A Structural Analysis of Agile Problem Driven Teaching

Pradip Peter Dey, Gordon W. Romney, Mohammad Amin, Bhaskar Raj Sinha, Ronald F. Gonzales and S. R. Subramanya

Abstract
Agile problem driven teaching has dynamically changing aspects with a wide range of interpretations allowing flexibly effective teaching adaptable in many environments. The central concept is that all major teaching activities are driven by a set of problems with agility for adaptation in a wide variety of teaching environments. Some typical problem solutions are demonstrated by the instructor within the scope of the course learning outcomes. Problems could be based on realistic or abstract situations. This paper exhibits how agile problem driven teaching activities are easily included in course contents and correctly mapped to course learning outcomes. Agile problem driven teaching is not the same as problem-based learning (PBL). Structural comparison of PBL and agile problem driven teaching reveals important differences between them with pedagogically important consequences. Although PBL is highly popular in certain environments, it is not necessarily appropriate for all learners and all topics since the teaching methods may not be dynamically modified. This paper proposes, as a key differentiator, a logical interpretation to dynamic behavior of APDT that highlights the role agility plays in certain aspects of APDT by altering pre-planned activities and dynamically adjusting to changing situations.

Key Words
Agile teaching, course learning outcomes, direct instruction, problem based learning.

INTRODUCTION
Crouch (2011) states in an article in Reader’s Digest, that “research universities are no place for undergraduates. Professors at big research universities are often more interested in doing research with graduate students than teaching your child. . . . So, they tend to host huge lectures and then foist undergrads off on teaching assistants who may or may not be supervised.” In a similar tone, other complaints about many aspects of teaching are also heard. The USA is losing its leadership in science, math and engineering education, according to the 2007 report of the National Academies, “Rising Above The Gathering Storm: Energizing and Employing America for a Brighter Economic Future,” (Committee on Prospering in the Global Economy of the 21st Century et al. 2007). Wallis (2008) concludes that “Recent test results show that US 10th-graders ranked just 17th in science among peers from 30 nations, while in math they placed in the bottom five.” There is enough evidence that this educational trend is temporally coupled with a closely followed severe economic down-turn. Undoubtedly, continuation of this trend is a danger to the U.S. economy, security, standard of living, and leadership. We live in a rapidly altering world, with a global job market, global educational competition, escalating energy problems, mounting trade imbalances, a globally integrated economy, and an unprecedented financial crisis. In order to compete in the global job market, the
new U.S. generation needs to acquire skills for solving current exigent problems in modern educational environments. Problem solving skills have provided advantages to individuals, humans, families and nations. Great nations are built by great problem solvers and education is the most important system for developing and enhancing problem solving strategies. Well educated engineers, technologists and scientists are in demand due to global competition. In order to build a vigorous economy with sustainable growth, creative educated problem solvers are needed in the USA. However, colleges and universities are not succeeding in producing innovative problem solvers. Several recent strategies show some improvements in student learning in certain environments (Borman, 2005). However, nationwide enhancements have not been realized despite these isolated successful cases. A new innovative approach with a distinctive combination of agile teaching, problem solving and direct instructions may bring more desirable rapid changes to the educational environments.

Agile Problem Driven Teaching (APDT) is closely related to Problem-Based Learning (PBL). Barrows (1985) is given credit for the classic model of PBL, which has two key features: “a rich problem is used that affords free inquiry by students, and learning is student-centered” (Hmelo & Evensen 2000). PBL is the educational process by which problem solving activities and instructor’s guidance facilitate learning. PBL is the pathway by which students “learn how to learn”. It challenges students to think critically, analyze problems, be pro-active, and discover and use pertinent learning resources (Barell, 2006; Duch 2011; Savin-Baden 2003). APDT is similar to PBL; however, it combines problem based free inquiry with direct instructions (Gersten & Carnine, 1986) in order to achieve the course learning outcomes. Support mechanisms of Scaffolding methods and metacognitive strategies (Holton & Clark, 2006) are also added to APDT in order to get additional benefits for the learners. The rest of the paper presents important aspects of APDT including a definition of APDT followed by a narrative account relating APDT to other popular methods such as PBL and scaffolding, a structural analysis of APDT and an example application followed by some concluding remarks. We emphasize a logical interpretation to dynamic behavior of APDT that highlights the role agility plays in certain aspects of APDT by altering pre-planned activities and dynamically adjusting to changing situations.

DEFINING AGILE PROBLEM DRIVEN TEACHING

APDT is primarily a teaching method although it supports activities for promoting learning. The primary goal of APDT is to have the teachers teach well so that their students can achieve high level of learning outcomes. A general definition of APDT is given below.

APDT is composed of three components:

1. A problem set component:
   This is a set of problems with two subsets such that each subset satisfies the same class of learning outcomes, known as the course learning outcomes (CLO’s). The first subset is referred to as the teacher’s subset (T-Set) which is used by the instructor to demonstrate the CLO’s. The
other subset is referred to as the students’ subset (S-Set) which is used by the students for practicing problem solving.

2. A teaching component:
This is a set of interactive activities performed by one or more teachers and a group of students. The activities include (2a) a set of guided instructions provided by one or more instructors using the T-Set, (2b) comprehensive supervision and facilitation of students’ problem solving activities utilizing the S-Set and (2c) all phases of solving including initial problem analysis, derivation of learning needs, generation of ideas & topics, formulation of research questions, identification of resources, problem reanalysis, proposition of solutions, reviews and formative assessments and selection of the best solutions initially in this order. All teaching activities are driven by the T-Set and S-Set in a plan; however, the plan may be changed dynamically with the support from the agile process component.

3. An agile process component:
This is a set of guidelines for dynamically changing the plan and sequence of activities based on feedbacks and formative assessments. It is divided into two subcomponents: (3a) one subcomponent for supporting students’ learning process, (3b) another subcomponent for supporting instructor’s teaching process.

The composition of the three components in APDT synergistically produces comprehensive instructions with a wide variety of interactions. This synergistic relation is explained in the next section. In APDT, what is to be taught next is decided by a set of problems; this is why it is problem driven. Each of the two subsets of this set must satisfy the CLO’s. One subset, the T-Set, is used by the instructor to demonstrate problem solving using appropriate supporting mechanisms. The other subset, the S-Set, is used by the students. Student teams investigate problems and generate learning topics among other activities in order to master the course learning outcomes. These topics are integrated into teaching by dynamically changing the teaching plan as problem solving continues. The changes are accommodated by the agile process component which provides flexible adjustments to the teaching process based on feedbacks. One of the subcomponents of the agile process component is specially designed to provide supports to the students as in scaffolding (Holton & Clark, 2006; Simons & Klein, 2007).

It is reasonable to explain certain aspects of APDT in comparison to other well-known methods such as PBL and scaffolding. We are impressed with the supports provided to learners by the scaffolding method (Holton & Clark, 2006; Simons & Klein, 2007) and adopted these support strategies in APDT. We are inspired by the achievements of PBL and immensely influenced by its rich mechanism. Like PBL, APDT emphasizes problem solving activities. However, unlike PBL, an important aspect of APDT is that the instructor plays an active role in teaching scenarios. This action includes providing direct instructions, stimulating group discussions, and in giving innovative guidance. In a PBL environment, the instructor is primarily a facilitator for students’ problem solving activities. This is why some researchers identify PBL as one of the minimally guided approaches and criticize it for its deficiency in providing direct
instructions (Krischner, Sweller & Clark 2005). In order to avoid this criticism, APDT is designed as a completely guided approach to have proper supervision of students’ activities. In APDT, the instructor is more than a facilitator; he takes complete responsibility for all academic activities although these activities are driven by two sets of problems, the T-Set and the S-Set. The use of the T-Set, in addition to the S-set, enables the instructor to cover all learning outcomes in a timely manner, in case students do not perform their work on time. Besides, demonstration of problem solving with the T-set provides additional supports needed by students for their problem solving activities which are advocated by the scaffolding method (Holton & Clark, 2006; Simons & Klein, 2007). In PBL, students are usually divided into teams to work on problems; the problems are expected to play an important role in learning activities, a common practice in medical science (Schmidt, 1998; Schmidt, 2000). In this strategy the instruction of the topic is organized around problem solving tasks. Some of these tasks involve problem analysis followed by relevant information gathering. Students may continue the analysis phase by their discovery and identification of possible solutions in conjunction with pros and cons surrounding each proposed solution (Adamowski, Frydecka, & Kiejna, 2007). Often, problems are complex and may not even be well defined. The discovery of new knowledge and its acquisition is made as students work through a problem. The role of the instructors in the PBL environment is mainly to facilitate the students’ effort. Students take more responsibility for their own learning and are engaged in discovery learning in the sense that students discover and work with content that they selected to be necessary in order to solve the problem. It is assumed that by working through the problem, students are better able to internalize the problem and comprehend the underlying concepts and fundamental relationships needed to solve the problem.

Patel, Groen, and Norman (1993) argue that teaching basic science with the PBL approach in a clinical context may have the disadvantage that contextualized basic knowledge is difficult to separate from the clinical problems into which it has been integrated. This poses a difficulty in distinguishing the basic science knowledge components. Although PBL students generated more elaborate explanations, they had less coherent explanations and produced more errors (Patel, Groen, and Norman 1993). Another common criticism of the PBL method is that students may not recognize what might be important for them, hence the need for the facilitator to be extra careful to assess each student’s prior knowledge. We introduce the model of APDT in which the instructor’s role more closely approaches the traditional instructor role with complete responsibility for the course learning outcomes. In APDT, the instructor’s presentation is more dynamic and can easily diverge to cover a variety of relevant topics according to inquiries received from students and other sources. Similar to its PBL counterpart, each student team is given a complex problem to solve within the scope of the course learning outcomes. However, unlike PBL, the role of the instructor in this approach is elevated to providing periodic coaching, proper guidance and direct instruction in order to accelerate the learning process. In particular, the instructor plays a key role contextualizing the problem, actively participating in the research, and analysis, but, also in generalizing the knowledge. The instructor becomes a facilitator once the initial body of knowledge needed to solve the problem is gathered. This gives students an opportunity to ingeniously construct the final solution in a comprehensive well-defined technique.
In the APDT method, the open problem along with student inquiries and the collective information gathering process drives the lectures, discussions, and analytical reasoning. It is at this stage where the agility in instruction becomes apparent, possible and critical. “When students cannot learn the way we teach them, we must teach them the way they learn” (Dunn, 1990). What is more important is willing to make changes to the teaching plan dynamically, during execution time, than having a fixed plan. The instructor must have extensive knowledge of the subject to efficiently process information and resolve students’ questions and further suggest new directions for information gathering. In this method, the new knowledge is shared among all teams in a form of a presentation by the instructor and the student teams. The flexibly open discussions provide an opportunity to further clarify issues, misrepresentations and misinterpretation associated with the problem and the newly acquired information. Agility in teaching/learning and grading helps to overcome the different challenges faced in different environments, by different learners, for different topics. According to Glickman "Effective teaching is not a set of generic practices, but instead is a set of context-driven decisions about teaching. Effective teachers do not use the same set of practices for every lesson . . . Instead, what effective teachers do is constantly reflect about their work, observe whether students are learning or not, and, then adjust their practice accordingly” (Glickman 1991). Agility is the basis of APDT and consequently, it may combine a variety of teaching strategies. In addition to PBL, many other teaching and learning methods can be employed including the following: lecture (Cashing 1990; Instructional Methods Information, 2011), technology-based teaching learning (Kearsley & Shneiderman, 2011), game-based learning (Prensky, 2004; Van, 2008), experience-based learning (Andresen, Boud & Cohen, 2000), inquiry-based learning (Eick & Reed, 2002; Papert, 1980), thinking-based learning (Swartz, Costa, Beyer, Reagan, & Kallick, 2008), community-based learning (Owens & Wang, 1996); brain-based learning (Johnson & Lamb, 2007), work-based learning (Bailey 2003; Cunningham, Dawes & Bennett, 2004), project-based learning (Helic, Maurer, & Scerbakov, 2004), team-based learning (Michaelisen, Knight, & Fink 2008), web-based Learning (Chumley-Jones, Dobbie, & Alford, 2002; O’Neil & Perez 2006), and participatory learning (Barab, Hay, Barnett, & Squire, 2001). There is no conflict between these methods and APDT since it can easily incorporate these methods. The main strategy in scaffolding is to provide adequate supports to students when they attempt comparatively difficult problems. This strategy is found to be very useful for certain areas including mathematics (Holton & Clark, 2006; Simons & Klein, 2007). APDT learns from the success of scaffolding and attempts to provide adequate support to students whenever needed. Scaffolding in combination with metacognitive strategies would enhance students’ problem solving abilities.

Cognition about cognition is metacognition. Metacognitive strategies are processes that one uses to monitor and control one’s cognitive activities for ensuring that a goal, such as correct problem solving, is achieved (Brown, 1987). These processes help to regulate and oversee cognitive functions. Recent research demonstrates that metacognitive strategies are effective in reducing errors in problem-solving tasks requiring analytic reasoning (Alter, Openheimer, Epley, & Eyre, 2007). APDT embraces metacognitive strategies and scaffolding for problem solving activities (Holton & Clark, 2006). The main goal of
APDT is to take problem solving activities to an electrifying level and integrate them into course topics governed by course learning outcomes of a modern curriculum. The resulting synergy produces an innovative combination of all major contributions of recent pedagogical approaches and will, hopefully, bring about inspiring changes not just to courses but to the educational system at large. Great educational changes may lead to grate societal changes with a positive impact on economy and standard of living.

PBL is often characterized as one of the minimally guided approaches (Krischner, Sweller & Clark 2005). After reviewing all major contributions, they conclude “Although unguided or minimally guided instructional approaches are very popular and intuitively appealing, the point is made that these approaches ignore both the structures that constitute human cognitive architecture and evidence from empirical studies over the past half-century that consistently indicate that minimally guided instruction is less effective and less efficient than instructional approaches that place a strong emphasis on guidance of the student learning process” (Kirschner, Sweller & Clark 2005). In a recent study, direct instructions are combined with aspects of PBL and obtained improved results in terms of learning outcomes (Swartz, Costa, Beyer, Reagan & Kallick, 2008). In APDT, direct instructions, agility and the productive PBL aspects of problem solving, the support strategies of scaffolding are creatively combined in a new structural framework.

STRUCTURE OF AGILE PROBLEM DRIVEN TEACHING

The structural aspects of APDT are schematically shown in Figure 1. All major teaching activities are driven by a set of problems and this illustrated problem set is represented at the top of Figure 1. A subset of this set of problems is selected by the instructor in order to demonstrate all the course learning outcomes. This subset is presented on the left side of Figure 1, referred to as the teacher’s subset (T-Set). There is another subset of problems which is used by students for practicing problem solving and referred to as the students’ subset (S-Set). Based on interests, students form teams and each team selects a problem or a subset of problems from the S-Set for analysis and solution. The boxes represent “activities” that lead to a problem solution. The typical control flow for problem solving is shown with unidirectional arrows between boxed activities. However, the process is agile and the control flow is very flexible in the sense that any problem activity may be addressed at any step in the process. Therefore, there are bidirectional arrows in Figure 1 that show how a problem solver can go back and forth from one activity to another activity. This flexibility of control flow significantly distinguishes APDT from PBL. In addition, all student activities are properly guided and supervised in APDT; this is indicated in Figure 1 by making the right side boxes and unidirectional arrows thicker than those of the left side. The central vertical column is drawn for the topics to be covered based on the learning outcomes of the course. However, these topics can be changed due to the contributions of the problems being solved. According to Swartz, Costa, Beyer, Reagan and Kallick (2008) the “thinking strategy map” for “skillful problem solving” involves the following questions: “(1) What is the problem? (2) Why is there a problem? (3) What are some possible solutions? (4) What would result from these solutions? (5) What solution is best and why?” These questions help in getting started with the investigation of the problems which may generate many
follow-up questions and then map to the elements of the problem solving activities of Figure 1 through examination of consequences.

Figure 1: Structure of Agile Problem Driven Teaching

The implementation of the structural elements of APDT may be simplified by carefully using technological tools. In particular, in one scenario, students and the instructor each will have Tablet PCs, in a networked environment, with a collaborative, interactive teaching tool such as Dyknow vision. In this networked environment, the instructor’s display is broadcast to student’s Tablets allowing students to synchronously follow the instructor’s perspectives and individually annotate discussions with their own stylus pen to their personal Tablet for a given problem solving task. The instructor can also permit a student to lead the class from his or her Tablet. This connectivity clearly facilitates a powerful medium for students to collaborate and share as they search for additional knowledge to solve the problem. Among useful features of Dyknos is a user friendly tool called “panel submission”. Panel submission is most useful in APDT, for in this mode each team can, anonymously, submit their findings and request comments from the instructor. The instructor can quickly scan through all the student submissions and select
one or more panels to share with the rest of the class. It is through this flexible sharing that the instructor can clarify misunderstandings, make additional comments or presentations, or provide new ideas for research and evaluation (Dey et al 2009). The instructor can also opt to privately give comments to student submissions. This promotes meaningful interactions with students. Many educators have reported an increase in student participation when Tablets are creatively used in the class. Panel submission and the follow up discussions often clarify issues, misrepresentations or misunderstandings without embarrassing a student. Additionally, the instructor can utilize panel submission to focus on problem analysis or new, relevant knowledge shared by students. Adjusting teaching methods based on learner feedback may play a vital role in multicultural learning environments (Dey et. al. 2009).

**MAPPING A COURSE ONTO AGILE PROBLEM DRIVEN TEACHING**

The APDT structure easily maps to standard course contents in a wide variety of subjects since the interpretation of APDT is legitimately broad. As an example mapping exercise, a graduate course on mathematical foundations of computer science (National University course number: CSC 610) is considered for structural elements of APDT. National University course curricula are designed based on course learning outcomes and, therefore, the majority of course-related activities must support accomplishing the course learning outcomes. The course learning outcomes for the mathematical foundations course are:  
- **CLO1**: Construct a computational model for a given problem and examine its consequences;  
- **CLO2**: Describe properties of computational models;  
- **CLO3**: Prove: For every Non-deterministic Finite Automaton there is a regular expression;  
- **CLO4**: Prove that a given language is Context-Free;  
- **CLO5**: Construct a processor for a given Context-Free language;  
- **CLO6**: Construct a Turing Machine for a given computational problem;  
- **CLO7**: Prove results of union, concatenation and complementation of various recursively enumerable sets.

A set of 17 problems drove the learning activities of the course. Out of the 17 problems, nine were selected for the T-Set which were numbered as T₁, T₂, T₃, T₄, T₅, T₆, T₇, T₈, T₉. The other eight were put into the S-Set which were numbered as S₁, S₂, S₃, S₄, S₅, S₆, S₇, S₈. The set of 17 problems are specified below in Table 1 with the supporting CLO’s listed with each problem along with the generated topics.

<table>
<thead>
<tr>
<th>Problem Number followed by Problem Description</th>
<th>CLO’s</th>
<th>Generated Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁: Construct a Finite Automaton for ab*. Examine if the automaton accepts (1) abbb, (2) baba, (3) a, and (4) abab.</td>
<td>CLO₁</td>
<td>Finite Automata, sets representing Regular Expressions, a Language as a set of strings, Regular Languages, strings accepted by Finite Automata</td>
</tr>
<tr>
<td>T₂: Describe the closure properties of Regular Expressions.</td>
<td>CLO₂</td>
<td>Union, Concatenation and Kleene star</td>
</tr>
</tbody>
</table>

Table 1: A Set of 17 Problems for CSC610
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Cls</th>
<th>Properties of Regular Expressions</th>
<th>Proofs, Non-deterministic Finite Automata</th>
<th>Pushdown Automata and Context-Free Grammars for Context-Free Languages</th>
<th>Automata as Processors of sets of strings, programming language patterns, visualization</th>
<th>Turing Machines as the class of most powerful processors, construction of Turing Machines for given problems</th>
<th>Turing Machines as the class of most powerful processors, construction of Turing Machines for given problems</th>
<th>Proofs for union of certain sets</th>
<th>Proofs for concatenation of certain sets</th>
<th>Turing Automata, sets representing Regular Expressions or Regular Languages, strings accepted by Finite Automata</th>
<th>Union, Concatenation and Kleene star, properties of Regular star</th>
<th>Proofs, Non-deterministic Finite Automaton</th>
<th>Pushdown Automata and Context-Free Grammars for Context-Free Languages</th>
<th>Automata as Processors of sets of strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₃:</td>
<td>Prove: For every Non-deterministic Finite Automaton there is a regular expression.</td>
<td>CLO₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₄:</td>
<td>Prove that $L₄ = { a^n b d c^n : n &gt; 0 }$ is Context-Free.</td>
<td>CLO₄</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₅:</td>
<td>Construct a Pushdown Automaton for $L₅ = { e^n : n \geq 0 }$.</td>
<td>CLO₅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₆:</td>
<td>Construct a Turing Machine (TM) for $L₆ = { c^n d a^n b : n &gt; 0 }$.</td>
<td>CLO₆</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₇:</td>
<td>Construct a Turing Machine (TM) for $L₇ = { a^n b^n a^n : n &gt; 0 }$.</td>
<td>CLO₇</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₈:</td>
<td>If $L₈₁$ and $L₈₂$ are sets representing Context-Free languages then prove that their union, $L₈₁ \cup L₈₂$, is also Context-Free.</td>
<td>CLO₈</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T₉:</td>
<td>If $L₉₁$ and $L₉₂$ are sets representing Context-Free languages then prove that their concatenation, $L₉₁ \cdot L₉₂$, is also Context-Free.</td>
<td>CLO₉</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₁:</td>
<td>Construct a Finite Automaton for $aba^*$. Examine if the automaton accepts (1) abaa, (2) baba, (3) ab, and (4) abab.</td>
<td>CLO₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₂:</td>
<td>If $L₁$ and $L₂$ are regular languages then $L₁ \cup L₂$, $L₁ \cap L₂$, $L₁^*$ are also regular.</td>
<td>CLO₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₃:</td>
<td>Prove: For every Non-deterministic Finite Automaton there is a regular expression.</td>
<td>CLO₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₄:</td>
<td>Prove that $L₄ = { a^n d e b^n : n &gt; 0 }$ is Context-Free.</td>
<td>CLO₄</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S₅:</td>
<td>Construct a Pushdown Automaton for $L₄ = { j^n b c a^n : n \geq 0 }$.</td>
<td>CLO₅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### S6
If $L_{61}$ and $L_{62}$ are sets representing Context-Free languages then prove that their union, $L_{61} \cup L_{62}$, is also Context-Free.

### S7
If $L_{71}$ and $L_{72}$ are sets representing Context-Free languages then prove that their concatenation, $L_{71}L_{72}$, is also Context-Free.

### S8
**PART_1.** Build the most powerful computing machine that you can think of. Your machine should be able to process complex languages such as $L_{12} = \{c^n d v a^d v j^n : \text{where } n > 0 \}$. Demonstrate that a string like $c d v a a d v j j$ would be accepted by the machine. You need to build the machine by defining its elements mathematically. You are not required to deliver the machine with hardware components. If you do not use standard notations provided in the textbook or discussed in the class then you need to explain your notations. It is known that Finite State Machines or Finite Automata can accept regular expressions. However, Finite Automata cannot process a language like $L_{12}$, mentioned above, which requires a more powerful machine. You should be able to build such a machine. Explain how your machine will accept strings from $L_{12}$.

**PART_2.** You are asked to complete the following three tasks:

1) In the first step, destroy the HALT state(s) or Final states and their incoming transitions of your machine of the assigned problem of PART_1 and examine the consequences.

2) In the second step, destroy the START state and the associated transitions of your machine (in addition to the destructions mentioned in step 1) and examine the consequences.

3) In the third step, reconstruct the machine so that it is distinct from the original machine of PART_1 (may have one or more additional states and/or transitions) and still process the same language.

### Table

<table>
<thead>
<tr>
<th>S6</th>
<th>If $L_{61}$ and $L_{62}$ are sets representing Context-Free languages then prove that their union, $L_{61} \cup L_{62}$, is also Context-Free.</th>
<th>CLO7, CLO1, CLO2, CLO4</th>
<th>Proofs for union of certain sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>S7</td>
<td>If $L_{71}$ and $L_{72}$ are sets representing Context-Free languages then prove that their concatenation, $L_{71}L_{72}$, is also Context-Free.</td>
<td>CLO7, CLO1, CLO2, CLO4</td>
<td>Proofs for concatenation of certain sets</td>
</tr>
<tr>
<td>S8</td>
<td><strong>PART_1.</strong> Build the most powerful computing machine that you can think of. Your machine should be able to process complex languages such as $L_{12} = {c^n d v a^d v j^n : \text{where } n &gt; 0 }$. Demonstrate that a string like $c d v a a d v j j$ would be accepted by the machine. You need to build the machine by defining its elements mathematically. You are not required to deliver the machine with hardware components. If you do not use standard notations provided in the textbook or discussed in the class then you need to explain your notations. It is known that Finite State Machines or Finite Automata can accept regular expressions. However, Finite Automata cannot process a language like $L_{12}$, mentioned above, which requires a more powerful machine. You should be able to build such a machine. Explain how your machine will accept strings from $L_{12}$. <strong>PART_2.</strong> You are asked to complete the following three tasks: 1) In the first step, destroy the HALT state(s) or Final states and their incoming transitions of your machine of the assigned problem of PART_1 and examine the consequences. 2) In the second step, destroy the START state and the associated transitions of your machine (in addition to the destructions mentioned in step 1) and examine the consequences. 3) In the third step, reconstruct the machine so that it is distinct from the original machine of PART_1 (may have one or more additional states and/or transitions) and still process the same language.</td>
<td>CLO6, CLO1, CLO2</td>
<td>Turing Machines as the class of most powerful processors, construction of Turing Machines for given problems, properties of Turing Machines, Halting Problem, Decidability, Recursively enumerable sets</td>
</tr>
</tbody>
</table>
Students generated all major topics of the course while investigating their problems and the learning outcomes were thoroughly studied. The teacher’s subset of the problems included a Pushdown Automaton for \( L_m = \{ \text{"c"}^n : \text{where } n \geq 0 \} = \{ \text{c}, \{\text{c}\}, \{\{\text{c}\}\}, \{\{\{\text{c}\}\}\}, \ldots \} \) as mentioned above in \( T_5 \). Pushdown Automata are designed to process programming languages such as Java and strings that have similar patterns. That is, a Pushdown Automaton will accept strings like \{c\}, \{\{c\}\}, \{\{\{c\}\}\}, \ldots \. Pushdown Automata use a stack data structure for matching equal number of \{‘s and \}’s without counting them. A stack is an interesting data-structure which allows operations such as push and pop and increases or decreases its stored contents in a Last-In-First-Out (LIFO) manner. Stacks are used for processing context-free languages as described in textbooks (Cohen 1997; Hopcroft, Motwani & Ullman, 2007). One needs to consider multiple ways of presenting automata to students in order to highlight their formal and intuitive relations to other fields such as programming languages. Pushdown Automata can be presented in various ways including state diagrams. In the following example, a Pushdown Automaton for \( L_m \) is presented visually as a finite set of states connected with transitions based on the notations given in (Hopcroft, Motwani & Ullman, 2007) with minor adjustments that show the stack explicitly with the bottom of the stack on the left, and define transitions with the pair: \( R,T/TP \) where R is the symbol read from the
input, $T$ preceding $/$ is the topmost stack symbol before the transition is taken, $TP$ following $/$ is the sequence of topmost stack symbol(s) after the transition is taken and $P$ is an optional symbol which appears only with "push transitions". The state diagrams for Pushdown Automata given in (Hopcroft, Motwani & Ullman, 2007) do not explicitly show the stack.

Ordinarily, static visualizations of Pushdown Automata can be done with a sequence of state diagrams, such as the one given in Figure 3. Suppose a string like $\{\{c\}\}$ is given as an input to the above Pushdown Automaton. Then, the machine starts at the start state and scans the first $\{\$ from the input and pushes a $\{\$ into the stack by taking the transition marked by, $\{\$, $Z_0/\{\$$. The meaning of this transition label is "when reading a $\{\$ and the stack is empty (marked by $Z_0$) push a $\{\$ onto the empty stack (marked by $Z_0$)". Then it consumes the next $\{\$ from the input by taking the same loop with the transition marked by $\{\$, $\{\}$. Next, it consumes the symbol $c$ by taking the transition marked by $c$, $\{\} which means "read a $c$ from the input when there is a $\{\$ on top of the stack and leave the stack unchanged". Next, it reads the fourth symbol, $\}$, from the input and pops a $\}$ from the stack taking the transition marked by $\}$, $\{\}$. Then, it scans the next $\}\$ by taking the same transition marked by $\}$, $\{\} again. Then, it reaches the final state by taking the transition marked $e$, $Z0/Z0$. At that moment the stack is empty and the entire input is consumed and therefore the input $\{\{c\}\}$ is accepted by the machine. The Pushdown Automata given above accepts any string with a sequence of $\{\s followed by a $c$ followed a number of $\}$'s that balances $\{\s. That is, strings such as $c$, $\{c\}$, $\{c\}$, $\{\{c\}\}$, . . . are accepted by the machine. An input is accepted by a Pushdown Automaton if all of the following conditions are met simultaneously: (a) the input is entirely consumed, that is, no other symbols left in the input; (b) the machine is in a final state; (c) the stack is empty.

One type of dynamic visualization of Pushdown Automata is shown in the form of an animation on the following web site: www.asethome.org/pda. This visualization is designed to provide supports, as referenced, to each student at their initial stages of learning Pushdown Automata. The visualization opens with the screen shown in Figure 4 and waits for the user to read and start the animation. Whenever user presses the START_ANIMATION button, the demonstration of processing the input string begins with a sound effect.
This visualization of Pushdown Automata is intended to be a demonstration of processing programming language structures without solving the problems assigned to the students. In addition to Pushdown Automata, visualization of a finite automaton was presented at the following web site: [http://www.asethome.org/fa/](http://www.asethome.org/fa/) (Dey, Sinha, Romney, Amin, Badkoobehi, Subramanya, & Sukhija, 2011). These visualizations provided support to students at their initial stages of learning. In addition to lectures, discussions, brainstorming, proofs, exams and quizzes, some animations were used for demonstrating various problems of mathematical modeling of computation. Animations, such as this often stimulate discussion on computational models, programming languages and parsing.

APDT structure allows the integration of logical aspects of course contents in a wide variety of subject areas (Dey et al. 2009). The most important aspect of mapping the APDT structure to the course is noting the frequency at which the activities are changed. In other words, switching back and forth from different activity boxes of the problems demonstrates agility. This reflects that the planned activities often are changed dynamically in order to accommodate learning events in a flexible or agile way. In the future, data should be collected for measuring the performance of the APDT method. At the time, however, only generic data on standard course evaluation are available for four previous CSC610 course offerings which are presented in Table-2.
Table-2: Course Assessment Data for CSC610

<table>
<thead>
<tr>
<th></th>
<th>December 2008</th>
<th>March 2009</th>
<th>March 2010</th>
<th>September 2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class GPA (4.0 scale)</td>
<td>3.31</td>
<td>3.22</td>
<td>3.22</td>
<td>3.51</td>
</tr>
<tr>
<td>Student Learning (5.0 scale)</td>
<td>4.35</td>
<td>4.86</td>
<td>4.66</td>
<td>4.74</td>
</tr>
<tr>
<td>Teaching (5.0 scale)</td>
<td>4.70</td>
<td>4.81</td>
<td>4.92</td>
<td>4.88</td>
</tr>
<tr>
<td>Course Content (5.0 scale)</td>
<td>4.24</td>
<td>4.75</td>
<td>4.89</td>
<td>4.64</td>
</tr>
<tr>
<td>Number of responding students/Out of total number of students</td>
<td>11/11</td>
<td>4/4</td>
<td>4/4</td>
<td>15/15</td>
</tr>
</tbody>
</table>

The mathematical foundations course, CSC610, was taught by one of the authors using the APDT method in March 2009, March 2010 and September 2011. It was taught using the classic PBL method in December 2008 by the same instructor. From the Table-2 data, no significant inferences can be made at this time, although teaching evaluation suggests slight improvements after adoption of APDT in March 2009. These data in combination with the logical analysis of the APDT structure may suggest that there were no major problems with the mapping of the course to the APDT structure. The students were generally satisfied with course. In future research, more focused data will be collected in an expanded study for measuring students’ satisfaction and performance.

CONCLUDING REMARKS

A structural analysis of APDT presented above reveals its elements and their relationship to each other. The logical structure is easily mapped to a course on mathematical foundations. Although, APDT is closely related to PBL, their significant differences in two major respects are worth noting: (1) APDT places more importance on direct instruction and extensive guidance than the classic PBL method, and (2) APDT is designed to adjust its strategies dynamically, with agility, in order to achieve course learning outcomes. The stage is set in our environment for future experimental studies about APDT performance and course assessment. We believe that the current educational challenges can be best solved by using new pedagogical approaches that focus creatively on the needs of the students and provide adequate support dynamically adjusting the process. Furthermore, we hope that knowledge will triumph, economies will grow, humans will innovate, peace and prosperity will return and the coming generation will thrive in a new learning oriented society.

References


**Acknowledgements**

The authors are extremely grateful to anonymous reviewers of the Journal of Research in Innovative Teaching (JRIT) for their valuable suggestions and comments which led to a thorough revision of the paper. Authors are also thankful to Ronald Uhlig, Ogun Tigli, Alireza Farahani, Albert Cruz and many others for their help and suggestions on earlier versions of this paper and/or related topics.

**About the Authors**

Pradip Peter Dey
Ph.D., Professor, Lead faculty for the MS in Computer Science Program, Department of Computer Science, Information and Media Systems, School of Engineering, Technology, and Media National University, San Diego, California pdey@nu.edu Major research interests: Computational models, mathematical reasoning, visualizations, software engineering, User Interface, education, mobile apps.

Gordon W. Romney
Ph.D., Professor, Department of Computer Science, Information and Media Systems, School of Engineering, Technology, and Media National University, San Diego, California gromney@nu.edu Major Research Interests: Information Assurance and Cyber Security, Information Integrity and Confidentiality, 3D Graphics, Anonymization of Data, Remote Wireless Sensors, Virtualization, Ruby on Rails, and JRuby.

Mohammad Amin
Ph.D., Professor and MSWC Lead, School of Engineering, Technology, and Media National University, San Diego, California mamin@nu.edu Major research interests: wireless communications, database, sensors, engineering education.

Bhaskar Raj Sinha
Ph.D., Associate Professor Department of Computer Science, Information and Media Systems, School of Engineering, Technology, and Media. National University, San Diego, California bsinha@nu.edu Major Research Interests: Computer Architecture, Digital Systems, Information Technology Management, Security, Database, Mobile Apps.