

# What about Teaching Learners in Accordance of Their Aptitude? A Study Based on DICE TDD System in Learning to Programming

Li-Ren Chien<sup>1</sup>, Daniel J. Buehre<sup>1</sup>, Chin-Yi Yang<sup>2</sup>,  
Checchen Liao<sup>2</sup> and Chyong-Mei Chen<sup>3</sup>

1. Department of Computer Science and Information Engineering, Chung Cheng University

2. Department of Information Management, Chung Cheng University

3. Department of Applied Mathematics, Providence University

Taiwan

clj@cs.ccu.edu.tw

## Abstract

Researchers in instructional psychology have demonstrated that adapting instructional methods and teaching strategies to accommodate key individual differences have led to improved performance. The Chinese educational philosopher Confucius argued teaching students in accordance of their aptitude in almost 500 BC.

Nowadays we designed DICE TDD system to enhance the learning performance of programming for learners. We are not only to be sure the benefits of TDD DICE system but also to recognize what kind of learners enhancing more while adapting TDD DICE.

We conducted an experiment containing of 330 tenth grade participants in the basic computer concepts course content with programming language C under the same instructor in Hsin Kuo High School in Taiwan R.O.C. We classified all learners into four kinds of learning style following Kolb's learning style inventory and randomly assigned each type of learning style into DICE with TDD and DICE with NonTDD.

The result of this research shows DICE TDD system benefits the learners in Converger most, Accommodator secondly then Diverger and Assimilator in Kolb's

learning style.

**Keywords:** Auto Grading System

## 1. Introduction

We believe in appropriate methods for training and programming development are mutually intricate and enduring matters. Programming is the core in the computer course [1]. In behalf of improving the learning performance of programming, we have a series of plans since four years ago. Practice is one of the most important steps in learning the art of computer programming [2]. First, we aimed to give more assignments and immediately score response to students. Given these institutional and class management limitations, the decision Hsin Kuo high school has been to automate the homework grading process. This allows students to electronically submit programming assignments and receive instantaneous feedback. This method of on-line assessment, referred to as DICE, was established and initiated in February of 2006 [3]. At present, DICE system has been in operation for Hsin Kuo High School's computer programming course for about three years, and over 3000 students have taken advantage of this evaluation method. DICE reached the initial goal of assigning more practice and evaluation immediately.

As anticipated, some well-known problems with the system have been encountered, which has caused a considerable number of users to be removed. Consequently, as it has been determined that this particular system should be upgraded to a more sophisticated evaluation method.

Due to the scarcity in the literature and the importance within the educational programming environment, there is a commitment here to better shed light on what attributes and attitudes students possess that leads to the ability to, without undo difficulty, learn programming, and how this understanding can aid the overall student population. Based on the interesting finding of Bostrom, R.P., Olfman L. and Sein, M.K. in 1990, we extend their finding to design DICE system into TDD and Non-TDD under exploration-oriented and instruction-oriented criteria.

TDD is a code development strategy in which one always writes a test case before adding new code [4]. The benefits of TDD are to build software better and faster and give the programmer a great degree of confidence in the correctness of his code [5]. After implementing DICE with TDD, we commenced to design training material from the ACM UVA online judge problems [6]. We applied DICE TDD and DICE Non-TDD in the computer programming curriculum.

Finally, we most want to know the adaption between the character of learners and training methods on learning performance in programming. We classified all learners into four kinds of learning style following Kolb's learning style inventory and randomly assigned each type of learning style into DICE with TDD and DICE with NonTDD. The result of this research shows DICE TDD system benefits the learners particularly the Converger, Accommodator and Diverger in sequence on Kolb's learning style.

## **2. Background**

We based on DICE system [7] to develop DICE with TDD [5] in 2007 and argued to develop an adaptive learning system [8]. We have three papers to present our ideal. Nowadays we would like to know how the interaction between training method with DICE with TDD, DICE with nonTDD and the individual difference-learning style [8] on the performance in programming learning. The details are in the appendix.

## **3. Evaluation**

We held an experiment to compare the effectiveness of DICE with TDD and DICE with NonTDD under the different learning style types. Our experimental group applied DICE with TDD training methods, whereas the control group is applied only DICE, denoted by Non-TDD.

### **3.1 Research Model**

We base on the theatrical development of end-user training on outcomes to design a research model for programming learning [9].

As mentioned, some learners needed to be given more guidance. Consequently, DICE with TDD guides learners to solve problems gradually. An instructor needs to divide the problem into sub problems and let students conquer each sub problem. After they have conquered every sub problem, then the whole problem will have been solved. It's more intensive than Non-TDD. According to the literature review of Jones in 2004, most results of the studies have nice performance in TDD learning. Hence, we have Hypotheses 1.

Participants in the DICE with TDD will score significantly higher on learning performance measures than participants in the Non-TDD group.

Researcher in instructional psychology has demonstrated that adapting instructional methods and teaching strategies to accommodate key individual differences including learning style has led to improved performance [10]. Snow in 1991 [11] also proposed that learners differ profoundly in what they do in learning and in their success in any particular learning situation. Therefore, we got the Hypotheses2 inference. Participants in training method with learning style will have different influence on learning performance.

In additional to the hypotheses 2, the relationship between training method and learning style has been studied. The interaction of individual training method and learning style may affect learning outcome through learning process and it was proven by Bostrom in 1990, Huey-Wen, C., 2000 and Simon 2000. We consider programming ability in Kolb's learning style inventory that learners with Converger in KLSI are best at finding practical uses for ideas and theories. They have the ability to solve problems and make decisions based on finding solutions to questions or problems. These learning skills are important for effectiveness in specialist and technology careers [12]. Furthermore, programming can be an engineer career. Math and science scores were shown to have strong correlation with programming learning performance [13]. Some conclusions drawn about learning styles in End User Software Training are Convergents, who combine active experimentation and abstract conceptualization, perform better than those subjects with other learning styles [9].

Individuals with an abstract learning mode have the ability or experience to discover the rules and structures inherent in a new domain such as computer software [9]. Another learning style in abstract learning mode is Assimilator. People with this learning style are best at understanding a wide range of information and putting it into concise,

logical form. Individuals with an Assimilating style are less focused on people and more interested in ideas and abstract concepts. The Assimilating learning style is important for effectiveness in information and science careers [12].

According to the computer anxiety research, the highest level of computer anxiety was possessed by Divergers and the lowest by the Convergents [14]. The fact is that the Divergers and Convergents would score at the opposing ends of this range. In Kolb's theory, a person labeled Diverging performs better in situation that call for generation of ideas, such as a brainstorming session. People with a Diverging learning style have board culture interests and like to gather information. They are interested in people, tend to be imaginative. In formal learning situations, people with the Diverging style prefer to work in groups, listening with an open mind to different points of view and receiving personalized feedback [12].

So far as to Accommodator, their tendency in may be to act on "gut" feelings rather than on logical analysis. In solving problems, individuals with an Accommodating learning style rely more heavily on people for information than on their own technical analysis [12].

Referring to the literature review and Kolb's learning style inventory, the character located at Converger and Assimilator in KLSI more agree with programming than Diverger and Accommodator. Therefore, learners whose characters are at Diverger and Accommodator may need more assistance from the instructors. Judging from their attribute, the assistance they need are to guide them to think gradually and DICE with TDD is under the instructional concept to develop. We suppose TDD has a stronger positive effect on learning performance for Divergers and Accommodator than Assimilator and Converger.

Hypotheses2.1: The instruction-oriented group (TDD) has a stronger positive effect

on learning performance for Divergers and Accommodator than Assimilator and Converger. Hypotheses 2.2: The exploration-oriented group (Non-TDD) has a stronger positive effect on learning performance for Assimilator than non-Assimilator.

### 3.2 Experiment Design

We conducted a post-test only control group design and treatment was applied by random assignment considering learning style and training method. We distinguished all learners into four kinds of learning style by KLSI testing. There are 330 samples in the experiment taking the programming course. Table 1 presents the sample size of the learning styles in training methods respectively and the total sample sizes.

Table 1 Random Assignment of Learning Style and Training Method

Item	Category	Training Method		Summary
		Non-TDD	TDD	
Learning Styles	Assimilator	62	57	119
	Accommodator	25	30	55
	Converger	58	65	123
	Diverger	18	15	33

After establishing of function concept, we started to apply the training method to each group for one month. Finally, we had an examination to all participants and took the grades as their score of the course and also as the scale of learning performance.

### 3.3 Data Analysis

We most concern the interaction between training method and learning style on learning performance. Hence, we analysis the data by two-way ANOVA for there are two interaction terms. Considering the sample sizes, unbalanced two-way ANOVA should be conducted for the unequal sample

sizes which are necessarily conducted by multiple regression models since it can demonstrate the quantitative information. We adopt stepwise variable selection algorithm based on AIC criterion to select suitable explained variables.

In data analysis process, we denote DICE with TDD as  $TM$ . Since four levels of learning style, we code learning style into dummy variables with  $KLST1$ ,  $KLST2$  and  $KLST3$ . Converger is designed as the baseline group and it corresponds to dummy coding  $(KLST1, KLST2, KLST3) = (0, 0, 0)$ . By logic, Accommodator, Assimilator and Diverger are respectively denoted as  $(1, 0, 0)$ ,  $(0, 1, 0)$  and  $(0, 0, 1)$ . The regression model is

$$E(Y) = \mu = \beta_0 + \beta_1 KLST1 + \beta_2 KLST2$$

$$+ \beta_3 TM + \beta_4 KLST2 \times TM$$

34 Where  $Y$  is dependant variable – learning performance and its expectation given a corresponded group is denoted as  $\mu$ .

The result is displayed in Table 2  
The estimated model is

$$\hat{\mu} = 13.1037 - 7.1389KLST1 + 5.9931KLST2 + 11.0978TM - 10.1069TM * KLST2,$$

where  $\hat{\mu}$  is the estimator of  $\mu$ .

This regression analysis outcome expresses that DICE with TDD can increases about  $\beta_3$ , 11.0978 in learning performance. After examining the alternative hypothesis  $H_1: \beta_3 > 0$ , positive effect for the learners in DICE with TDD has P-value 0.00035, which indicates that DICE with TDD benefits the learners.

Table 2 Output of Regression Analysis

Variable	Regression Coefficient	Estimator	Estimated standard error	P-value
Intercept	$\beta_0$	13.1037	2.5213	<0.0001
KLST1	$\beta_1$	-7.1389	3.7033	0.0548
KLST2	$\beta_2$	5.9931	3.9172	0.1270
TM	$\beta_3$	11.0978	3.2544	0.0007
TM KLST2		-10.1069	5.4179	0.0630

This regression model suggests considering learning style with training method simultaneously. It claims that the DICE with TDD and assimilator has the interaction designating that it has different effect on the learners with Assimilator than other learning style.

DICE with TDD has effects depending on whether a learner is Assimilator or not. This evidence is illustrated in  $\beta_4 = -10.1069$  and the estimator of the regression coefficient of interaction term between DICE with TDD and KLST2. After testing alternative hypothesis  $H_1: \beta_4 < 0$ , DICE with TDD has minor effect on assimilator learners in learning performance than other styles, P-value 0.0315 reaches the statistical significant level  $\alpha = 0.05$ .

The model estimates that a learner with assimilator has 19.0968 in DICE with nonTDD. Nevertheless, the same character learners have grades with 20.0877 on learning performance after DICE with TDD applying. It indicates that DICE with TDD only can increase 0.9909 for the assimilator learners. Contrarily, the learners who are non-assimilators applying DICE with TDD have higher grades about 11.0978 than those in non-TDD. Hence, this study offers us the sufficient support to instruct the learners by individual style difference.

So far as to Accommodators, under uniform training method the learners in Accommodator usually have poor grade on learning performance. The model declares

that a learner with accommodator learning style has fewer grades 7.1389 than a learner with Converger.

We test this negative trend. The P-value is 0.0274 exhibiting that this study has strong evidence to verify a learner with accommodator has poor performance in programming learning. The statistical model presents an interested result; learners with Diverger character have the same grade range with Converger  $H_1: \beta_1 < 0$  in the two training groups.

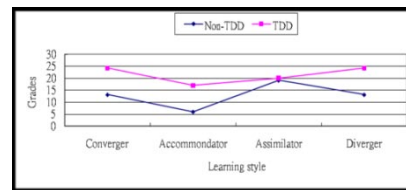


Fig.1. Research Model

### 3.4 Result

After the data analysis conducting, we get the significant findings that DICE with TDD benefits the learners more than DICE with non TDD.

Considering the interaction of training method and learning style, DICE with TDD has more positive influence on Accommodator, Converger and Diverger than Assimilator. Two training methods have no significant difference on Assimilator.

### 4. Conclusions and Future Work

The result from data analysis shows DICE with TDD benefits most learners on learning performance. Considering the interaction of training method and learning style, DICE with TDD has more positive influence on Accommodator, Converger and Diverger than Assimilator. Two training methods have no significant difference on Assimilator.

From the observation of training process, we found learners in DICE with TDD group more enjoy discussing with their mates than DICE with Non-TDD. As to DICE with

Non-TDD meeting puzzles, they tend to rely on the direction of the instructor.

While taking learning style into consideration, the result shows DICE with TDD assisting Accommodator, Converger and Diverger more than Assimilator. According to the literature review, Accommodator and Diverger will need more instruction in programming learning. Hence, we defined the hypothesis2-1 DICE with TDD has a stronger positive effect on learning performance for Divergers and Accommodator than Assimilator and Converger. However, Converger has best learning performance in computer software using in the past literature review. But computer software usage is never equal to the programming which not only uses the computer software but also need logical thinking especially the training material and tests referred from ACM UVA online judge system where the problem sets combine computer science and mathematics concepts. Assimilators are important for effectiveness in information and science career and will be good at programming in nature. The result is very reasonable except Assimilator all need to be guidance gradually.

There is less difference between Assimilator conducting DICE with TDD or DICE with Non-TDD. However, TDD has its limitation; it is difficult to use in situations where full functional tests are required to determine success or failure. Some researchers may argue that starting too early with a test-first approach can lead to the "paralysis of analysis" [15].

Though we have no evidence to display DICE with non-TDD has positive effect on learning performance, we suggest applying Non-TDD to Assimilator after concerning the weakness of TDD in literature review.

## 5. References

[1] Robin, A., Rountree J. and Rountree N. "Learning and Teaching Programming: A

Review and Discussion," Computer Science Education, (33-2), 2003, pp. 137-172.

[2] Christopher, D., David, L. and Jams, O. "Automatic Test-Based Assessment of Programming: A Review," ACM Journal of Educational Resources in Computing, (5:3), September 2005.

[3] 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, 2007, Putrajaya, Malaysia.

[4] Beck, K. Test Driven Development: By Example, Addison-Wesley, 2003.

[5] Li-Ren Chien, D. Buehrer and Chin Yi Yang. "Using Test-Driven Development in a Parse-tree Based On-line Assessment System", IADIS International Conference e-Learning, Lisbon, Portugal, 2007.

[6] <http://acm.uva.es/problemset/>

[7] 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, 2007.

[8] Li-Ren Chien, D. J. Buehrer, Chin-Yi Yang. (2007) "An Adaptive Learning Environment in the DICE System with a TDD Model" in Interactive Computer Aided Learning, Villach, Austria, 2007.

[9] Bostrom, R.P., Olfman, L., and Sein, M.K. "The Importance of Learning Style in End-User Training," MIS Quarterly, 1990(14:1), pp. 101-119.

[10] Bergin S. and Reilly R. "Programming: Factors that Influence Success," Technical Symposium on Computer Science Education, Proceedings of the 36th SIGCSE technical symposium on Computer Science Education, 2005, pp. 411-415.

[11] Snow, R.E. "Individual Difference in the Design of Educational Programs," American Psychologist (41:10), October 1986, pp. 1029-1039.

[12] Snow, R. E., "Aptitude-treatment Interaction as a Framework for Research on

Individual Differences in Psychotherapy, ” Journal of Consulting and Clinical Psychology, 59(2), 205-216, 1991.

[13] Kolb A.Y., Kolb D.A. “The Kolb’s Learning Style Inventory-Version 3.1 2005 Technical Specifications, ” Boston, MA: Hay Resource Direct, 2005.

[14] Davis, F.D. and Yi, M.Y. “Improving Computer Skill Training: Behavior Modeling, Symbolic Mental Rehearsal, and the Role of Knowledge Structures, ” Journal of Applied Psychology, (89:3). pp. 509-523.

[15] Don Colton., Leslie Fife., and Andrew Thompson. A Web-based Automatic Program Grader, Proc ISECON 2006, v23.

## Appendix

### 1. Auto Grading System: DICE

The DICE project springs from the requirements of a teaching assistants’ job in a programming language course. Previously, we usually asked students to deliver their programming assignments weeks’ later after class. This policy caused a serious plagiarism problem. So we decided to implement an on-line assessment system that could judge the students code in the course automatically and immediately.

DICE was implemented to be in an OS-independent, distributed, client-server environment, with a parse-tree-based automatic assessment system. We also plan to have an automatic grading and intelligent tutoring system to be based on parse trees.



The teacher starts his testing plan of a programming language (C or Java) by making a problem set. He is asked to organize his problem descriptions, input datasets, and standard output to a specified directory. Each student’s data will be stored in 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 to the judge server 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 with each client, to supervise what the client is doing, or terminate the client’s session.





The teacher can login as a client and get more rights than students after passing the teacher validation. A teacher 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 course unit. There are many problems waiting for the student. He is asked to solve those problems within a stipulated time range. After login, he can look over the problem set, and send his answer as a source file or an executable file. The system will judge his answer by executing the executable file or recompiling his code and executing it. The student's program will be fed with the input dataset that was prepared by teachers. The system compares the output of student's program with standard output file 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.

The plagiarism problem, which we mentioned at the beginning of this section, should be solved in the system. We think that

the key is providing rapid, concrete and immediate feedback by the automated assessment tool to which students can submit their code. Because most students will be concerned only with their own problem solving, the probability of cheating will come down to that of traditional testing. But we still provide for 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, 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.

## 2. Test Driven Development

TDD (Test Driven Development) is a code development strategy that has been popularized by extreme programming. TDD is an evolutionary approach to development which combines test-first development, of writing a test before writing just enough production code to fulfill that test, and refactoring. In TDD, one always writes a test case before adding new code.

The following sequence of a TDD cycle is based on Beck's theory. The first step is to quickly add a test, basically just enough code

to fail. Next you run your tests, often the complete test suite although for sake of speed you may decide to run only a subset, to ensure that the new test does in fact fail. You then update your functional code to make it pass the new tests. The fourth step is to run your tests again. If they fail, you need to update your functional code and retest. Once the tests pass, the next step is to start over. You may first need to re-factor any duplication out of your design as needed.

The biggest benefit of TDD is to help build software better and faster. It offers more than just simple validation of correctness; it can also drive the design of a program. By focusing on the test cases first, one must imagine how the functionality will be used by clients (in this case, the test cases). Therefore, the programmer is only concerned with the interface, and not the implementation. This benefit is complementary to “design by contract”, as it approaches code through test cases rather than through mathematical assertions or preconceptions. What is the primary goal of TDD? One view is that the goal of TDD is specification and not validation. In other words, it’s one way to think through your design before writing the functional code. Another view is that TDD is a programming technique. As the argument of Ron Jeffries, the goal of TDD is to write clean code that works.

However, TDD has a limitation; it is difficult to use in situations where full functional tests are required to determine success or failure. Examples of these are GUIs (graphical user interfaces), programs working with relational databases, and some that depend on specific network configurations. Management support is essential. Without the entire organization believing that TDD is going to improve the product, management will feel that time spent writing tests is wasted.

### 3. Cases of TDD in Learning

TDD provides benefits that learners experience for themselves. It is applicable on small projects with minimal training. It gives the programmer a great degree of confidence in the correctness of his code. It is easier for learners to understand and relate to, more than traditional testing approaches. It promotes incremental development, promotes the concept of always having a “running version” of the program at hand, and promotes early detection of errors introduced by coding changes. Finally, it encourages students to test features and code as they are implemented.

As TDD seems attractive, the idea of using TDD in the classroom is not revolutionary. Computing and information technology educators have begun to call for the introduction of TDD into the curriculum. 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]. Some researchers may argue that starting too early with a test-first approach can lead to the “paralysis of analysis”. TDD has gone to school since 2001; Table 1 is a review of TDD studies applied in learning.

Table2-1 Previous Studies of TDD in Learning ]

Study	Dependent Variables	Results
Edward, 2003	Software Quality/reliability	TDD Significantly higher
	Programmer confidence	TDD Significantly higher
	Preference of TDD	Significantly higher after

Kaufma , 2003	Software quality/reliability	TDD Significantly higher
	Programmer productivity	TDD Significantly higher
	Programmer confidence	TDD Significantly higher
Muller, 2002	Software quality/reliability	No significant difference
	Programmer productivity	No significant difference
	Program understanding	TDD Significantly higher

#### 4. DICE with TDD

Since 2005, we have established DICE system as a test-based assignment, tutoring, and problem solving environment. All training work, including assigned practice, turn-in and assessment, can be run on DICE system. DICE has been working at Hsing Kuo High School in Taiwan R.O.C. for over 3 years. A running DICE System is shown in Fig5.

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 was that only clearly defined questions with a completely specified interface could be used. It led students to focus on output correctness first and foremost, and it did not encourage or reward good performance while testing. Another of the perceived shortcomings was that its inflexibility prevented assessment of more complex questions. When a complex question arrived, we found that some underachievers just sat before their computer and waited for the bell to ring. So we needed a more sophisticated mechanism to help underachievers.

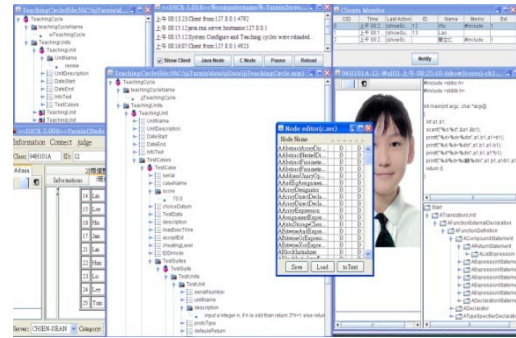
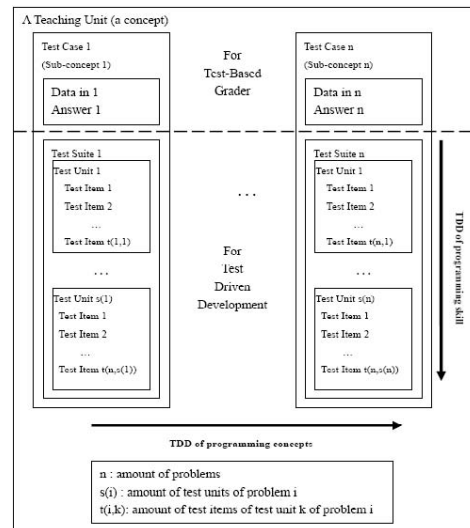


Fig. 5: A running DICE system

In the second stage, we referred to training method criteria and TDD concepts to establish a new training model for learning programming, which was named the DICE TDD Model. It provided sixteen kinds training methods for learners.

In the third stage, we conducted Kolb's] learning style instrument as a test of individual differences. We found the best fit between learning styles and training methods which would result in satisfactory learning outcomes. We proved that different learners needed different training methods in the DICE system



#### 5. Learning Style

Researcher in instructional psychology has demonstrated that adapting instructional

methods and teaching strategies to accommodate key individual differences including learning style has led to improved performance. The mechanism through which an individual's learning style influences his/her learning is explained by learning style theories. In this section, we will review the definition of learning style and introduce Kolb's learning style inventory (KLSI).

Kolb's Learning Style Inventory views learning as a discovery process that incorporates the characteristics of problem solving and learning. The effectiveness of KLSI has been widely accepted by adult learning researchers .

Kolb's learning style inventory, as defined and modified in 2005, includes four learning styles: Converger who are the best at finding practical uses for ideas and theories. They have the ability to solve problems and make decisions based on finding solutions to questions or problems. Individuals with a Converging learning style prefer to deal with technical tasks and problems rather than with social issues and interpersonal issues. Diverger performs better in situations that call for generation of ideas, such as a brainstorming session. People with a Diverging learning style have board culture interests and like to gather information. Assimilators are best at understanding a wide range of information and putting it into concise, logical form. Individuals with an Assimilating style are less focused on people and more interested in ideas and abstract concepts. Accommodators have the ability to learn from primarily "hands-on experience". They enjoy carrying out plans and involving themselves in new and challenging experiences. Their tendency may be to act on "gut" feelings rather than on logical analysis.