Adaptive Hypermedia: A simple way to create Adaptive Strategies using LAGBlocks

Javed Khan and Alexandra I. Cristea
Computer Science, University of Warwick,
Coventry, CV4 7AL , United Kingdom

Abstract

Adaptation specifications are very useful in creating adaptive behaviors to support the needs of the learners, authors often lack the time or the skills needed to create new adaptation specification from scratch. Meaningful adaptation specification can only be created by using an adaptive programming language such as LAG. Creating adaptation specification requires the author to know and remember the programming language syntax. This places a knowledge barrier for the author.

In this article we propose a visual framework (LAGBlocks) for LAG adaptation language and an authoring tool (VASE), to create adaptive specification by manipulating LAGBlocks. Thus, we aim to make adaptive specifications (rules) easier to create and to share for authors with little programming knowledge.

Authors are not forced to learn and recall confusing syntax, helping to reduce the learning curve. These pieces can be dragged and joined with mouse with very little programming knowledge.

Keywords: ***Adaptive Educational Hypermedia, Adaptive Authoring Tool, Adaptation Specification***

1. Introduction

Adaptive Educational Hypermedia (AEH) improves the usability of hypermedia, by building a model of various aspects of a learner and uses this information to adapt the content and the navigation to the needs of the learner. Adaptive strategies are sets of adaptive rules to specify the conditions under which adaptation behaviors can occur. The AEH approach has been shown to be useful [1] as it displays more relevant content according to the information stored in various models (user, goal and presentation model. However, authoring and creation of hypermedia is not trivial at all. Unlike in traditional authoring for hypermedia, a linear path is not enough. Instead, various alternatives have to be created for the given material. For example, in order to create a personalized, rich 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 needs to be added for labeling 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 [2].

There are only a small number of tools available which can provide authoring of adaptive strategies in LAG programming language. MOT [3] is one of such tool that provide a generic platform for authoring and reusing adaptive strategies. Since there are no standards for adaptation, we will use one of the candidates, the LAG adaptation language [4].

This means that any system that can import LAG language will be compatible with VASE (Visual Adaptive Strategy Environment) authoring system for adaptive hypermedia. MOT supports two export formats, CAF [6] and the LAG [7] language. These formats present a compact way of storing and exchanging adaptive hypermedia information between different Adaptive Educational Hypermedia Systems. CAF is an XML representation of the content and lesson structure. The LAG programming language interacts with the User and the Presentation Models to adapt the course (the Domain and Goal Models) to a particular user.

MOT3.0 [3] uses the PEAL (Programming Environment for Adaptation Language) [8] to write adaptive strategies, in which authors can use an adaptation language (LAG [7]) to program adaptive behaviors. Overall, PEAL was found to achieve its goals, the evaluation of PEAL [8] found that some further common programming environment features would have made it even more convenient. However, non-programmers and programmers new to LAG and the MOT system found programming in the LAG language intimidating [8].

Learning to program can be a difficult task. In addition to learning confusing and non-intuitive syntax, authors must learn how to structure their thinking and understand the execution of their program.

We have developed an adaptive authoring environment called VASE (Visual Adaptive Strategy Environment) to create adaptation specification rules visually for authors with little or no programming experience, to aid authors in the creation of adaptation specification. VASE will allow users to visually create adaptation specifications (adaptive rules) by dragging and connecting puzzle-pieces of LAG language constructs. This means that the users would not need to remember LAG programming syntax in order to create adaptive specification in LAG, thus it lowers the threshold to programming.

VASE applies many useful programming environment features to ease the development in LAG (adaptation language). *User Roles* are used to display relevant options for the individual user [5], to limit the displayed options to the user according to their role experience. The ongoing development on this authoring tool means that it has user-roles to a show simplified view of the system to the users according to their knowledge and experience about the system.

This research aims to create, share and re-use existing adaptation specification by providing a visual environment for creating, editing and sharing adaptation specification. This will allow an author to visually create adaptation strategies with little knowledge of LAG adaptation language, thus lowering the threshold programming knowledge requirements to create adaptation specification.

2. Related Work

Continuous research [8c], [15c], [9c], [11c] on better authoring tools has resulted in several advanced authoring tools. Recent developments include the MOT 2.0 [9], the MOT 3.0[3] and the GRAPPLE [10] authoring tools. However, authoring of adaptive strategy remains a difficult and time consuming task for adaptive hypermedia author.

GRAPPLE

The Generic Responsive Adaptive Personalized Learning Environment (GRAPPLE) [10] is an enhanced learning environment that guides learners through the learning experience, automatically adapting to personal preferences, prior knowledge and learning goals. GRAPPLE includes authoring tools to create adaptive learning material for the learners. The Grapple Authoring Toolset (GAT) has three main components: a Domain Model authoring tool (DM), for creating a conceptual representation of an application domain (or "course"), a Pedagogical Relationship 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.

MOT

My Online Teacher (MOT) [3], [9] is an advanced adaptive authoring system based on the LAOS framework [4a]. MOT3.0 [3] uses the PEAL system [8] as an adaptive strategy editor, in which authors can use an adaptation language (LAG [7]) to create adaptation behaviors (strategies) for the AEH. The separation of different tasks into different tools is known as the 'separation of concerns' [11], and is useful to promote the reuse of static and dynamic materials. The separation allows the two parts to be authored by different author roles [12].

This is vital if the aim is not only to simplify the authoring process but to spread the authoring load and encourage reuse.

Graphical Programming Systems

Graphical 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.

Many graphical programming systems exist 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

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:

- **Forgiveness -** 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.

OpenBlocks Design

StarlogoTNG [13] is a graphical programming environment for secondary students and teachers to study and create 3D simulations and games. It only requires users to connect puzzle-piece like objects called blocks of varying shapes and colours to build their program

OpenBlocks [14] is an open-source Java library for creating user-interface (UI) elements for blocks-based programs. The design of *OpenBlocks* is inspired by the design of StarLogoTNG[12].

OpenBlocks consists of two packages, *CodeBlocks* & *Slcodeblocks*. *CodeBlocks* is the basic underlying library that is responsible for most of the functionality. *Slcodeblocks* is code from StarLogoTNG [ref] project, which uses and extends *CodeBlocks* library to fully implement the *StarLogoTNG's* UI.

3. Motivation Scenario

Visual programming elements have the potential to express richer semantics to the user than text-based programming elements. Interactive feedback can also be applied to the visual environment to better support the user throughout the system.

We have created VASE, a visual authoring tool designed for authors with little or no programming experience. We have extended OpenBlocks framework [14], to create visual representation of LAG programming language called (LAGBlock). LAGBlocks represents entire LAG language visually with LAG’s data types and constructs. LAG has three data types, each represented by a unique shape, giving clues on how these will fit together.

In addition to VASE, we have also created a visual-to-LAG converter, to convert visual adaptive strategy back to LAG code so the adaptive strategies created visually using VASE could be converted into LAG code, to be compatible with the tolls which support LAG adaptation language such as ADE and MOT.

4. The Proposed Solution

We have created the following components to help the author in adaptive strategy creation:

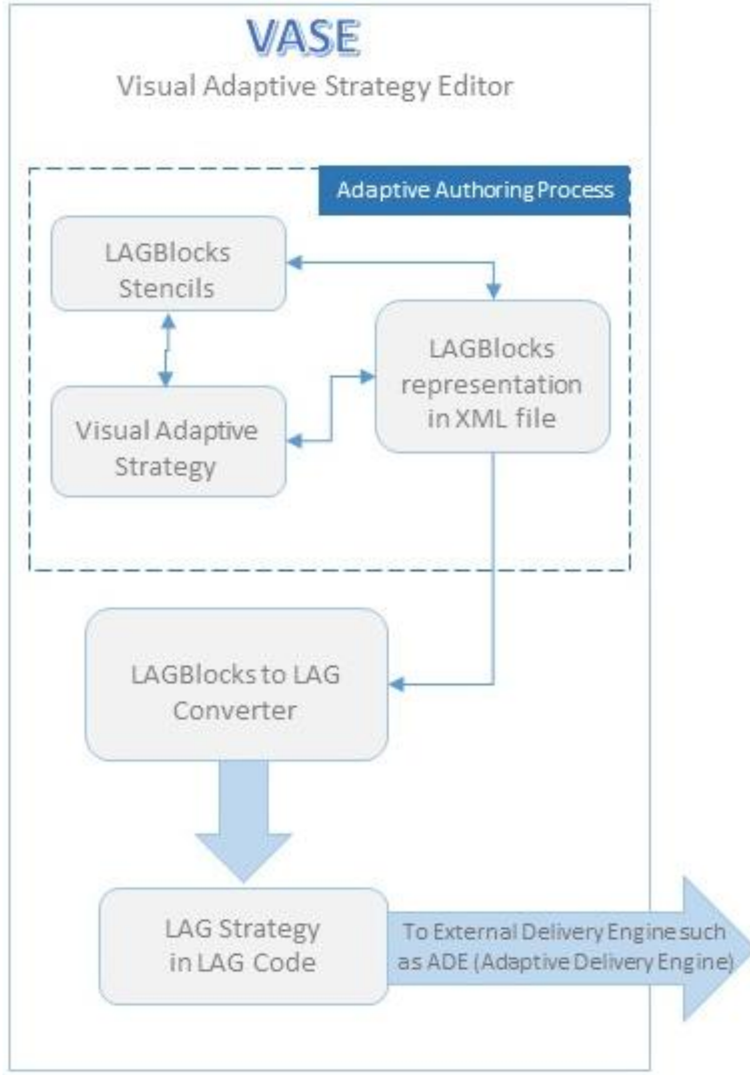
- LAGBLocks
- VASE (Visual Adaptive Strategy Environment)
- LAGBlocks to LAG converter

LAGBlocks
Are a visual extension for LAG language to represent any LAG statement in a visual form. The purpose of LAGBlocks was to allow Adaptive Hypermedia authors to create adaptive strategy without having to type any LAG programming statements.

VASE
The authoring environment which lets the author manipulate LAGBlocks is called VASE, short for **V**isual **A**daptive **S**trategy **E**nvironment. Main aim of VASE was to make programming more accessible to authors with little or no programming experience.

Existing adaptive strategies would be edited visually by moving LAGBlocks around by mouse, to alter the logic of adaptive strategy. The LAGBlocks which have matching connectors can only fit with other matching ones, this provides code *correctness*. An auto-complete feature has been implemented to provide code *completeness*.

Creating and editing adaptive strategies visually has shown to be useful. This new extension can represent any LAG programming language statement. This extension will allow authors to create adaptation specification (behaviors) visually, without having to type/remember any LAG programming syntax. Advance users will be able to write LAG code, however, this will be controlled via the user-roles to enable system features for a group of users.



The shape of a construct gives clues on how to use the construct. For instance, a block will only fit another block which is compatible with its connector.

The LAG programming language syntax is represented in a XML file so the visual elements can be used instead of writing LAG programming syntax.

This visual representation of adaptive strategy is converted to LAG programming language syntax behind the scene for inter-operability between the tools which use LAG as the adaptation language such as ADE (Adaptive Delivery Engine) [15] and MOT (My Online Teacher) [9].

5. Implementation

LAGBlocks Function

Proposed shape for each data type in LAG

a. 

for LAG string type

b. 

for LAG Boolean type

c 

for LAG int type

Following new LAGBlocks construct have been created:

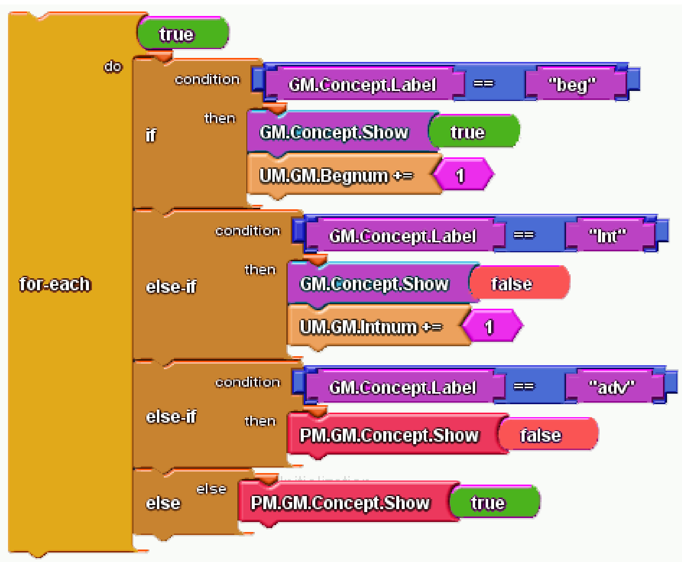
- LAG-Blocks Logic - to represent LAG if statement.
  LAG-Blocks User - to represent LAG User Model.
- LAG-Block Presentation Blocks - to represent LAG presentation model.
- LAG-Blocks Goal Model - to represent LAG goal model.

We have extended OpenBlocks to create visual representation of LAG programming language called (LAG-Block) to represent all of the LAG's data types and constructs. LAG has three data types, each represented by a unique shape, giving clues on how these will fit together. This has shown to be useful while creating and sharing adaptive strategies.

To save time, existing LAG strategies can be created using LAG-Blocks, the authors can edit strategies just by selecting it from a list, instead of building it from scratch.

New adaptive specification can be created by mixing and merging existing ones, authors don't have to create strategy from scratch. New strategies can be created with ease, by opening existing strategies and to save as a new strategy. We were able to create a visual representation for twelve adaptive strategies already written in LAG. This will encourage non-computer savvy authors to create and to reuse existing adaptive strategies.

Figure 1.2 shows a visual *for-each* adaptive strategy in LAG Blocks, a block programming environment where each block represents a LAG command. The shapes and colors of blocks also provide a visual overview of code, allowing users to more easily browse and follow the pathways of their code.



VASE User-Interface

The *VASE* 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

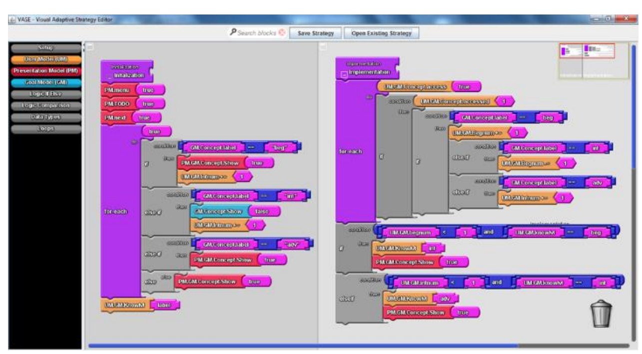


Fig: 1.4 Screenshot of VASE (Visual Adaptive Strategy Environment)

LAG-Blocks Manipulation in VASE

In LAGBlocks, users manipulate and connect puzzle-piece like objects to build their programs.

Users can manipulate and connect LAGBlocks to build adaptive strategies. LAGBlocks are manipulated directly by the user via mouse click and drag-n-drop actions. To encourage users to directly interact with blocks with their mouse, blocks can be dragged and dropped. Each connector shape will only connect with its matching connector shape.

The shape of these blocks dictates the syntax of the language, only blocks with complementary shapes can connect together. In addition, allowing only complementary blocks to connect and to prevents users from making syntax errors.

LAGBlocks to LAG Converter

LAGBlocks converter is used to convert visual representation of adaptive strategy into LAG programming code so it can be used by all programs which are compatible with LAG programming language. For instance, it can be delivered to students via the Adaptive Delivery Engine [15].

LAGBlocks converter can be used to export into different programming language, making our authoring tool interoperable authoring environment.

4. Conclusions

Adaptive Hypermedia (AH) [1] allows the presentation of hypermedia content to be personalized to the user, dependent on information about the user such as knowledge and preferences. An Adaptive Hypermedia System (AHS) needs to store and process content, user models and an adaptation specification [2] [3] [4]. A User Model (UM) stores data about the user, including preferences, current knowledge levels etc. User Models are updated during the interaction with the system. Research [3], [7], [8], [9] has shown that AEHS include adaptive functionality based on three components: the document space, observations, and the user model.

In this paper we describe the research towards improving the usage, sharing and creation of adaptation specification (adaptive rules) for adaptive hypermedia, specifically the design and implementation of a tool aimed at weak programmers or non-programmers. To make our visual extension to be more interoperable, the visual language definition are stored in XML file which is then converted by our LAG converter to create LAG code. This is to ensure that our authoring tool can work with other tools which use LAG as the adaptation language. For instance, ADE (Adaptive Delivery Engine) [15]

We have proposed a visual programming paradigm for LAG programming language for creating adaptation specification by simply dragging and connecting visual elements. This allows the inexperienced author to create adaptation specification with little knowledge of LAG programming language.

We have demonstrated that a visual extension for LAG programming language can lower the threshold for creating adaptive specification. We have shown that non-computer savvy author can quickly learn to use our LAGBlocks visual extension to create and edit existing strategy without having to remember or type LAG language code.

Acknowledgments

I would like to thank the 2 anonymous subjects from Warwick University MOAC Department, who took part in our study. The co-operation was organized through MOAC, Doctoral Training Centre, at the University of Warwick.

Thanks to Andrew Staley from The Energy Brokers

Ltd, Mattia Bailoni and Martha Bailoni from the University of Leicester, also took part in our study.

References

[1] Brusilovsky, P. (2001) Adaptive hypermedia, User Modelling and User Adapted Interaction, Ten Year Anniversary Issue, 11 (1/2), 87-110.

[2] Alexandra I. Cristea, David Smits, Jon Bevan, and Maurice Hendrix. Lag 2.0: Refining a reusable adaptation language and improving on its authoring. In EC-TEL, pages 7–21, 2009.

[3] Foss, J.G.K. & Cristea, A.I., The next generation Authoring Adaptive Hypermedia: Using and Evaluating the MOT3.0 and PEAL tools, HT '10 Proceedings of the 21st ACM conference on Hypertext and hypermedia
2010, ISBN: 978-1-4503-0041-4, pp 83-92

[4] F. Ghali, "Social personalized e-learning framework," University of Warwick, 2010.

[5] Khan J. A, Cristea A.I, & Stewart, C. (2011) "Adaptive Authoring of Adaptive Hypermedia towards, Role-based, Adaptive Authoring", CATE 2011, pp734-042. Publication date July 2011

[6] Alexandra I. Cristea, David Smits, and Paul de Bra. Towards a generic adaptive hypermedia platform: a conversion case study. Journal of Digital Information, 8(3), 2007.

[7] Cristea, A.I, Smits, D., Bevan, J. & Hendrix, M.: LAG 2.0: Refining a Reusable Adaptation Language and Improving on Its Authoring. EC-TEL 2009: pp7-21

[8] 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, p. 43.

[9] Ghali, F. and Cristea, A. MOT 2.0: A Case Study on the Usefulness of Social Modeling for Personalized E-Learning Systems. Frontiers in Artificial Intelligence and Applications, Volume 200, 2009 Artificial Intelligence in Education - Building Learning Systems that Care: From Knowledge Representation to Affective Modelling, ISBN 978-1-60750-028-5, pp 333 - 340, DOI: 10.3233/978-1-60750-028-5-333.

[10] GRAPPLE, [Online] http://www.grapple-project.org/summary (last accessed 05/05/2013)

[11] Hendrix, M., & Cristea, A.I. (2008) A Spiral Model for Adding Automatic, Adaptive Authoring to Adaptive Hypermedia. J. UCS 14(17): 2799-2818

[12] Scotton, J.D. & Cristea, A.I. (2010) Reusing Adaptation Strategies in Adaptive Educational Hypermedia Systems. In: The 10th IEEE International Conference on Advanced Learning Technologies.

[13] "Starlogo TNG" 2007 [online] available: http://education.mit.edu/starlogo-tng

[14] Roque, R. V. (2007). OpenBlocks: an extendable framework for graphical block programming systems (Doctoral dissertation, Massachusetts Institute of Technology).

[15] Scotton, J., Stewart, C., & Cristea, A. I. (2011). ADE: The Adaptive Display Environment for Adaptive Hypermedia. In Proceedings of the ACM Hypertext 2011 International Conference, Eindhoven, The Netherlands.