ALAT: Finally an Easy To Use Adaptation Authoring Tool

Paul De Bra, Natalia Stash,
Wouter Boereboom, Celine Chen
Dept. of Math. and Computer Science
Eindhoven University of Technology (TU/e)
Eindhoven, the Netherlands
{p.m.e.d.bra, n.v.stash}@tue.nl,
{w.boereboom, j.chenyuexu}@gmail.com

Joris Den Ouden, Martijn Kunstman,
John Oostrum, Egon Verbakel
de Roode Kikker
Eindhoven, the Netherlands
{j.den.ouden, m.kunstman, j.oostrum,
e.verbakel}@deroodekikker.nl

ABSTRACT

Research papers about adaptive hypermedia systems, frameworks or applications tend to focus on the end-result: how the applications are used by end-users, how adaptation improves user satisfaction, learning, etc. What they do not describe is how difficult and labor-intensive the creation of the applications can be. In this paper we present ALAT, a new authoring tool for the Generic Adaptation Language and Engine GALE, developed at the TU/e. ALAT is specifically designed in close collaboration with an educational software company to ensure that specifying the desired adaptation can be done by non-technical authors. This is achieved by combining a simple responsive authoring-interface with underlying templates that help generate the adaptation code for GALE.

CCS Concepts

•**Information systems** → **Web applications; Web interfaces;** •**Human-centered computing** → **Interactive systems and tools;** *Interaction design; Systems and tools for interaction design;*

Keywords

authoring, adaptive hypermedia, interface design, usability

1. INTRODUCTION

Adaptive hypermedia [1, 14] nowadays comes in two flavors: *expert-driven* and *data-driven*. Everyone is experiencing automatic personalization on many web-based services (like YouTube, Facebook, Amazon, etc.). This is all realized through data-driven adaptation. In special-purpose applications, such as an on-line course text that is offered as hypermedia (through a website), an expert, skilled author or pedagogical designer needs to define adaptation rules, not purely based on content and navigation paths followed by (other) users, but based on insight as to which navigation or *learning* order between learning objects or concepts makes sense. In some cases this is not universal advice but depends on personal traits such as a *cognitive* or *learning style*.

Special purpose systems have been created for adaptive learning, Interbook [2] being one of the oldest but still best known examples. An author would create an annotated Word file: a course text with delineated sections, each associated with *outcome concepts* you learn about when studying the section and *background concepts* that are "prerequisites" for studying that section. A compiler would then turn this into an adaptive website. The possible forms of adaptation were all hardcoded in the system. Authoring was easy but the adaptation possibilities limited.

At the other extreme general purpose systems have been created, like AHA! [6] and its successor GALE [17, 16]. In AHA! and even more in GALE nothing is hardcoded. The user modeling and adaptation are defined using the GALE adaptation language GAM that allows arbitrary Java code to be used in the rules. Creating content (e. g. learning material) is completely separate from defining the adaptation and can be done using any HTML authoring tool. In line with other adaptive hypermedia research papers, when we published reference [17] we emphasized what GALE was capable of and ignored the authoring process. The (adaptive!) thesis [16] describes the adaptation authoring *language* but no easy to use tools for "generating" adaptation in this language.

In Section 2 we briefly review different authoring approaches and interfaces that were created for adaptive hypermedia, including the GRAPPLE authoring tool GAT [7] we helped to create. We highlight good ideas and problematic ones in different authoring tools. Section 3 briefly recalls the adaptation functionality of GALE to illustrate the challenge to make that functionality available to non-technical authors. Section 4 describes the new authoring tool ALAT. The aim of this description is to show how most of the adaptation power of GALE can be made usable to non-technical authors through the new authoring tool.

2. ADAPTIVE HYPERMEDIA AUTHORING

When the first adaptive course at the TU/e [3] was turned into the more generic adaptive system AHA! adding adaptation rules became difficult. An authoring tool was created that completely separated the creation of *content* from that of *conceptual structure*. This separation has become the standard way of authoring in all other authoring tools we refer to in this section (and in many more). As AHA! and later also GALE just serve "pages" like a Web server (and in fact as a Web server extension) we can allow users to use whichever is their favorite Web page creation tool.

The *Graph Author* for AHA! was first described in [5]. A snapshot of this interface is shown in Figure 1. The figure shows two panes: the left pane shows part of a hierarchy of *concepts*, also called a *domain model*. The right pane shows all "pedagogical rules" in what we call the *adaptation model*. The adaptation rules that implement the behavior are drawn from a template. The design of such templates is left to an expert. Authors just "draw arrows". When the domain and adaptation models become much larger than

HT'16 July 10 - 13, 2016, Halifax, NS, Canada

ISBN 978-1-4503-4247-6/16/07. . . $15.00

DOI: http://dx.doi.org/10.1145/2914586.2914627

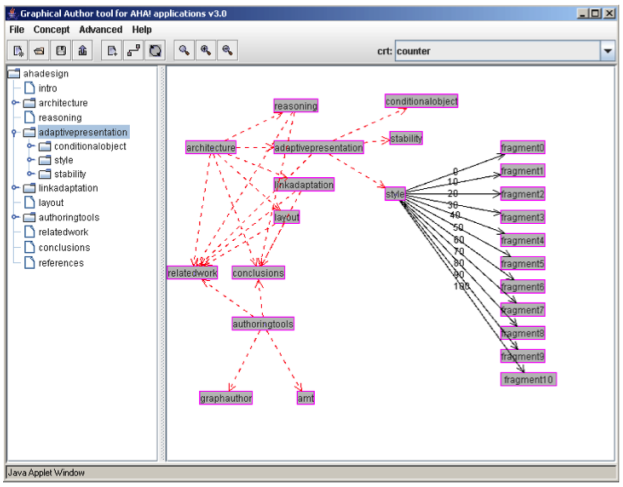

Figure 1: The Graph Author (tool for AHA!).

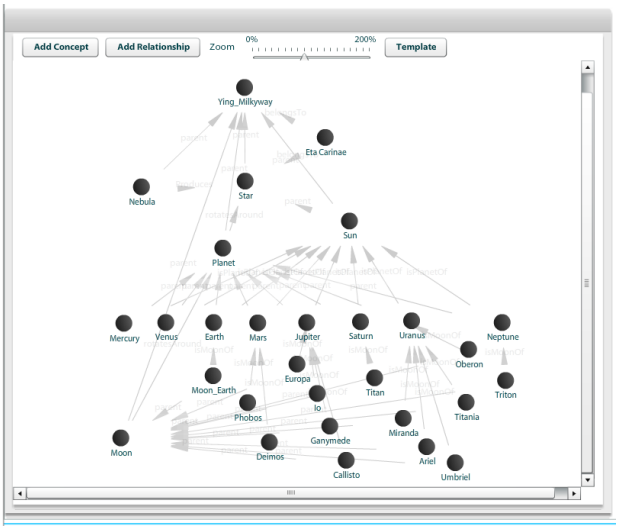

Figure 2: The Domain Tool in GAT.

in the figure the accordion menu keeps the domain model readable whereas the graph representing the adaptation model will become cluttered and unreadable.

In the European research project GRAPPLE[1] GALE [16, 17] was created as successor to AHA! and a corresponding graphical authoring tool GAT [7] was created as well. The GAT tool consists of three parts: a *domain tool*, a *pedagogical relationship tool* and a *course tool*. Figure 2 shows the graph part of the domain tool and Figure 3 shows the graph part of the course tool.

The Domain Tool[2] in GAT allows for labeled binary relations between concepts. The domain model can thus become a richly interconnected collection of concepts, much like a *domain-specific ontology*. It can be much richer than the purely hierarchical domain model shown by the GraphAuthor tool in figure 1. At the same time the graph presentation (instead of an accordion menu) quickly becomes cluttered and unreadable as the domain model grows.

For the adaptive part of an application GAT uses the Course

[1]See http://grapple.win.tue.nl/ for a description and deliverables.
[2]The Domain Tool was mainly developed by Giunti Labs.

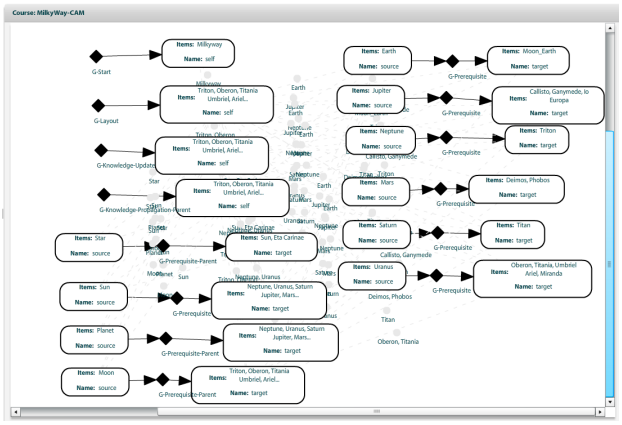

Figure 3: The Course Tool in GAT.

Tool[3] (initial design ideas described in [13]), shown in figure 3. It uses a very compact graphical representation for a collection of adaptation rules that would make for a huge graph in the GraphAuthor, but it does not make it easy to see which incoming and outgoing pedagogical relations (like prerequisites) each concept has.

During a course on adaptive hypermedia students were given a choice of using GAT to develop a GALE application or using the underlying adaptation language GAM directly. Students using GAT were generally less happy than students using GAM. Still, when in the next run of this course we asked students to only use GAM and evaluate it, writing the adaptation rules was found to be the most technically challenging part of authoring [15]. Hence the need for a new authoring tool described in this paper.

Our experience mirrors what we find about authoring tools created in other projects. The creators of ACTSim [11] made the creation of adaptive educational soft skill simulations easy by creating a limited special-purpose tool. Still, they too use a graph presentation that becomes harder to view as the models become larger.

WOTAN [9] combines an indented list hierarchy (or accordion menu) with an interface representing the current project in a directed graph. To some extent that corresponds to the domain and adaptation model representation in the Graph Author. The graph presentation uses automated layout and clustering of groups of nodes to prevent visual clutter. WOTAN also "cheats" in its indented list view: concepts can have multiple parents and therefore appear multiple times in the indented list.

We also looked into MOT 3.0 [8] that is based on the five-layer LAOS framework [4] and offers an editor that is essentially a text editor for writing adaptation rules, at least as difficult to use as the GAT *pedagogical relationship tool* for writing adaptation rule templates.

Finally we also considered AMAS [12] that focuses on ease of authoring and usability by non-experts. Subsequent designs, described by Gaffney, Conlan and Wade in [10] focused on the user experience and led to a graph-based interface that again appears to work well on small examples but we question whether it would be usable with large graphs.

The investigation into these different authoring environments has taught us that an interface for displaying and editing a *concept hierarchy* appears to work well and scales well with growing applications, and that displaying a graph of relations between concepts always leads to visual clutter with larger models.

[3]The Course Tool was mainly developed at the University of Warwick.

3. GALE ADAPTATION FUNCTIONALITY

ALAT has been designed together with an educational company (developing a platform for on-line courses mostly for grade school) and the interface design therefore has to be suitable for authors of such learning material. But ALAT is also, and perhaps even first and foremost a new authoring tool *for GALE* [16]. This means that it tries to enable authors to use as much as possible of the user modeling and adaptation functionality GALE offers, but without requiring any technical skills.

Rather than explain GALE's functionality using GALE's own adaptation language (GAM) described in [16, 17] we use the blueprint format used by ALAT. In GALE an application (or course) consists of *concepts*. Each concept has a number of (named) *properties* that have a fixed value and (named) *attributes* that have a value that is computed through rules. Typical use of these attributes (of which the names and meaning can be chosen arbitrarily) includes:

- a Boolean attribute *suitability* to check that all *prerequisites* for the concept are satisfied.
- an integer attribute *visited* to count how many times the user has visited the concept.
- a real (Double) attribute *knowledge* to keep track of the user's knowledge of the concept.

In GALE the attribute values can be updated when certain events occur (like the user visiting a concept) or can be computed from the values of other attributes (and possibly properties).

Finally, concepts can also have relationships between them. In GALE there are two predefined relationships:

- When concept A *extends* concept B it inherits all the properties and attributes (from B). We can define one "generic" concept with properties and attributes and then have all other concepts extend it.
- Through the *parent* relationships we build a *hierarchy* of concepts. GALE also offers *views* to present (parts of) the concept hierarchy as a navigation menu (for instance as an accordion menu).

In ALAT we use concept *blueprints* (templates) that define a structure common to *all concepts* of an application and that define some *special concept types* with additional structure for those. A course can contain concepts to be studied and concepts that represent tests for instance. The author selects a type for each new concept and does not need to know anything about the code that defines the behavior of concepts of that type. The selection list (in ALAT) that is offered to authors to choose a concept type is generated from the blueprint and can thus be different for different applications (or application areas) for which ALAT is used as authoring too. Below we show an (incomplete) example blueprint template. As you can see the blueprints are written using JSON syntax.

```
{
  "defaultAttributes":[
    {
      "name":"suitability",
      "type":"Boolean",
      "value":"true"
    },
    {
      "name":"knowledge",
      "type":"Double",
      "value":"0",
      "operator":"AVG"
    },
    ...
 ],
 "conceptTypes":[
    {
      "name":"text-topic",
      "default_attributes":[
        {
          "name":"info",
          "type":"String",
          "value":"This is an information concept!"
        },
        ...
      ],
      "default_rules":[
        "visited",
        "knowledge_update",
        ...
      ]
    }
  ]
}
```

The "default_rules" refer to what is defined in a different blueprint for which we show a small (incomplete) example:

```
{
  "def_att_rules":[
    {
      "name":"hasprerequisite",
      "type":"binary",
      "target":"suitability",
      "tooltip":
         "Target concept must be learned
          before source is recommended.",
      "code":"${%target%#knowledge} > 0.8",
      "operator":"and"
    }
  ],
  "persistent_att_rules":[
    {
      "name":"visited",
      "type":"unary",
      "properties":[
         {
           "name":"visited",
           "type":"Integer",
           "defval":""
         }
      ],
      "tooltip":"stores number of concept
                 visits in 'visited'",
      "code":"#[visited]:Integer event +
       'if (${#suitability}) { ${#visited}++;}'"
    }
  ],
  "def_relations":[
    {
      "name":"rotatesAround",
      "tooltip":"source concept rotates
                 around the target object."
    }
  ]
}
```

The blueprint contains three parts:

- The first part (def_att_rules) corresponds to attributes whose value is computed (whenever needed). For these attributes their *value* is not stored permanently but the *code* to compute that value is stored. The concept's attribute for which the value is computed is "suitability" (called the "target" attribute). The code fragment computes the suitability by check-

Figure 4: Hierarchical presentation in ALAT.

ing whether the knowledge (attribute value) of the target concept is greater than 0.8. So if concept A has concept B as a prerequisite then the suitability of A depends on the knowledge of B being greater than 0.8. If A has several prerequisites these pieces of code are combined using the logical "and" operator to form the complete code for A's suitability.

- The second part (persistent_att_rules) defines how an event triggers an update to some attribute value. That value is stored permanently. The example defines a rule called "visited" which defines the updates for an attribute that is also called "visited". The code fragment increments the visited attribute when the concept is visited while being suitable. The event code for an attribute is actually a piece of Java code. Several rules can contribute to the code, and each piece of code is added to the event code. Hence the "event +" part which indicates that the code is concatenated (as a string) to the already existing event code.

- The third part (def_relations) just defines relations that have no associated behavior. They can be used in code fragments for attributes or other relations and for generating adaptation in the (page) presentation. The "rotatesAround" relation between *planet* and *star* can be used in generating part of the page describing what a star or a planet is.

When using ALAT the "adaptation model" is assembled from the small code fragments shown in the blueprint. This "assembly" does not provide the complete adaptation power of GALE but offers the same power that GAT did. In all the examples of GALE applications we have seen so far, some of which are mentioned in [15], the code that was hand-written in GAM could all have been produced by ALAT in a relatively straightforward way.

4. ALAT: THE NEW GALE AUTHORING TOOL

ALAT is the result of many brainstorming sessions, followed by mockup design by a professional designer, followed by many iterations of coding, testing and refining. As the look and feel of the final tool still stays quite close to the earlier mockup design we illustrate ALAT using screen shots from the actual tool only. The tool not only completely hides the technical details you saw in section 3 but even does not require any knowledge of these details. The blueprints contain *tooltips* that are shown in ALAT and that should be written in a language the non-technical author understands.

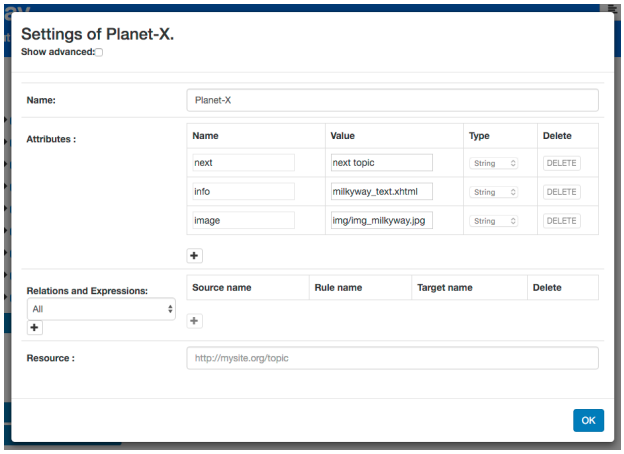

Figure 5: Concept Details in ALAT.

4.1 Creating and Navigating the Concept Hierarchy

Because of what we learned in section 2 ALAT does not use a graph-like domain or adaptation model presentation but shows a hierarchical domain structure through an accordion menu. There is a "flat" and a "hierarchical" view. We only show the hierarchical one, in figure 4. At each level that is shown there is a blue bar with + button to add a concept at the corresponding level, an X button to delete (a concept or complete sub-hierarchy) and three dots to access concept details.

From De Roode Kikker we learned that in school a teacher may wish to offer only part of a course (purchased from a publisher) to his students. For this purpose ALAT places a checkbox in front of each concept. When the teacher unchecks some boxes these concepts (and the whole subtree below them, if applicable) will automatically not appear in the course and all associated adaptation rules will not be included in the generated adaptation model.

4.2 Editing Concept Details

When an author adds a concept a small dialog box appears to enter the concept name and type. The "conceptTypes" from the blueprint determines the possible choices for the type. The concept details dialog that follows is shown in figure 5 in which a newly Milkyway concept named "Planet-X" is shown. Comparing this dialog box to the first blueprint from section 3 it is clear that most information is hidden. The attributes and rules that are common to *all* concepts are hidden. Only the attributes that are added for the chosen concept type are shown. In the Milkyway example from [17] these are "next", "info" and "image".

In figure 6 we show the concept details for the concept "Earth" in which the author has added attributes and relationships. The dialog box contains four different parts:

- Name: Each concept must have a name that is unique within the course. That name is given when the concept is created but it can be changed later through this dialog box (as long as it stays unique).

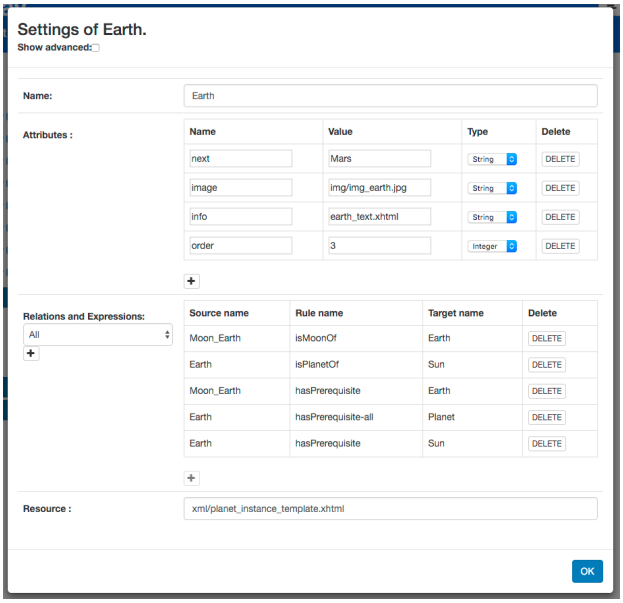

Figure 6: Concept Details for "Earth" in ALAT.

- Attributes: The attributes that are common to all concepts are *not* shown. The attributes "next", "info" and "image" follow from the Milkyway concept type and the attribute "order" was added by the author through this interface. The hidden standard attributes can be made visible by ticking the "show advanced" checkbox. Unlike the additional attributes the standard ones cannot be deleted as doing so would break standard adaptation rules associated with them.

- Relations and Expressions: again, the standard ones coming from the blueprint are not shown unless the "show advanced" checkbox is ticked. They would include "visited" and "knowledge update" for instance. In adaptive courses a concept may have some *prerequisites*, and in our example the concept Earth has two prerequisites: Planet and Sun. We not only show some prerequisites but also added some more relationships (isMoonOf and isPlanetOf). Binary relations have a *source* and *target* concept and are shown in the Concept Details window for *both* concepts. We see for instance that Moon_Earth has Earth as a prerequisite and that Moon_Earth has an isMoonOf relationship with Earth. When adding a relationship it is selected from the list (that shows All in the figure). This list is determined by the second blueprint we showed in section 3 and can be extended with new relationships (that then cannot have associated code). Relationships are always created for the source concept. The dialog box shown in figure 7 is used to select target concepts. You can select multiple concepts and ALAT will create a relation for each of them.

- Resource: this final element is the name of the file to be used for the page. The resources can come from any website and need not reside on the server on which the adaptation model is stored.

The authoring process for the domain and adaptation model of an application/course ends with a simple press of the "generate" button. ALAT generates a file called "concepts.gam" in the main directory for the course.

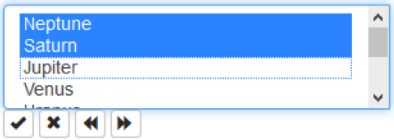

Figure 7: Dialog box to select concepts for a relation in ALAT.

5. DISCUSSION AND FUTURE WORK

Designing authoring interfaces for adaptive (educational) hypermedia has been a process of trial and error throughout the past two decades, not only for the TU/e research group but as section 2 has shown also for other researchers. It has also not been widely published about. Publications concentrate on the end-user experience, not on the process of designing or creating the applications or frameworks.

In the design of ALAT we have worked closely with an educational software company. This has confirmed that it is essential to make an authoring interface very simple. So we opted for showing a concept hierarchy, hiding all adaptation-related details from the authors (through blueprints), yet allowing experts to add features (attributes, relations, adaptation rules) if desired. The interface is guided by the blueprints in deciding what to show in selection boxes and which attributes and relations to provide. The ability to "deselect" parts of an application was also an explicit wish from teachers using courses that have been prepared (sold) by a publisher.

While we are confident that ALAT is much easier to use than any authoring interface we created in the past we still need to run extensive evaluations. An authoring experiment is planned in the second quarter of 2016, not in time for this paper but for a future publication.

We also plan on adding an interface for a graph visualization tool so that the graph of relationships between concepts can be shown. But based on our previous experience we intend to use that graphical interface as "read-only", not as an editor.

The attentive reader may have noticed that in this paper we have only considered adaptation based on what the individual user has done. Group adaptation, or collaborative filtering, has not been considered. This is not a limitation of the authoring interface ALAT per se but is a restriction built into GALE. Although GALE can (and does) include the user's identity in its data model, as what is called an "entity" and can handle groups as entities and can have adaptation rules that update the "user model" for such a group entity GALE currently has a deliberate limitation that an entity cannot access the user model of another entity. This guarantees that every user model is kept private. Only when GALE gains a controlled way to access group entities can we consider making group adaptation available in the authoring process as well. We are looking forward to our experiments with different author- and user-groups to find out which desires for this and possible other future extensions pop up.

Readers who like to experience ALAT first hand and experiment can visit http://gale.win.tue.nl/ALAT and register. The tool allows you to create new adaptive applications, while keeping the (pages) on your own server.

6. ACKNOWLEDGMENTS

This research is partly funded by the City of Eindhoven in the "Adaptive Learning" project. Additional funding came from the Eindhoven University of Technology and from De Roode Kikker.

7. REFERENCES

[1] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–110, 2001.

[2] P. Brusilovsky, J. Eklund, and E. Schwarz. Web-based education for all: a tool for development adaptive courseware. *Computer Networks and ISDN Systems*, 30(1-7):291 – 300, 1998. Proceedings of the Seventh International World Wide Web Conference.

[3] L. Calvi and P. De Bra. Proficiency-adapted information browsing and filtering in hypermedia educational systems. *User Modeling and User-Adapted Interaction*, 7(4):257–277, 1997.

[4] A. Cristea and A. De Mooij. Laos: Layered www ahs authoring model and their corresponding algebraic operators. In *The Twelfth International World Wide Web Conference, Alternate Track on Education*, 2003.

[5] P. De Bra, A. Aerts, B. Berden, B. de Lange, B. Rousseau, T. Santic, D. Smits, and N. Stash. Aha! the adaptive hypermedia architecture. In *Proceedings of the fourteenth ACM conference on Hypertext*, pages 81–84. ACM, 2003.

[6] P. De Bra, D. Smits, and N. Stash. The design of aha! In *Proceedings of the seventeenth ACM conference on Hypertext*, page 133. ACM, 2006, adaptive version at http://aha.win.tue.nl/ahadesign/.

[7] P. De Bra, D. Smits, K. van der Sluijs, A. Cristea, and M. Hendrix. Grapple: Personalization and adaptation in learning management systems. In *Proceedings of the ED-MEDIA World Conference on Educational Multimedia and Hypermedia*, pages 3029–3038. AACE, 2010.

[8] J. Foss and A. Cristea. The next generation authoring adaptive hypermedia: Using and evaluating the mot3. 0 and peal tools. In *Proceedings of the twentyfirst ACM conference on Hypertext*, pages 83–92. ACM, 2010.

[9] M. Freire and P. Rodriguez. Comparing graphs and trees for adaptive hypermedia authoring. In *Proceedings Third International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia (A3EH; in conjunction with AIED)*, pages 4–12, 2005.

[10] C. Gaffney, O. Conlan, and V. Wade. The amas authoring tool 2.0: A ux evaluation. In *Proceedings of the twentyfifth ACM Conference on Hypertext and Social Media*, pages 224–230. ACM, 2014.

[11] C. Gaffney, D. Dagger, and V. Wade. Authoring and delivering personalised simulations-an innovative approach to adaptive elearning for soft skills. *Journal of Universal Computer Science*, 16(19):2780–2800, 2010.

[12] C. Hampson, O. Conlan, and V. Wade. Challenges in locating content and services for adaptive elearning courses. In *Eleventh IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 157–159. IEEE, 2011.

[13] 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 *Third European Conference on Technology Enhanced Learning (EC-TEL)*, pages 132–143. Springer LNCS 5192, 2008.

[14] E. Knutov, P. De Bra, and M. Pechenizkiy. Ah 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. *New Review of Hypermedia and Multimedia*, 15(1):5–38, 2009.

[15] V. Ramos, P. De Bra, and d. Smits. Gale extensibility evaluation : a qualitative approach. In *World Conference on E-Learning in Corporate, Government, Healthcare and Hither Education (E-Learn)*, pages 296–305, 2013.

[16] D. Smits. *Towards a Generic Distributed Adaptive Hypermedia Environment*. PhD thesis, Eindhoven University of Technology, adaptive version on http://gale.win.tue.nl/thesis/, ISBN 978-90-386-3115-8, 2012.

[17] D. Smits and P. De Bra. Gale: a highly extensible adaptive hypermedia engine. In *Proceedings of the twentysecond ACM conference on Hypertext*, pages 63–72. ACM, 2011.