An adaptive learning environment in DICE system with TDD model

Li-Ren Chien, Daniel J. Buehrer, Chin Yi Yang

An adaptive learning environment in DICE system with TDD model

Li-Ren Chien [1], Daniel J. Buehrer[2], Chin-Yi Yang[3]

National Chung Cheng University

Key words: *DICE, Test-Driven Development, Test-based Grader, Parse-Tree based, Automatic Grading, Computer Aided Assessments, Adaptive Learning*

Abstract:

We introduce an adaptive learning environment named DALM (DICE adaptive learning model) based on DICE test-driven development (TDD) model that was implemented in a parse-tree based automatic on-line grader. In our opinion, there are three variables in DALM. The individual differences (I) classify the learners into several groups. The training method (T) that equivalent to TDD model in DICE can be controlled by DICE system. The learning outcome (O) presents the learning performance of learners with an individual difference level after chosen TDD model. The learning outcome is related to the individual difference and the training method denotes as O = f (I, T). A model learning phase will rank the <I, T> pair by learning outcome. The learner will be adapted to the best fit training method (T) by his/her individual difference (I) base on the result of model learning phase. We expect such an environment can adapt different kinds of learner to suitable DICE TDD teaching unit.

1 Introduction

We aim to develop an adaptive learning system for improving the learning performance of programming learning. To achieve our aim, we segment our research plan into four stages.
In first stage, we need an automatic grading system. Since 2005, we have commenced to establish DICE system for test-based assignment tutoring and problem solving environment. [1] All training work including assign practise, turn-in and assessment could be run on DICE system. Dice has worked in HKHS for 1 year.
As following, we refer training method criteria and TDD concept to establish a new training model for programming learning which named DICE TDD Model [2]. It provides sixteen kinds training methods for learners.
In the third stage, we conduct Kolb's [3, 4] learning style instrument as the test item of individual difference. We will unearth the best fitness between learning styles and training methods which will result in satisfying learning outcome. We will prove different learner needs different training method in DICE system.
As educational psychologists suggested to match training methods to individual difference is important, we will develop an adaptive learning system for individuals. Reliable instruments about individual difference and student database in school will be conducted in DICE. In early work, we will discover the best fitness between the variable of individual and learning performance. When having a big population, we will classify the best fitness between learning performance and training criteria. Finally, DICE will get a new criterion for best learning.

2 Brief Introduction to DICE

DICE [1] was implemented in an OS-independent, distributed, client-server environment, with a parse-tree-based automatic assessment system. The teacher starts the testing plan of a programming language (C or Java) by making a problem set. He is asked to organize his

Figure 1: Client of student

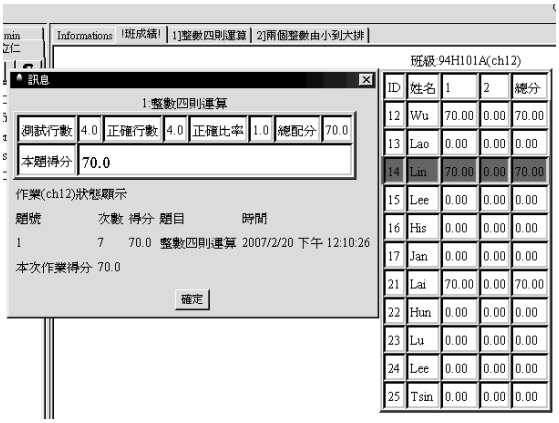

Figure 2: Grading result

Figure 3: Monitor a particular client

Figure 4: Client of instructor

problem descriptions, input datasets, and standard output to a specified directory. The students' data can be stored in either text files, Excel spreadsheets or a database that could be connected to by JDBC. After the teacher starts the server at a particular port, the students can login from an IP network, and so can other teachers.

The servers can be deployed on the same host by using different ports or on different hosts by using the same port. A load balancer will distribute the clients to the different hosts based on the loading on each host. At the server side, the system manager can monitor the actions of the whole system. He can dialog every client and supervise what the client is doing, or terminate the client's session.

The teachers can get all of the functions of the server from any client computer. He also can get the parse tree of each student's answer. Throughout the term, the teacher can merge the testing results over the semester into an Excel file.

A student will login and be assigned to a server after a teaching unit. He is asked to solve those problems within a stipulated time range. After login, he can look over the problem set,

and turn answers to the server. DICE will judge the answers by executing the executable file or recompiling the code and executing it. The student's program will be fed with the input dataset that was prepared by teachers. The system compares the answers of students with standard results to decide the score that he gets. The result is immediately sent to the student. For a lightweight and database-free system, all information is stored in files. The system information, like the examination questions from the teachers, the answers and scores of the students and so on, are organized with pure text files and directories. We also have a connection by JDBC to traditional databases for student information for some built-in environments. We provide four levels of plagiarism detection to avoid some cheating actions like answer resending, adding white space, variable renaming, semantic copying and so on. We used SableCC [5], a parser tool that was developed by Etienne Gagnon. An Abstract Syntax Tree (AST) is built by a C or Java parser for each student's program. The AST is translated to a Polish Reverse Notation (PRN) form for the evaluation. As we translate the student's answer to a PRN string then we can do pattern matching on it. Because the PRN was translated from the AST, we can treat it as semantic symbols of the original string.
Some screenshots of DICE were shown from Figure 1 to Figure 4.
In summary, we have implemented an Automated Assessment System for test-based assignment tutoring. According to this system, we can push the students of a computer language course to put more effort into improving their coding ability and significantly reduce the burden of grading the programs.

3 DICE TDD Model

After running DICE for years, we found some well-known problems of a test-based grader. These caused the underachievers to be eliminated from the DICE system. One problem is that only clearly defined questions with a completely specified interface can be used. It leads students to focus on output correctness first and foremost, and it does not encourage or reward for good performance while testing [6]. One of the perceived shortcomings is that its inflexibility prevents assessment of more complex questions [7]. When a complex question arrived, we found that some underachievers just sat before the computer and waited for the bell ringing. We need a more sophisticated mechanism to help underachievers.
Over the past five years, the idea of including software testing practices in programming assignments within the undergraduate computer science curriculum has grown from a fringe practice to a recurring theme [8]. Some researchers may argue that starting too early with a test-first approach can lead to the "paralysis of analysis" [9]. We believe the TDD with instructor made-test suites will help overcome the shortcomings of test-based graders.
As Figure 5 showing, a program assignment was given after each teaching unit. The instructor makes a test plan consisting of problems (or so called test cases). A test case was composed by the test-based grader, like data sets and TDD-like data sets.
In our opinion, the TDD model in DICE [2] can be coordinated to two dimensions, one for test cases in a teaching unit ($\mathbf{X_\Omega}$) and the other for the test units in a test case ($\mathbf{Y_\Gamma}$). The axis X_Ω represents the coupling degree of test case sequences in a teaching unit. At the same time, the axis Y_Γ represents the coupling degree of test unit sequences in a test case. In our aspect, X_Ω represents more concepts in a teaching unit than Y_Γ, since Y_Γ represents more programming skills than X_Ω.

Table 1, DICE TDD classification by $\mathbf{X_\Omega}$- $\mathbf{Y_\Gamma}$

$\mathbf{X_\Omega}$ / $\mathbf{Y_\Gamma}$	$X_{\Omega\varnothing}$ (x=0)	$\mathbf{X_{\Omega>}}$ (x=1)	$X_{\Omega m}$ (x=2)	$\mathbf{X_{\Omega\vdash}}$ (x=3)
$Y_{\Gamma\varnothing}$ (y=0)	Exploration (0)	Concept-like (2)	Concept-modular (4)	Concept-Instruction (7)
$\mathbf{Y_{\Gamma>}}$ (y=1)	Skill-like (1)	Like (5)	Concept-modular, Skill-like (9)	Concept-Instruction, Skill-like (11)
$\mathbf{Y_{\Gamma m}}$ (y=2)	Skill-Modular (3)	Concept-like, Skill-Modular (8)	Modular (12)	Concept-Instruction, Skill-Modular (14)
$\mathbf{Y_{\Gamma\vdash}}$ (y=3)	Skill-Instruction (6)	Concept-like, Skill-Instruction (10)	Concept-modular, Skill-Instruction (13)	Instruction (15)

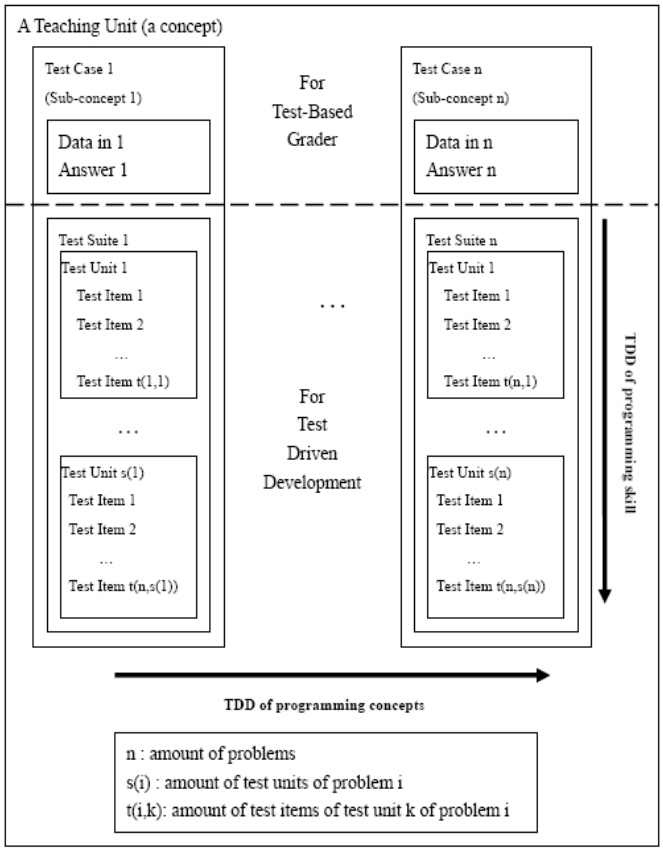

Figure 5: The TDD Model in DICE

4 DICE Adaptive Learning Model (DALM)

As educational psychologists suggested that matching training methods to individual differences is important, we introduce an adaptive learning model in DICE for individuals. Individual differences are never equal to learning style. According to the literature review of individual differences, it could be classified by developed taxonomy.

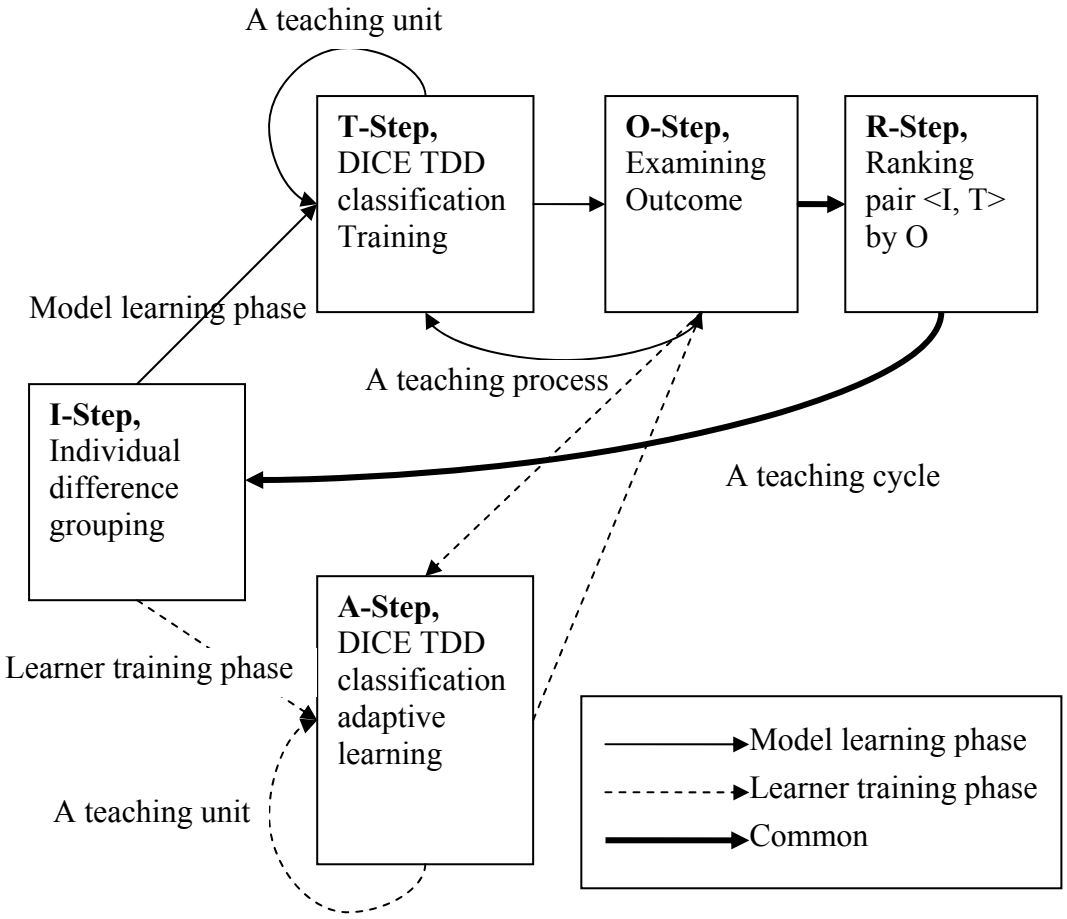

Figure 6, DICE adaptive learning model

In our opinion, there are three variables in DICE adaptive learning model (DALM). The individual differences (I) are discrete attributive explanatory variables that classify the learners into several groups by learner's individual differences. We will take Kolb's learning style inventory (KLSI) as an instance and will be described in following section. The second one will focus on the training method (T) that equivalent to TDD model in DICE is also a discrete explanatory variable but can be controlled by DICE system.
Finally, the learning outcome (O) is a continuous dependent variable presents the learning performance of learners with an individual difference level after chosen TDD model. In our aspect, the learning outcome is related to the individual difference and the learning method denotes as O = f (I, T).
As figure 6, DALM consist of learning parse, learning parse and grouping parse.

4.1 Grouping phase

The purpose of grouping phase is to classify learners to different group by their individual differences.

- **I-Step:** Individual difference grouping. The learners will be grouped to different levels by their individual differences. DALM will support two kinds of mechanics to accept the classification of learner made by instructor.
  The first one is to deliver a reliable questionnaire beforehand to learners to group their individual differences. For example, KLSI [3] divides learning style to AC (Abstract Concept) -CE (Concrete Experience) and AE (Active Experimentation) -RO (Reflective Observation) plane. KLSI was assigned two discrete values in 1984 [3] and 14 values in

2005 [4] on every coordinate. We linarite KLSI by I = C*d +E where C present AC-CE axis as E present AE-RO axis and d = 2 for KLSI in 1984 as d=14 for KLSI in 2005. Now we can classify the learner's individual differences to d^2 levels by developed questionnaire.
The other way to classify learners is adapted DICE to a database by JDBC that stored learner's information. The attributes in tables or views can be combined arbitrary to group the learners. For example, an instructor can group his leaner by their sex and mathematic score from a known view to some levels.

4.2 Model learning phase

The purpose of model learning phase is to build a best adaptive learning pair <Individual difference, DICE TDD classification training> by learning outcome.
In DALM learning phase, DICE will group learners in the system to different levels by system manager's setting before course commencing. After that, the teaching unit will be classified by DICE TDD model to different levels. The learners are assigned to different DICE TDD training level by statistics methods during a teaching cycle. After a teaching cycle with teaching units, an examination will be held to measure the learning outcome of learners. Finally, DICE system will collect set of triple < I, T, O> to DALM database year after year. DALM rank dynamical < I, T> pair by O at each end of learning phase.
The model learning parse circulates from I step, T step, O step to R in a learning cycle. As figure 6, TOP steps are described as following:

- **T-step:** DICE TDD classification training. In DICE TDD model [2] we have defined a plane with conceptual (X) and training (Y) axes. The value in each axis was descried from exploration to instruction by the relationship of content of a teaching unit. As table 1, we linarite DICE TDD classification from exploration to instruction to sixteen levels. For each teaching unit in DICE system was classified by the relationship of test suites and the connection of test units. The classification could be assigned by the instructor or by the machine learning use the definition of DICE TDD classification. Different kinds of DICE TDD teaching unit of the same concepts in a teaching unit will be random assigning to number in groups of I step for statistics. A teaching process may involve many such steps. For example, an instructor may decide a teaching process consists of three teaching units and each teaching unit possesses three different kinds of DICE TDD practical training.
- **O-step**: Examining outcome. After period of T step, the system will take a normal exam by the grader or by traditional methods to assess the outcome of training. The level of outcome is set to a continue value form 0 to 100 as a common evaluation for different <I, T> pairs. A teaching process should take many different T-O cycles to collect triple <I, T, O> to DICE system.
- **R-step:** Ranking pair <I, T>. After a teaching process, DICE system will ranking those <I, T, O> triples by statistics for instances by ANOVA. The result of ranking will be simplified to a triple <I, T, P> that means I kinds of learner using T training method will get the priority of P. The R step will be repeated over and over again to make the priority of <I, T> can be modified dynamically. This mechanic should make the adaptive performance of DALM more facilely.

4.3 Leaner training parse

After model learning parse, DALM can be used on learner by A-step. The R-step via O-step should be hold for model learning when need.

- **A-Step:** TDD classification adaptive learning. For we have classified the learner to different groups by individual difference at I step. And we have a priority set of <I, T> in DALM after R step of model learning parse. Now we can assign a most suitable TDD teaching unit for every leaner. The basis of adaptation is different kind of individual difference learner is suite for different level learning of exploration to instruction.

5 Conclusion and Future Work

In this paper, we introduce a new adaptive learning model based on the test driven development training method and the individual difference by the learning outcome. We developed a three phase stage consists of the grouping, the model learning and the TDD training to adapt learner to the most suitable training method. DALM collects triple <individual difference (I), DICE TDD classification (T), learning outcome (O)> as samples year after year. For we set our DICE system to operate in a particular institution, we expect those samples will adapt perfectly in the place. The future work of DALM is to develop a mechanic to collect samples from different DICE servers that be distributed in difference place. We plan to develop a comprehensive adaptive learning model to DICE in the future.

References:

[1] Li-Ren Chien, D. Buehrer and Chin Yi Yang, DICE. A Parse-Tree Based On-Line Assessment System for a Programming Language Course. *The Third Conference on Computer and Network Technology*. HsinChu, Taiwan, 19-20 April, 2007.
[2] Li-Ren Chien, D. Buehrer and Chin Yi Yang, Using Test-Driven Development in a Parse-tree Based On-line Assessment System. *The IADIS International Conference e-Learning 2007*. Lisbon, Portugal, 6-8 July, 2007 (accepted)
[3] Kolb, D.A. and Fry, R. Toward an applied theory of experiential learning. *In Thories of Group Process, G.L. Cooper(ed.), John Wiley and Sons, Inc*., New York, NY, PP.33-54, 1975
[4] Kolb A.Y., Kolb D.A. The Kolb's learning style inventory-version 3.1 2005 *technical specifications, Boston, MA*: Hay Resource Direct. 2005.
[5] Gagnone, Hendren L J. SableCC-an Object-oriented Compiler Framework. *Proceedings of Tools 26: Technology of Object-Oriented Languages*. 1998.
[6] Stephen H. Edwards. Improving student performance by evaluating how well students test their own programs. *ACM Journal of Educational Resources in Computing*, 3, 3, Article 01. 2003.
[7] Christopher, D., David, L. and Jams, O. Automatic Test-Based Assessment of Programming: A Review, *ACM Journal of Educational Resources in Computing*, Vol. 5, No 3, Stempember 2005. Article 4. 2005.
[8] Stephen H. Edwards. and Manuel A. Pérez-Quiñones. Experiences using test-driven development with an automated grader. *Journal of Computing Sciences in Colleges.* Volume 22, Issue 3, January 2007
[9] Don Colton., Leslie Fife., and Andrew Thompson. A Web-based Automatic Program Grader, *Proc ISECON 2006,* v23.

Author(s):

Li-Ren, Chien, PhD candidate.
Chung Cheng University, Department of Computer Science and Information Engineering
#168, University Rd, Min-Hsing, Chia-Yi, Taiwan, R.O.C
clj@cs.ccu.edu.tw

Daniel J. Buehrer, Professor.
Chung Cheng University, Department of Computer Science and Information Engineering
#168, University Rd, Min-Hsing, Chia-Yi, Taiwan, R.O.C
dan@cs.ccu.edu.tw

Chin Yi Yang, Miss.
Chung Cheng University, Department of Information Management
#168, University Rd, Min-Hsing, Chia-Yi, Taiwan, R.O.C
vivian@hkhs.tnc.edu.tw