Problem/Project-Based Learning Pedagogy – P^2BL

George Mobus

Institute of Technology

Active Learning

- Deeper student engagement
- Greater actual understanding as measured by assessment
- Longer retention

Social Learning

- Learning is accelerated in social groups
- Students improve their communications and cooperation (soft) skills learning in groups

Higher-order Outcomes

- Learn to learn – become lifelong learners
- Self-directed and motivated discovery (curiosity)
- Problem-solving confidence (empowerment)
- Willingness to take risks – increased expression of creativity

Complete Change

- Eliminate formal lectures
- Minimize reliance on textbooks
- Minimize summative testing to key checkpoints and knowledge areas only
- Allow students to fail and recover (almost everyone should be able to redeem themselves)
- Act as guide, coach, occasional tutor

Problems and Projects

- The hardest part is designing problems and projects that are inherently instructive and provide adequate documentation
- Problems are smaller projects with a clearly defined goal to be reached in a week
- The problems can be stages in an overall project so that students grasp the significance of the objectives (they realize they will need to retain what they learned for later)
- The whole project lasts over the quarter and is very challenging

Projects

- Set up in phases (with problems forming early phases)
- Each phase is designed to demonstrate key principles and skills
- Phases build on prior phases so that the whole is seen as an integrated approach
- The project should be seen to produce something that is actually useful

Example Software Projects for Architecture and OS courses

- Build a working model (simulation) simple RISC computer – learning objective: How a computer computes!
  - 16-bit words, 8 registers in RF, ALU
  - 5-bit opcodes, load & store, 3 addressing modes
  - 64k memory, byte addressable
  - Up to 8 I/O devices, minimal keyboard and monitor
  - Test with simple hand-assembled programs

Problems (Phases)

All done in C

- Build simulated logic gates and simple circuits
- Build simulated full adder (16-bit) using the logic gates from prior problem and simulated registers from D-flip-flops
- Build a four function ALU from the full adder and additional logic (e.g. not (invert), bit-wise and); set condition flags – negative, zero, carry, and overflow; have a simple test interface
- Build a more elaborate ALU supporting more arithmetic functions (e.g. 16 operations); register file, instruction register with decoding, and a simple console-based debug monitor

Project

- Build the complete CPU, main memory, and I/O ports
- Build a menu-driven debug monitor to load and run programs, step through programs, display regions of memory, etc.
- Design a basic assembly language (with data allocation and other pseudo-ops)
- Write a few simple programs that test all of the instructions in the instruction set, hand assemble them and create load files for testing.
- Write user documentation and a project report
- Present the work, highlighting any unique features

Other Projects

- For OS: Build a simulator of a scheduling OS kernel with processes, I/O interrupts, some synchronization services (e.g. semaphores)
- For Algorithms: Build a complete graph modeling library with both adjacency list and array representations, graph traversal methods, etc. Do this as a kind of container class to be used in developing applications (e.g. Dykstra's algorithm for shortest paths to solve a routing problem).

Major Hurdles

- Standard course matrix scheduling is too rigid
- The culture of using grades to “weed out” is still in the background
- Belief that coverage is all-important is inhibiting (Did I teach them everything they should know????)
- Best projects are multi-disciplinary and our degree programs emphasize narrow disciplinary approaches
- Students are so indoctrinated in the teach-to-the-test culture that they are uncomfortable when they are first in this environment (expect low teaching evaluations when teaching first timers)

Cognitive Dissonance with Teachers

- Teachers new to P ^{2}BL generally feel guilty that they are not standing in front of a class lecturing
- Teachers can tend to hover over student groups while they are working – over intervening
- Teachers are NOT the font of all knowledge, but know where to find answers
- Teachers are too used to the idea that some students will get lower grades and they must not be rigorous enough if they don't!

Have trust that students are natural learners and will rise to the occasion if given the chance.