BALANCING ADAPTIVE CONTENT WITH AGENTS
Modeling and Reproducing Group Behavior as Computational System

Harri Ketamo
Satakunta University of Applied Sciences, Pori, Finland

Keywords: Adaptive systems, User modelling, Software agents, Artificial intelligence, Social behaviour.

Abstract: To ensure the quality of adaptive contents, there should be continuous testing during the development phase. One of the most important reasons to empirically test the content during the development phase is the balance of the adaptive framework. Empirical testing is time-consuming and in many cases several iterative cycles are needed. In 2007 we started to develop methods of testing in a computational test bench. The idea to speed up the production process was based on software agents that could behave like real user community. The study shows that we can construct very reliable artificial behaviour when comparing it to human behaviour in group level. On design phase's usability tests, we are especially interested in group behaviour, not on single action etc., which means that the method suits for it's purposes.

1 INTRODUCTION

Adaptation can be seen as being high end personalization: In adaptation, the system optimizes the output with technologies that, in general, can be divided into two main groups: static (indirect) adaptation and dynamic (direct) adaptation. In static adaptation the rules are fixed beforehand by developers. In dynamic adaptation the system tracks the user and optimizes the navigation paths according to the user's behaviour. Dynamic adaptation of system requires at the least a user model, a context model and artificial intelligence. (Kinshuk, Patel & Russel 2002; Brusilovsky, 2001; Manslow 2002).

Because the idea of adaptive educational systems is to produce individual and optimized learning experiences the high end user models as well as methods are relatively complex (e.g. Raye, 2004; Lucas, 2005). Furthermore, the social dimension should not be forgotten: In very large samples, the most successful outcomes in group level may contain valuable guidelines for adaptation.

Before we can ensure the quality of the adaptation, at the least we have to know if 1) the adaptive framework is in tune, 2) is there out layered content and 3) are the learning paths really unique?

It is possible to theoretically check the first three questions by constructing an algorithm that computes all the possible combinations, but there will be no such system that can compute the problem in a reasonable amount of time within large domains.

Another solution to this is to model human behaviour as a system and then run the learning material using these artificial users. In our study the behaviour of software agents are constructed according to social behaviour: We have constructed several archetypes of a user by clustering the behaviour of human users.

2 RESEARCH TASK AND PROCEDURE

2.1 Research Task

In this study, the adaptation is studied in terms of Complex Adaptive Systems: self-organization, entropy and emergence. These concepts describe and define the high level system properties. In a computational system that aims at simulating group behaviour, focusing on high level phenomena might incur the best outcome.

Earlier results, received from preliminary studies of the application (e.g. Ketamo, 2008) are referenced in this study. The computational core is based on the author's previous work that had been applied, for example, as the background of an educational game

Ketamo H.
BALANCING ADAPTIVE CONTENT WITH AGENTS - Modeling and Reproducing Group Behavior as Computational System.
DOI: 10.5220/0002781502910296
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

series (e.g. Ketamo & Suominen 2008; Ketamo 2009).

2.2 Material in Testing

Mathematics Navigator is a product family distributed by Otava Publishing Company Ltd.. Mathematics Navigator is based on an adaptive content management system (Figure 1) and content objects. Mathematics Navigator is a high-end platform designed to produce dynamically adaptive and self-organizing content. Dynamic adaptation is done by varying the sequences of course elements (bits of theory, examples, exercises) and supplying individual learning paths for the students. In this study Mathematics Navigator is used as a test and evaluation platform.

Adaptation of content is done according to a user's / student's learning and studying behaviour. This kind of modelling requires at the least 1) a user model that records the skills and learning of each user and 2) a domain or context model on the relations between the bits of information of learning content. Mathematics Navigator gathers information on a student's actions and, based on this information, creates and modifies an understanding of his/her mathematical competence. Mathematics Navigator adjusts the exercises and content to support the development of students. On the user interface (Figure 2), a student has a graphical representation of the development of his/her learning profile.

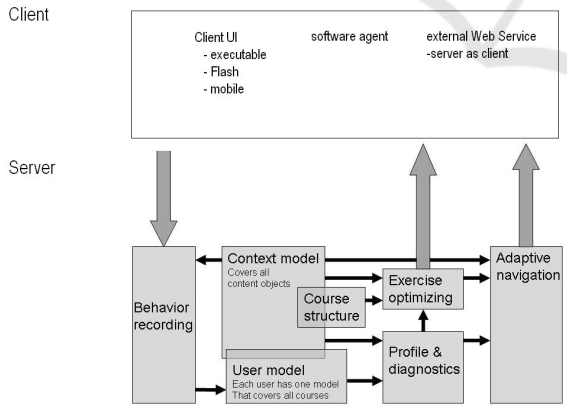

Figure 1: The high level architecture of Navigator.

The user interface of Mathematics Navigator is based on a menu-bar and three main areas (Figure 3). On the left side of the interface is a table of contents and a content-related competence profile of the user. The table of contents presents two different views of the content: 1) a traditional book-like table of contents and 2) an exercise-adapted table of contents.

The exercises are presented one at a time in the bottom-right corner of the interface. The user can't proceed to a new exercise before the current one has been answered by picking an answer from a total of 4 alternative answers. The exercises are selected to support an individual user's learning needs. There are no fixed paths for learning: everything is based on the student's competence profile and estimated need for practice and content. Guiding factors in exercise selection are: 1) course structure (a traditional table of contents), 2) the measured and estimated learning abilities and areas of weaknesses, and 3) critical points, derived from the learning community's actions.

The competence profile values are indicated by colours that vary from red (insufficient skills or skills not yet estimated) via yellow to green (good skills). Those skills mastered and measured within a certain theme will be transferred with certain estimates to other themes requiring similar (by proximity or by hierarchy) skills.

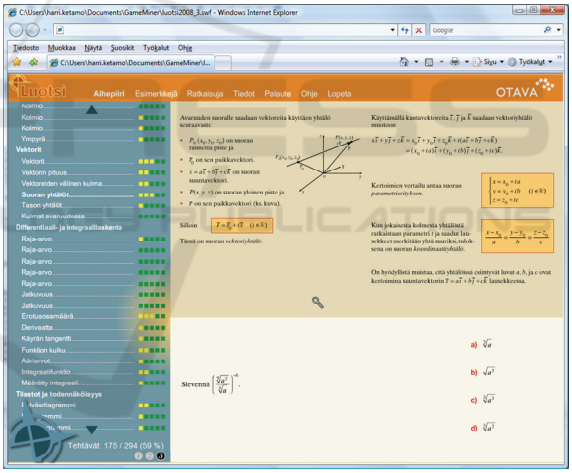

Figure 2: The user interface of Mathematics Navigator and a basic mathematics course (in Finnish).

The entire content - theory, examples and exercises - is based on the idea of content objects. A content object is a unit containing the smallest possible bit of content that can be used independently without the support of other content objects. Naturally, products are based on content objects that strongly support one another. The content objects are described by 1) detailed rank ordered keywords (tags) that define the structure of content and 2) single relations between objects in a preferred order. The preferred order is calculated as a state of semantic network, formed according to tags. The method can be described in

general in terms of weighted proximity between objects (Figure 3).

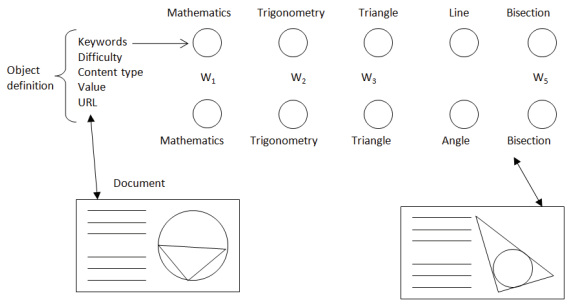

Figure 3: Defining proximity between the content objects.

The calculated proximities between the content objects can be visualized as weighted graph (Figure 4) which basically visualizes the whole idea behind the content model.

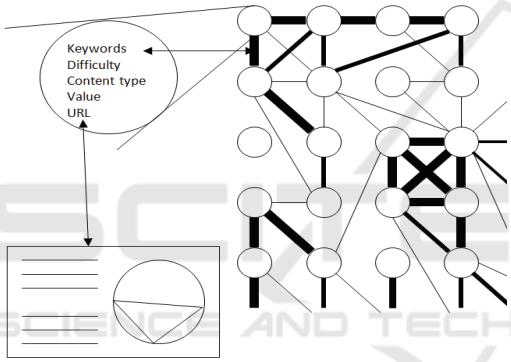

Figure 4: Calculated proximities between content objects visualized as weighted graph.

This kind structure and content model offers numerous possible paths through the material. A special feature of the modelling is the capability of self-organization: The clusters inside the material were self-organized according to mathematical definitions and context. This self-organization is a key feature of Mathematics Navigator and it remarkably reduces the costs of producing adaptive content. In this study the aim is to reduce the costs of empirical testing, but we can find similarities in information modelling between Mathematics Navigator and the test method.

In Mathematics Navigator, a learner's performance can be estimated in relatively complex domains with very detailed and extensible user models, combined with a learnable system. Of course, this kind of modelling takes some time: the system must learn the performance level of the user. Empirically formed decision trees were used when the system had not yet constructed a detailed enough profile about the learner in order to dynamically adapt the content and exercises to his/her learning needs.

Empirical testing of such systems takes a lot of time and in many cases iterative cycles are needed that incur new time costs on development.

2.3 Procedure

This study is a design study with actions like the design of the system, implementation of the system, empirical testing and evaluation. The design and implementation of the system was done in the autumn 2007-winter 2008 and the design was revised during the summer 2008 as a consequence of pre-tests.

Empirical testing (n=447) was done in two phases: the data from human users was collected during the content development projects of Mathematics Navigator in 2005-2008. The data collected during those studies was meant to support the development of those specific projects. However, the data was found to be complete enough to serve as empirical data for this study. This existing data makes it possible to run tests while the work was in progress.

The empirical testing reported in this study is based on the autumn 2008 -version of the system. Tests were run for up to 1000 artificial workers, but in most cases the system found its balance using less than 100 artificial workers. The limit of the test sample size is defined by an estimator meant to measure the possibility of changing the visualization structures (Figures 6-8). In fact, the testing costs would not be higher even if we run the tests using millions of artificial workers. Naturally, the need for computing time would have increased.

The system is implemented with software agents using the same application interface as the host system, which means that they can act as human users. The host system that they are logged in does not recognize which connections are from artificial users and which are from human users. This method requires that the user interface should be described in detail. This description focuses only on interactions, not on visualisation. Because Mathematic Navigator is a Web Service, the user interface was defined with SOAP and WSDL and no further definition for this study was needed.

3 RESULTS

In this study, artificial users represent archetypes of human users. Artificial users are based on complex modelling which means, that there are several archetypes of users as well as variance inside the archetypes. The common denominator for all is that everything is based on human user- and group behaviour.

In the first step, data collected from human users is used to construct a model about behaviour in Mathematics Navigator. This model is formed from a group behaviour point of view. The key elements in the model were patterns of UI actions and particularly the probabilities and uncertainties of following action according to observed patterns. In the model the averages and variances were not the key point. In fact the modelling is not statistical at all: The model aims to predict behaviour, not to test any hypothesis. The patterns of behaviour were clustered in order to build the archetypes of users, which increases the similarity of artificial users to human users.

One interesting find during the pre-tests was that, when increasing uncertainty in decision-making (increasing variance in decision in statistical terms) up to specified limits, the system could point out possible problems in a shorter time. This finding is relatively straightforward: Because we are not looking for statistical evidence, we can focus on pointing out problems without proving them statistically. This helps us in two ways. Firstly, the computational requirements were decreased. Secondly, the estimator, calculated to detect when the simulation is ready, could be used in a more reliable way. The challenges of avoiding behaviour that never happens with human users are great. On the other hand, we never can be sure how users behave in digital environments. At this point, we have to accept the fact that we might construct agents that fail when compared to human users' behaviour.

As a result of modelling, a finite state system with a structure equation about action probabilities, uncertainty and disorder was formed. In Figure 5 one part of the model behind the decision-making system (probability network) of the artificial user is visualized. User and context modelling was solved technically by constructing a dynamically extensible Semantic Networks. The user model could be exported e.g. as an XML Topic Map and could be manipulated by xPath.

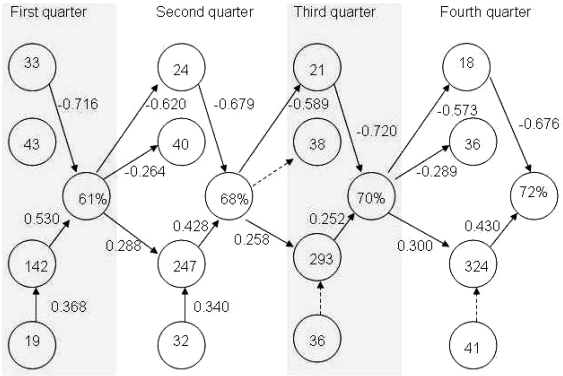

Figure 5: Adaptation schema as a system model.

The three key elements of Complex Adaptive Systems are self-organization, entropy and emergence. All of these can be found in the visualized learning paths of Navigator's users (Figure 6). Similar results could be achieved when the artificial user's behaviour is visualized (Figure 7). In the figures the paths are formed by connecting content objects browsed during use of Mathematics Navigator. In the figures the nodes are essential pieces of content that can be defined as being 'the backbone of the domain'. The connectors show the paths that users had passed through. When reading the figures, the assumption is that there were numerous connectors between nodes, but some directions or patterns are more frequent than others.

In terms of Complex Adaptive Systems, this systematic organization of connectors pointed out from disordered can be called emergence. In Figures 6 and 7 only connectors that are stronger than average are presented. Learning paths and progression can be read from left to right. In Figures 6-8, only the most significant paths are visualized. If all paths were to be visualized, the figures would not be readable at this scale.

The most important finding is that the visualizations remain the same when comparing a human user's visualization and visualizations constructed according to the behaviour of artificial users.

Naturally there are differences: artificial user's visualization (Figure 7) is rougher than a human user's visualization (Figure 6), because it is based on archetypes.

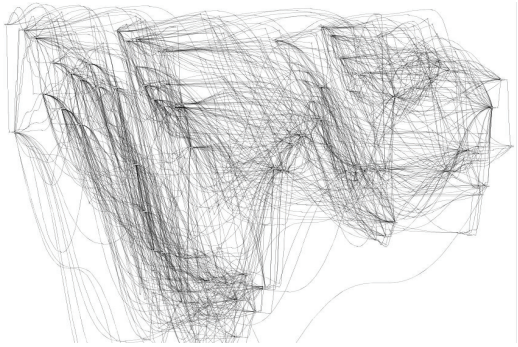

Figure 6: Emergence formed by a human user's learning paths.

Figure 7: Emergence formed by artificial user's learning paths – current version (partial visualization in order to show similarities to visualization in Figure 6).

However, during the development of the algorithm, the roughness of the group behaviour of artificial users has decreased (compare to figure 8). When in pre-test version of artificial users (figure 8) there were relatively few strong navigation paths, in current version (figure 7) the visualization of behaviour remains more versatile.

The visualizations indicate that, at a general level, human users and artificial users cause similar emergences and as a group they behave in a similar way. However, this similar behaviour cannot be seen at an individual level. We have to focus on group behaviour at a very general level.

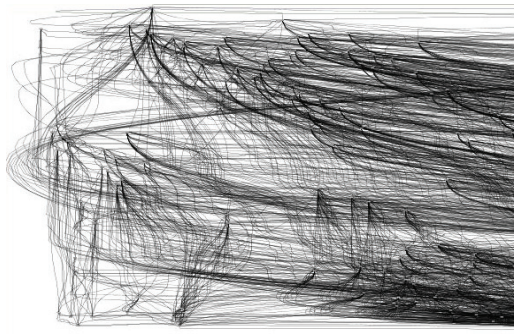

Figure 8: Emergence formed by artificial user's learning paths – pre-test version (partial visualization).

From the product development point of view, the question is about how modelling and visualizations can be used is to check if the framework is in tune. One common problem caused by the complexity of a system is visualized in Figure 9: The definitions of the content objects construct a framework that can point out a 'local minimum' on competence for an unexpectedly long time. In visualization this is seen as a connector back to the object itself.

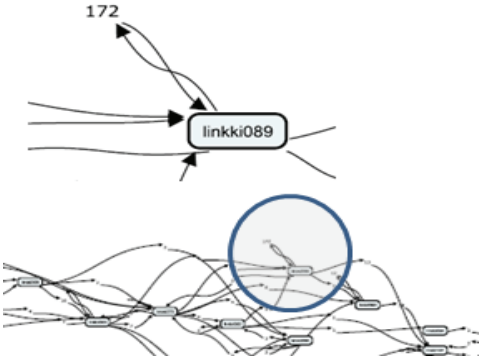

Figure 9: Problems with content: the framework is not in tune.

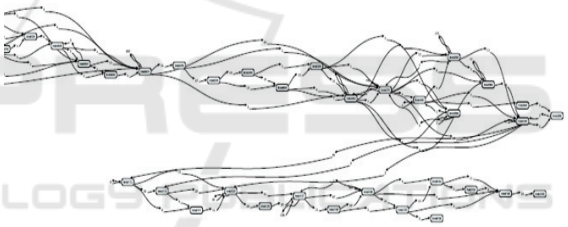

Figure 10: Unexpected, but correct re-organization of the content.

Another example that could sound like a problem is visualized in figure 10, where we can see a strong parallel navigation patterns. At first it seems an unexpected behaviour of the system. In fact, in the visualization we can see a refresh path, organized for those of the artificial users whose mathematical skills were not developed enough during the course. The refresh path was launched by 2-5 last exercises in the course.

Artificial users cannot correct such problems but authors of the content can check and improve the definitions of content objects. In most cases such an event is fixed by adding relevant keywords into the material.

4 CONCLUSIONS

The aim of the study was to construct a method of reducing time and workload costs when developing new courses in the Mathematics Navigator platform. The main questions to be studied computationally were the following: 1) Is the adaptive framework in tune? 2) Is there out layered content? and 3) Are the learning paths really unique?

One of the efficient solutions was artificial users that represents archetypes of human users. The artificial users was implemented as software agents using the same application interface as the host system, which means that they can act as human users. The host system that they are logged in does not recognize which connections are from artificial users and which are from human users.

Questions 2 and 3 could be studied in detail: Learning paths are relatively unique but not randomized. There is clear emergence as well as remarkable entropy in paths formed by human users and artificial users. Also, out layered content can be noticed easily. However, Question 1 (Is the framework in tune?) was challenging: Being in tune is not only a computational task, it also deals with curriculum. In the current system, the curriculum was not in focus and therefore we could not say that the framework is in tune according to the curriculum. At best we can say that the framework is very probably in tune in a solution specific mathematical context.

In general, the studies show that we can construct similar group behaviour between human and artificial users and the saved resources in development can be significant. In terms of resources, the development project's savings are estimated to be as high as 2-4 months of a developers work and 2-3 months of total time in project schedule.

The next phase of the study is to apply the method to other relevant contexts, for example in game development or in applications of social media. The game development applications could use relatively similar mechanics described in this study. However, social media has brought an increasing need for intelligent agents, that could search and pick the most important pieces of content out of the enormous information overload.

Teachable software agents could be used as personal media readers that could pick up those personally meaningful messages. Furthermore, behaviour on reading messages is relatively close to navigation behaviour in general level: In both cases there are individual behaviour patterns and what is different between individual behaviour and normal/average behaviour explains something from the goals of the behaviour.

REFERENCES

Brusilovsky, P. (2001). Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*. Vol 11, pp. 87-110.

Ketamo, H. (2009). Semantic networks -based teachable agents in an educational game. WSEAS Transactions on Computers, vol 8(4), pp. 641-650.

Ketamo, H. (2008). Cost Effective Testing with Artificial Labour. In proceedings of 2008 Networked & Electronic Media Summit. Saint-Malo, France, 13-15.10.2008, pp.185-190.

Ketamo, H. & Suominen, M. (2008). Learning-by-Teaching in Educational Games. In proceedings of Ed-Media 2008. 30.6.–4.7.2008, Vienna, Austria., pp. 2954-2963.

Kinshuk, Patel, A. & Russell, D. (2002). Intelligent and Adaptive Systems. In H.H. Adelsberger, B. Collins & J.M. Pawlowski (ed.). Handbook on Information Technologies for Education and Training. Germany: Springler-Verlag, pp. 79-92.

Lucas, P.J.F. (2005). Bayesian network modelling through qualitative patterns. *Artificial Intelligence,* Vol 163 (2), pp. 233-263

Manslow, J. (2002). Learning and Adaptation. In Rabin, S. (ed.) AI Game Programming Wisdom. Massachusetts: Charles River Media, Inc., pp. 557-566.

Reye, J. (2004). Student Modelling based on Belief Networks. International Journal of Artificial Intelligence in Education, Vol 14, pp. 63-96.