

# INTERACTIVE TUTORING MODEL USING INFORMATION CYCLING ON THE WWW

J. Smid\*

## Abstract

The goal of this article is to describe a user system framework that provides tutoring and testing. Because this tutoring system helps to introduce definitions and simple relations among concepts, it is suitable for classroom use. For every individual, the mode and amount of information delivered is personalized by the system order to achieve optimal information delivery, each user is characterized by a set of parameters. These parameters are used to minimize the time of the learning process. We consider only the parameters that we believe are the most significant, namely, the number of presentations that allow the user to pass a test without the need for future cycling of the same unit. The system was developed using the PHP and XML programming environment. We present preliminary classroom results along with statistical simulations of the system. Typical application domains were simple procedures from discrete mathematics, probability, computer science terminology, statistics, and elementary vocabulary. The presented implementation via the XML data structure is intended for the future WWW where servers can exchange tutorial materials developed by different authors.

## Key Words

Interactive tutorials, WWW-based implementation, user characterization

## 1. Introduction

In this article we will describe an information-cycling system. The task of the intelligent tutoring system (ITS) can be briefly characterized as sequencing, diagnosis, and responding to user requests for instructional material. In the ideal case, the material is constructed on demand, and intelligent remediation can be performed at any level. The ITS system's functionality has two aspects: material sequencing and mistake diagnosis. A logical mistake should be corrected by providing the appropriate information. The strategy of presentation may depend on the user. The extent to which the content and the presentation strategy should be controlled by the user profile is one of the crucial questions of ITS. Weber [1] reports moderate to no effect of a single adaptation technique.

\* Computer Science Department, Morgan State University, 1700 East Cold Spring Lane, Baltimore, MD 21251, USA; e-mail: jsmid@jewel.morgan.edu  
(paper no. 202-1329)

In our Information Cycling and Diagnosing (ICD) system, each unit is presented by cycling items of a unit in the form of definitions, statements, and quizzes. A final test allows the user to leave the unit and to pass to the next one. A user path through the system can be characterized by a set of parameters. We have limited our attention to only two adaptive parameters for each information unit. We believe that the most important parameters control the number of unit cycles and the unit exit criterion. This means that each user is associated with a recommended number of cycles for each unit, and this parameter can be adjusted in order to minimize the length of the teaching process. The other meaningful parameter that we have not used is the amount of information presented at a time. A further issue that was not investigated is the parameters that describe the scoring information to the user. These issues have important psychological value and should be analysed separately and rigorously.

The knowledge domains we have investigated were algorithms for solving linear algebraic equation using the Gauss-Jordan method (GJ), theoretical computer science statements such as the Euler circuit theorem (Konigsberg bridges theorem), and elementary French vocabulary. Our goal is to compare tutoring mini-systems that can be effective in the introductory university-level classroom. We are interested in using interactive educational tools to introduce key vocabulary and concepts. From our classroom experience, we have seen that students can benefit substantially from interactive information cycling. When students are exposed to vocabulary training before an instructor-led lesson, difficult concepts can be introduced more easily. This idea has been applied in classrooms in which the majority of the students are required to take mathematics or computer science but are not required to understand deeper principles such as proofs. This system is particularly useful for students who need to gain an understanding of the basic vocabulary and concepts. Introducing the different number of cycles (parameters) or repetitions of items of a given unit is motivated by the fact that even within the same classroom, there are differences in presentation needs. The differences among users in such a classroom setup reflect skill level rather than background level. In accommodating a variety of presentation needs, the user system reflects skill level in both reducing redundancy for more skilled users and

maintaining an appropriate level for those less skilled. The targeted users for our ICD experiments were undergraduate students of mathematics and computer science classes at the Morgan State University. Five classes, ranging from 10 to 20 students, participated in the experimentation.

## 2. Flexible and User-Adaptive Systems

We call systems flexible when they allow a user to proceed to the following unit as soon as a test question(s) is (are) correctly answered. Such systems are simpler than user-adaptive systems and can be implemented using a simple environment such as JavaScript. Alternatively, we define user-adaptive systems as those that utilize more than merely pass-fail data. The defining characteristic of a user-adaptive system is its (optimal) utilization of the user profile data. Each individual's profile is established by collecting data intrinsic to the user, such as the number of cycles necessary to memorize a set of information. The profile data are acquired during a user session in which the individual is classified in a category based on the performance data. The category is established on a pre-test basis or simply initialized to one pass per each unit. Because flexible systems use only pass-fail criteria, the literature usually refers to only user-adaptive systems as being adaptive.

## 3. Evaluation of User Path

A number of successful systems for learning and teaching have been developed (e.g., [2, 3]). The task of measuring complexity of adaptive and non-adaptive systems was addressed at the *Workshop on Empirical Evaluation of Adaptive Systems* [3, 4]. The parameters describe, for example, how much information and how many times it should be provided. Although some systems use adaptive parameters, they are not clearly evaluated in the literature against non-user-based systems. Regardless of the nature of adaptive parameters, the goal of a user-adaptive system is to lower the complexity of the individual path. However, the complexity of the user behaviour as a path between information gathering and knowledge testing is difficult to evaluate. Therefore, in this work we restrict our study to a simplified description where the cycle path is a record of number of cycles per each unit. The goal is to analyse and evaluate the difference between the number of operations of the user in adaptive systems versus flexible systems. In some instances, such as GJ elimination, the number of operations estimated based on the user profile can be significantly higher and closer to the user's needs. Consequently, the user-adaptive system is more efficient because the user passes only one exit test, whereas the non-adaptive system user must remediate through entire units including the exit test. The complexity of implementing a user-adaptive system is much greater than that of implementing flexible

systems and must be always considered in the overall analysis.

## 4. User Path Optimization

Flexible interactive systems continually provide a sequence of information items until test questions are successfully answered. In the ICD system, the user profile specifies the number and type of cycles that are needed to answer the test question correctly. Depending on the nature of the mistake, the user either repeats an information cycle at the same unit or is returned to some of the parent (prerequisite) units. One example of a flexible system is the GJ elimination training tool experiment presented in [5]. In this experiment, the user is offered GJ elimination steps as a quiz until a criterion is satisfied. No user modelling is required because the number of cycles is added one by one as necessary. However, if a learning goal contains the GJ procedure as one of its components then a user model may be required. More extensive training may be needed for the units that have the GJ as a prerequisite. In an average situation, one unit has several prerequisite units. For example, a typical graph of a recommended path is not a mere series of units, but a path through a directed acyclic graph that was studied in [6]. Therefore, in user systems the test questions, or more precisely, the user's wrong answers should also point to a particular prerequisite knowledge. Thus, the exit test involves a task that requires all prerequisites. For example, the ability to correctly apply the Euler theorem (Konigsberg bridges theorem) can be used as an exit test. Tests for prerequisites involve definitions of graph, connected graph, cycle, Euler cycle, and degree. For a typical situation, see the Euler theorem diagram and normal equations diagram in Figs. 1(a) and (b). This approach can be applied for different domains. The similarity of the tutoring of domains such as Konigsberg bridges and French grammar is based on the representation of information in the form of a tree where each vertex represents a statement, definition, example, sentence, or sentence component. We will consider simplified situations that are still pedagogically important. In the simplest scenario, the micro-strategy within a unit cycles item of the unit (definition, quizzes) until a criterion is satisfied, assuming that each definition and quiz have been seen once successfully. The macro-strategy is to follow the recommended sequence of units, which in some instances, as in the above-mentioned case of the user returning to the parent unit, may be repeated. Each cycle (pass) through the unit is recorded. The numbers of cycles generate a sequence that we call the cycle path  $c_1, c_2, c_3, \dots, c_n$ . The cycle path is not the actual path through the unit items. We ignore this potentially useful information in our simplest model because we believe the number of cycles itself is the most relevant information. This belief comes from personal teaching experience: we do not regularly use this type of information for teaching. A set of definitions, statements, and quizzes is used in each unit. The number of included components is higher than the number of actually used components for each user. This is to assure

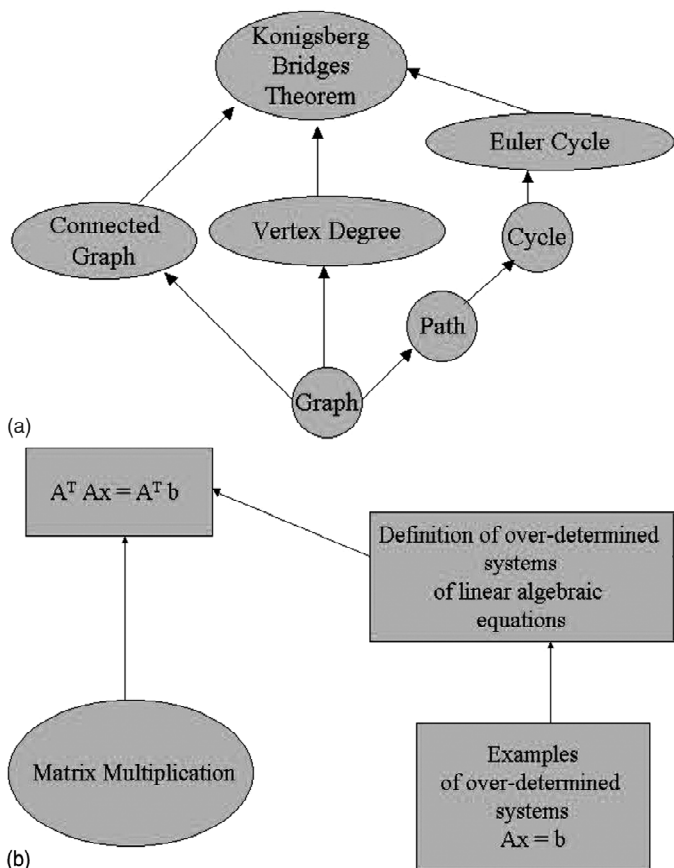


Figure 1. (a) The Euler theorem domain. (b) The normal equations domain.

sufficient variability of presentations. In the following we describe two cases for user-adaptive and flexible systems.

*Case 1.* First, we will assume that units are independent. Each unit is self-contained and consists of several definitions, statements, and tests. Units are visited by the user in a sequence. Each user is assigned an initial number of cycles for each unit. This initialization is based on a pre-test or on a constant sequence  $(1, 1, 1, \dots, 1)$ , that is, each unit is cycled one time and, if needed, more cycles are added. A unit exit test follows. When the exit test is failed, one cycle is added to the cycle path vector and the unit is presented again. The user type is represented as a cycle path vector  $c_1, c_2, c_3, \dots, c_n$  where  $n$  is the number of units. The user types are initialized to constant vectors  $(i, i, i, \dots, i)$ ,  $i = 1, 2, 3, \dots$ . Note the difference between the initialization of an individual user and a prototype user. The  $i$  represents the best estimate of the number of cycles required for a given user. This number  $i$  is later adjusted by the system based on the user answers. Using one of the presentation strategies, we repeat items within the same unit, never returning to the previous unit (in this simplest scenario). Examples of presentation strategies are:

1. present definitions/statements and then ask questions in the multiple choice form;
2. present definitions/statements and then ask questions in the form of a missing word;

3. start with presenting questions and then follow schedules 1 or 2.

Clearly there are many different strategies with different degrees of efficacy.

The distance from a new partial (not all units completed) or complete cycle path to known user types is measured, and the cycle path is assigned to the closest user type. This membership determines the number of cycles needed to complete the given unit. The unit is completed when the exit criterion is met. The criterion is defined either by an exit test or by a score level. All cycle paths are clustered, and representative members of resulting clusters are used as user types. The variations of the cycle path vector components  $c_1, c_2, c_3, \dots, c_n$  indicate the degree of difficulty of a unit. In the ideal case of uniformly authored units, the components  $c_i$  would be constant for each user type. If there is more than one presentation strategy, we can utilize all strategies in order to minimize the number of operations. One identification procedure can be formulated as follows: assuming that we have more than one strategy available, we can alternate them and attempt to match subcycle paths with user types.

*Case 2.* We assume that one unit depends on a set of other unit-prerequisites. Dependent units can be visualized as children nodes of parent nodes in a graph. For dependent units, we made a requirement about test questions. Each test question about a child node is phrased so that it points to the remediation unit(s). The frequency of visits of prerequisite units is used for modifying the number of cycles estimated from individual units.

A user can be either under- or over-classified. If a user is incorrectly under-classified a penalty is paid, because of additional cycles and exit tests that need to be presented when remediation occurs. The penalty for finding the correct remediation also applies. The over-classification produces a penalty for redundant exit tests and operations within a unit. The main objective of the cluster analysis is the classification of users. This means that each future user can be, after several units, classified as falling within a certain class. Likewise, his or her path through the system can be optimized. By optimal, we mean that the cost of the user's learning counted in the number of operations (cycles) and unsuccessful tests is minimal for a given teaching strategy and stopping criterion.

## 5. Examples

One of the learning strategies (Case 2) allows the user to return to the prerequisites. This strategy assumes that authoring provides test questions that allow us to identify the cause of the incorrect answer. We assume that each unit is in the form of a small tree with three or four parents. A good example is again the Konigsberg bridge theorem (Euler circuit theorem) (Fig. 1).

We will take a look at a fragment of a session with the system. For example, the system can ask the user to list the properties of a presented graph. Possible answers

might have the following outcome:

1. An incomplete list of properties (degree of vertices, connected graph) indicates a definition for remediation.
2. An incorrect statement about the existence of an Euler cycle indicates that more work at the theorem unit level should be done.

### 5.1 Types of Experiments

The current system is concerned with several types of experiments. The first two experiments are based on abstract presentations and example-based presentations. The abstract presentation provides more compressed information than the example-based.

The goal of these experiments is to collect sequences of the numbers of cycles, which are then clustered into several typical sequences. These typical sequences will become prototype paths, and new users will be classified according to these sequences. The third type of experiment deals with dependencies of units, and the goal is to collect the number of cycles under the assumption that some units may have one or more prerequisite units.

### 5.2 Numerical Experiments

We have performed classroom experiments and also simulated users mathematically [7]. The number of users was chosen to be approximately the same for each cluster. Data simulation was provided by random variables based on a heuristic from teaching classes, and experiments with systems that collected data (PHP system) or with stateless systems (no memory, JavaScript, and HTML systems).

The GJ experiment (Fig. 2) shows the variation in the number of cycles for groups of users. More details of this problem can be found in [6]. The significance of these experiments is that they demonstrate that a user can require dramatically more prerequisite information than the average user. In this experiment, the learning goal is to be able to apply the GJ method for solving a system of linear algebraic equations. The prerequisite for learning the GJ method is that the user is trained to do only one elimination step at a time. In the numerical experiments of the user we generated cycle paths. In the simplest scenario (no remediation using previous units) the penalty for incorrectly identified numbers  $c_i$  is:

$$op \times (c_i - c) \tag{1}$$

when the correct number of cycles  $c$  is smaller than  $c_i$ , and:

$$p \times (c - c_i) \tag{2}$$

for  $c$  greater than  $c_i$ . The term  $op$  stands for the number of operations for one unit, and  $p$  is the penalty for additional unit exit tests.

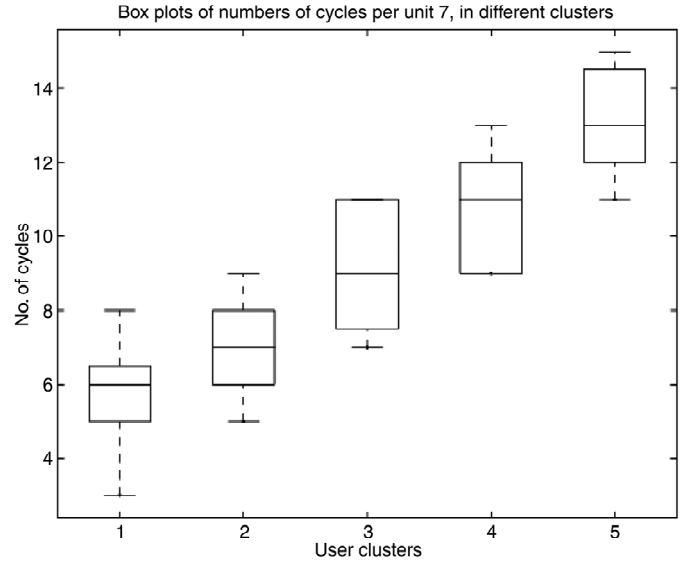


Figure 2. Numbers of cycles required by users in GJ experiment.

## 6. Identification of User Types, Simulation Results

In the numerical simulations of the users, we defined a sequence of units and generated a number of cycle paths with different probability distributions for passing each unit. For each user, these probabilities were mutually dependent and slightly increasing after each additional information (cycle) provided to the user. This process created cycle paths that were clustered. Clustering algorithm used a variant of the standard K-means procedure, where the distance was defined as a weighted Euclidean distance in the space of paths. The optimal number of clusters was selected to minimize the sum of all distances of paths from corresponding means. In order to reduce the final number of clusters, the criterion was penalized with the Schwarz's BIC penalty term. Fig. 3 shows the result of one of the simulation experiments, which selected five clusters of cycle paths. The left part of the figure shows the distribution of the total number of path cycles and their classification. The box plots of Fig. 4 compare, for all five clusters, the distribution of numbers of cycles used for one of the units. The number of users in individual clusters is not realistic and is due to assumptions in our numerical simulation. The distributions of paths, through all units, actually split the set of paths into five subsets of cycle path. Each new user, after passing several units, can be classified on the basis of the distance to different clusters. In subsequent units, the user is provided with the information (the number of cycles) optimal for his or her class. The optimal number of cycles can be set, for instance, as 75% quantile of the corresponding distribution. In the box plot, this number corresponds to the upper bound of the box. An incorrect classification of a user can increase either the time (number of cycles) spent at the unit when the unit has already been mastered, or it can increase the number of attempts needed to pass the final unit test while the knowledge of the user is not yet sufficient. In the simplest scenario (no

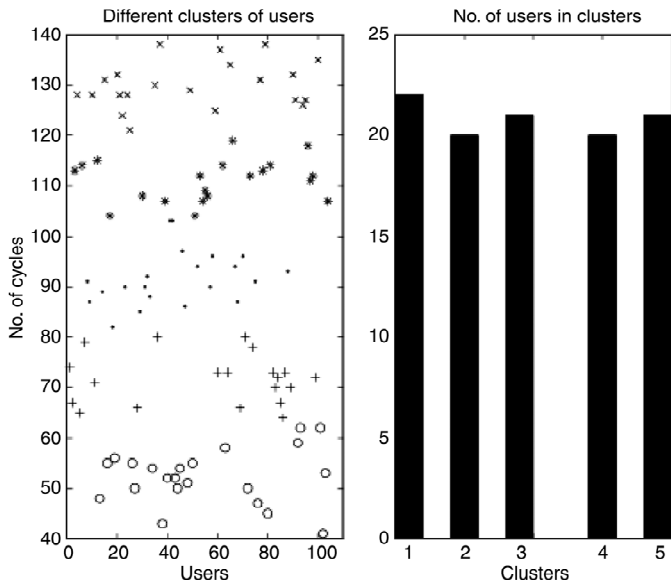


Figure 3. Clusters of users.

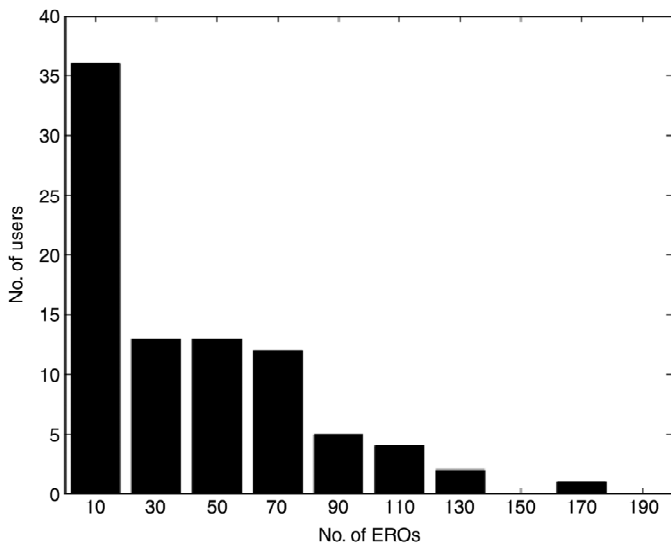


Figure 4. Example of distribution of cycles per one unit for different clusters of users.

remediation), the overall penalty for incorrectly identified user is the sum of penalties for all units. Thus, as the model is described with the aid of probability distributions (and identified by methods of mathematical statistics), we cannot avoid an error of the first kind, that is, even if a user is classified properly, he or she will need more operations than is recommended for that user's type (actually, in the example described the probability of such an error is 25%).

Nevertheless, the mean cost of such a mistake is by definition statistically smaller than the cost of incorrect classification. Moreover, the classification is a dynamical process, and the user can be reclassified while still operating within the system. As mentioned above, there are two types of penalties:

$$P = PE + PR \quad (3)$$

that are assessed for suboptimal behaviour of the system. The  $PE$  term represents the penalty for a unit exit test

due to a return to the unit. The return to a unit follows a failed test in a child unit. The  $PR$  term represents the penalty for finding out the correct remediation unit.

## 7. Implementation on the WWW

One of the requirements we set for the system implementation is that the system not requires proprietary components and be able to run on the web under the TCP/IP protocol. The current implementations are relatively simple and are based on static HTML, HTML/PHP documents. The documents are retrieved either by a PHP control module or using a PHP control module.

For the simplest experiments, where there is no permanent record of the user activities, we developed and used a system written in JavaScript and HTML. A simple system that writes user information on a server was written in PHP. We experimented with such a simple system that can be modified without intensive software skills. Consequently, the developer can experiment with a variety of user-scoring and information-sequencing procedures. All components (servers with PHP modules) were available free of charge on both platforms (Win, Linux). In our experiments, we used a simple graphical user interface design at this stage of the system development, as our primary goal is to collect data. The user data analysis will be utilized as a tool for additional refinement of the model for in-classroom use. The user interface or, more accurately, the information the user can access is clearly important and deserves additional attention. The future WWW will work with XML data structures (see, e.g., [8]). We set as one of our future goals the development of a protocol where different servers with archives of XML domain data communicate automatically among themselves and complement their information. For example, one server can specialize in tutorials such as normal equations and some other server can offer the Konigsberg theorem domain. This approach will establish a network of developers for authoring and revising tutorial domains. The following is an example of a unit when the information is marked up using XML. The information is coded using trees. One tree captures the dependency structure of units or subunits. The other trees capture definitions, questions, and answers. The XML information can be parsed and can then be searched for required content, and presentation pages can be created and passed to the control module.

```
<?xml version="1.0"?>
<unit_structure id="unique unit structure
  identification, e.g. 1s Normal Equations">
<children>
unit id's that are children of this unit,
  e.g. 2 Over-determined equations;
  3 Matrix Multiplication
</children>
<parents>
unit id's that are parents of this unit,
  e.g. 0. Least-squares fitting problem
</parents>
```

```

<friends>
unit id's that are related but neither
  children or parents, e.g. 4 matrix
  decomposition
</friends>
</unit>

```

```

<answer>
A'Ax
</answer>
</test>
</UNIT>

```

The following shows a fragment of unit 2 from the above tree structure:

```

<?xml version="1.0"?>
<UNIT id="2 Over-determined equations">
<statement number="1">
Let A be an  $m \times n$  matrix. For the case  $m > n$ 
  a system  $Ax = b$  of linear equations is
  called over-determined because there are more
  equations than unknowns. Such a system usually
  does not have an exact solution.
</statement>
<test number="1">
<question>
An over-determined system of equations usually
  _____ solution. (1) does not have (2) has
  more than one solution (3) has exactly one
</question>
<answer>
1
</answer>
</test>
<statement number="2">
Over-determined system  $Ax = b$  is such that the
  matrix A usually does not reproduce right-hand
  side  $b$  for any vector  $x$ . This means that  $Ax = b$ 
  is not exactly true for any choice of vector  $x$ .
</statement>
<test number="2">
<question>
The over-determined system  $Ax = b$  is such that the
  matrix A usually does not _____ right-hand
  side  $b$  for any vector  $x$ . This means that  $Ax = b$ 
  is not true for any choice of vector  $x$ .
</question>
<answer>
reproduce
</answer>
</test>
<statement number="3">
A typical linear least-square fitting problem
  can be written in the matrix for  $m$  as an
  over-determined system  $Ax = b$ , the matrix is
   $m \times n$ ,  $m > n$ . This rectangular system can be
  transformed into the  $n \times n$  system of normal
  equations  $A'Ax = A'b$ , the  $A'$  is the transpose
  of A.
</statement>
<test number="3">
<question>
A rectangular system  $Ax = b$  can be transformed
  into the  $n \times n$  system of normal equations
  _____= $A'b$ , the  $A'$  is the transpose of A.
</question>

```

This XML unit is parsed and a set of HTML/PHP pages is generated. This set is then presented by a control module. One of the appealing features of the XML data structures is that they become a standard for the new WWW, and that gives reasonable hope for the wide spread of XML-coded tutorials and other items suitable for public exchange over the Internet.

## 8. Conclusion

We have considered a class of flexible information-cycling systems and associated user-adaptive systems. In the latter class, the cost of the user path is lowered because of the user orientation and adaptation of the system. The optimization is achieved with the help of probabilistic modelling of the user chance (and its evolution) to master the units, and the statistical evaluation of user data. The optimization is important for classroom applications where the time the user must spend practising is minimized. We are currently in the process of establishing a procedure for extensive user data collection along with both online and offline data processing. In the future, on-demand information systems will have evolving natural language interface and applications will run within the semantic framework [9, 10]. The more pressing issue is how to author units and domains, as this is a very time-consuming process. We envision that in the near future servers will provide archives of tutorial material coded in XML and agents maintaining these servers will provide updates of units available on other friendly servers.

## Acknowledgement

The author is grateful to P. Svacek and P. Volf for their support with data analysis and programming.

## References

- [1] G. Weber, Adaptive learning systems in the World Wide Web, in J. Kay (Ed.), *UM99—User Modeling 1999, Proc. of the 7th International Conf.*, Banff, Canada, 1999, 371–377.
- [2] G. Weber & M. Specht, User modeling and adaptive navigation support in WWW-based tutoring systems, 1997, available at <http://www.psychologie.uni-trier.de/projects/ELM/elmart.html>.
- [3] P. Brusilovsky, Adaptive hypermedia, *Modeling and User-Adapted Interaction*, 11, 2001, 87–110.
- [4] J. Smid, M. Obitko, & P. Volf, Model parameters and model performance, *Proc. of the UM2001—Workshop on Empirical Evaluation of Adaptive Systems*, Sonthofen, Germany, 2001, 25–32.
- [5] S. Weibelzahl, Evaluation of adaptive systems, *UM01—User Modeling 2001, Proc. of the 8th International Conf.*, Sonthofen, Germany, 2001, 292–294.

- [6] J. Smid, P. Svacek, & J. Smid Jr., Processing user data for intelligent tutoring models, *Workshop within KDD-2000: Post-Processing in Machine Learning and Data Mining: Interpretation, Visualization, Integration, and Related Topics*, Boston, MA, August 2000, 50–55.
- [7] P. Volf & J. Smid, Randomized evaluation of states of an ITS, *Modeling and Simulation of Systems Conf. (MOSIS '01)*, Ostrava, Czech Republic, 2001, 227–232.
- [8] T. Berners-Lee, J. Hendler, & O. Lassila, The semantic web, *Scientific American*, 284 (5), 2001, 34–43.
- [9] A. Kobza, Generic user modeling, *User Modeling and User-Adapted Interaction*, 11, 2001, 49–63.
- [10] Wahlster, Semantisches Web und Wissenmanagement, available at <http://www.dfki.de/~wahlster>.

US government agencies and teaches computer science and mathematics courses at Computer Science Department, Morgan State University, Baltimore, USA, at the rank of assistant professor. His current research interests are web-based tutoring and information systems, mathematical modelling, WWW-based computation and dialogues in distributed systems.

## Biography



*Jan Smid* has studied physics, mathematics and computer science. Dr. Smid graduated from Charles University, Prague. Since his graduation, Dr. Smid held a number of industrial and teaching positions in Europe and in the USA. Dr. Smid was involved in areas ranging from computational mechanics, programming corporate graphics systems to being a research scientist for NASA/GSFC projects. Currently Dr. Smid consults for the