum’ User Model variable. This variable is decremented as the beginner content is accessed until all beginner concepts have been viewed. The intermediate content is then displayed which has a similar variable called ‘intnum’ and the user is set to ‘int’.

All non ‘beg’/’int’/’adv’ labelled content is shown from the start so a course would be displayed identically to the [Show All](#) strategy if those labels were not used.

End Of Course Message

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(CAF\)](#)

When this course is first accessed a 'todoCount' variable is created in the User Model and incremented by the number of concepts in the course. On each concepts first access, this variable is decremented by 1.

When the 'todoCount' reaches 0, all the course content has been accessed and the layout of the course is changed. The navigational menu on the left of the UI is changed to the Warwick University logo and the Next section at the bottom of the UI displays a 'COURSE FINISHED' message.

This strategy would work with any course content that it is applied to as it is completely generic.

Q&A

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(LAF\)](#)

The course shows all content initially apart from Goal Model concepts that link to Domain Model attributes of type 'answer'. These concepts are hidden initially.

As each lesson is accessed the concepts making up the lesson have a variable called 'lessonSeen' incremented including the hidden concepts. Once this is incremented past 1 then the concepts Presentation Model 'show' variable is set to true. This has the effect of showing the answers to the questions in the demo the second time around.

This strategy works best with content which has some content labelled with 'answer'. Otherwise it is identical to the [Show All](#) strategy.

Rollout

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(CAF\)](#)

This course uses the weights on concepts labelled 'showafter' or 'showatmost' to display or hide content. For example, a concept labelled 'showafter' with a weight of 5 will be shown after the lesson has been accessed 5 times. If a concept is labelled 'showatmost' and labelled with a weight of 5 will be shown until it has been accessed 5 times.

The strategy will have the same effect as the [Show All](#) strategy unless the course is labelled with 'showafter' and 'showatmost' with weights.

Show All

[VIEW COURSE](#) | [VIEW STRATEGY](#) | [VIEW CONTENT FILE \(CAF\)](#)

The strategy initializes the course by displaying everything in the course.

APPENDIX H: LAG Strategies

Beginner-Intermediate-Advanced

```
// Beginner > Intermediate > Advanced
// Works with any CAF/LAF file with "beg", "int" and "adv" labels
```

```
initialization(
```

```
    PM.next = true
    PM.ToDo = true
    PM.menu = true
```

```
for-each (true) (
    if ( GM.Concept.label == "beg") then (
        PM.GM.Concept.show = true
        UM.GM.begnum += 1
    ) elseif (GM.Concept.label == "int") then (
        PM.GM.Concept.show = false
        UM.GM.intnum += 1
    ) elseif (GM.Concept.label == "adv") then (
        PM.GM.Concept.show = false
    ) else (
        PM.GM.Concept.show = true
    )
)
```

```
    UM.GM.knowlvl = beg
)
```

```
implementation (
```

```
for-each ( UM.GM.Concept.access == true ) (
    if (UM.GM.Concept.accessed == 1) then (
        if (GM.Concept.label == beg) then (
            UM.GM.begnum -= 1
        ) elseif (GM.Concept.label == int) then (
            UM.GM.intnum -= 1
        ) elseif (GM.Concept.label == adv) then (
            UM.GM.advnum -= 1
        )
    )
)
```

```
if (UM.GM.begnum < 1 and UM.GM.knowlvl == beg) then (
    UM.GM.knowlvl = int
    PM.GM.Concepts[GM.label == UM.GM.knowlvl].show = true
) elseif (UM.GM.intnum < 1 and UM.GM.knowlvl == int) then (
    UM.GM.knowlvl = adv
    PM.GM.Concepts[GM.label == UM.GM.knowlvl].show = true
```

```
)  
)
```

Dimming

```
// Text Dimming  
// Works with any CAF/LAF file with "dim" labels
```

```
initialization (  
  for-each ( true ) (  
    PM.GM.Concept.show = true  
    if ( GM.Concept.Labels['dim'].value == 1 ) then (  
      PM.GM.Concept.dim = true  
    )  
  )  
)
```

```
implementation (  
  PM.GM.Concepts[accessed>1&&access==true].dim = false  
)
```

End Of Course Message

```
// End of Course Message  
// Works with any content file
```

```
initialization (  
  for-each true (  
    PM.GM.Concept.show = true  
    UM.GM.todoCount += 1  
  )  
)
```

```
implementation (  
  for-each ( PM.GM.Concept.access == true and UM.GM.Concept.accessed == 1 ) (  
    UM.GM.todoCount -= 1  
    if ( UM.GM.todoCount < 1 ) then (  
      Layout[W].type = text  
      Layout[W].content = "<img src='http://www2.warwick.ac.uk/services/communications/corporate/downloads/img/the_warwick_uni_black.jpg' width=180>"  
      Layout[S].type = text  
      Layout[S].content = "<b>COURSE FINISHED</b>"  
    )  
  )  
)
```

Hide After Multiple Attempts

```
// Hide after multiple accesses
// Runs on any CAF/LAF file

initialization(
  PM.GM.Concepts.show = true
)

implementation (
  for-each UM.GM.Concept.accessed > 1 (
    PM.MENU.GM.Concept.show = false
    PM.TODO.GM.Concept.show = false
    PM.NEXT.GM.Concept.show = false
  )
)
```

Positions

```
// Positions
// Needs a specific content file
initialization (
  for-each (true) (
    PM.GM.Concept.showContent = true
    PM.TODO.GM.Concept.show = false
    PM.NEXT.GM.Concept.show = false
    PM.MENU.GM.Concept.show = false

    if (GM.Concept.label == menu) then
      (
        PM.MENU.GM.Concept.show = true
      )
    if (GM.Concept.label == todo) then
      (
        PM.TODO.GM.Concept.show = true
      )
    if (GM.Concept.label == next) then
      (
        PM.NEXT.GM.Concept.show = true
      )
    )
  )
)
implementation (
  for-each (GM.Concept.label == todo)
  (
    PM.TODO.GM.Concept.show = true
  )
)
```

```

for-each (GM.Concept.label == next) [68]
(
  PM.NEXT.GM.Concept.show = true
)
for-each (UM.GM.Concept.accessed > 0)
(
  PM.MENU.GM.Concept.show = true
)
)

```

Q&A

```

// Q&A
// Runs on a specific setup of content file

initialization (
  PM.GM.Concepts.show = true
  PM.GM.Concepts[GM.type=='answer'].show = false
)

implementation (
  for-each ( PM.GM.Concept.access==true ) (
    PM.GM.Concept.lessonSeen += 1
    if ( PM.GM.Concept.lessonSeen > 1 ) (
      PM.GM.Concept.show = true
    )
  )
)
// Could be written as
// PM.GM.Concepts[access==true].lessonSeen += 1
// PM.GM.Concepts[lessonSeen>1&&access==true].show = true
)

```

Relations

```

// Relatedness Example
// Runs on any LAF file with Relations with "preReqOf" relations specified

initialization (
  PM.GM.Concepts.show = true
  PM.GM.Concepts.Relations['preReqOf'].show = false
)

implementation (
  for-each PM.GM.Concept.access==true (

```



```

    PM.GM.Concept.Relations['preReqOf'].show = true
  )
// Could be written as
// PM.GM.Concepts[access==true].Relations['dependsOn'].show = true
)

```

Rollout

```

// Rollout
// Runs on any CAF/LAF file with showafter and showatmost labels

```

```

initialization(

```

```

    UM.GM.Concepts.beenthere = 0
    PM.GM.Concepts.show = true

```

```

    for-each GM.Concept.label == showafter (
      if GM.Concept.weight > 1 then (
        PM.GM.Concept.show = false
      )
    )

```

```

)

```

```

implementation (

```

```

    for-each UM.GM.Concept.access == true (
      UM.GM.Concept.beenthere += 1
    )

```

```

    for-each enough(UM.GM.Concept.beenthere >= GM.Concept.weight
      GM.Concept.label == showatmost
      ,2) (
      PM.GM.Concept.show = false
    )

```

```

    for-each enough(UM.GM.Concept.beenthere >= GM.Concept.weight
      GM.Concept.label == showafter
      ,2) (
      PM.GM.Concept.show = true
    )

```

```

)

```

Show All

```
// Show All
// Runs on any CAF/LAF file

initialization(
  PM.Next = true
  PM.ToDo = true
  PM.Menu = true
  PM.GM.Concepts.show = true
)

implementation ( )
```

Show unlabelled content first then all

```
// Show unlabeled content first
// Runs on any CAF/LAF file with a mix of labeled and unlabeled content

initialization(
  PM.GM.Concepts[GM.label==""].show = true
  for-each PM.GM.Concept.show == true (
    UM.todoCount += 1
  )
)
```