Experiences with Using Assessment Based, Double-Loop Learning to Improve Engineering Student’s Design Skills

Paul J. Fortier, Judith Sims-Knight, Benjamin Viall, University of Massachusetts Dartmouth
285 Old Westport Road
North Dartmouth, Massachusetts 02747-2300
pfortier@umassd.edu

Abstract

The problem with traditional engineering education revolves around the use of stove-pipe curricula, using passive lectures and cookbook laboratories with pre-determined results. Real-world engineering is open-ended and team-oriented requiring active participation.

This paper describes a proof of concept study focused on the efficacy of assessment based double loop learning aimed at improving engineering student’s design skills. Our study began by examining student’s present design skills over all 4 years of the electrical engineering (ELE) and computer engineering (CPE) curriculums. During the second year of the study, junior engineering students are provided with a new open-ended design course where students use design process assessments to improve upon their use of sound design practice, thereby improving their outcomes in design.

1. Introduction

In today’s competitive environment, employers seek engineers who can work within interdisciplinary teams to solve open-ended problems. This requires students become innovative practitioners, lifelong learners with solid design skills. Innovation occurs within the design process, requiring altering how students develop design skills fostering open-ended thinking.

Open-ended design as a process requires engineers who can apply knowledge from prior experience, tools, concepts, theories and general and or specific knowledge to create the desired product, processes or technique. Instructing students in the art of open-ended design has not been part of traditional curricula. Student’s who come to understand that “disciplined creativity” is the purpose of engineering will be proactive engineers better prepared for their professional lives and be better motivated to master the analysis tools necessary for synthesis.

Teaching “disciplined creativity” will require a substantial commitment on the part of faculty to embrace new teaching and learning strategies and a significant effort to develop new practices and course materials. The goal of this curriculum modification effort was to develop, implement, demonstrate and assess a sustainable technology enhanced studio-based open-ended design course for the electrical engineering and computer engineering programs with a philosophy that focuses on educating engineers who are practiced in the art of “disciplined creativity.”

In this paper, the definition of open-ended design is further explored as a framework for disciplined creativity and as a theme upon which we utilize the concept of an artist studio as the foundation for an open-ended engineering design course. We also look to define what types of activities define an open-ended design experience, and explore the techniques for assessing student achievement in such a course.

2. Educational Framework

Evidence is accruing that students are not learning sufficient skills in open-ended design. This is true whether one considers design in a broad or narrow sense. The broad sense, called architectural design, occurs when the problem specification is handed to the designers and they must decide what kind of a system to use. The narrow sense of design occurs after an architectural design is chosen. The designer determines components of the design and proceeds to refine and refactor, until a workable design is created.

Although research has demonstrated that senior undergraduates are better than freshmen in components of problem analysis, an important part of
architectural design [1,2], their skills are minimal [3,4,5]. For example, Eckerdal [3] studied detailed designs of 150 senior computer science students from 21 institutions in four countries and found that 62% of the so-called designs had no design information, and only 3% of the designs could be called complete. Furthermore, those with finished designs had only average grades, suggesting that curricular are not fostering design expertise.

The accrediting body for engineering (ABET), introduced a new accreditation process in 2000 [6], requiring engineering programs institute a continuous improvement process based on assessment of student learning outcomes, including “an ability to design a system, component, or process to meet desired needs”, and “a recognition of the need for, and an ability to engage in, lifelong learning”. Preliminary data on changes in engineering education [7] indicates programs are increasing their emphasis on these outcomes (74 percent of chairs report some change in emphasis on engineering design, and 75 percent on lifelong learning).

Furthermore, faculty increasingly provides active learning opportunities in their courses (50 percent report some to significant increase in use of design projects). In addition, 93 percent of program chairs report some or significant increase in use of student assessment data. Nonetheless, their use of the data for continuous improvement of the curriculum is much lower: only 55 percent reported that half or more of their faculty supported the use of assessment in continuous improvement and only 35 percent of chairs reported they used such data for continuous improvement.

2.1 Why Teaching Design Process is Crucial

In assessment-based continuous improvement curricula, specific student outcomes are identified and assessed, with results used to develop student improvement plans for identified student learning weaknesses. It does not, however, provide a mechanism to target improvement. A manufacturing analogy may be useful. Assessing the diameters of screws produced will yield the required information about the proportion outside desired limits, but it does not reveal how to improve the process to reduce error rates. In the Shewhart/Deming model the loop begins by analyzing the process of making the screws to formulate hypotheses about factors leading to improvement.

Eckerdal [3] assessed the design outcome and found senior undergraduates lacking. Knowing that students perform poorly does not tell how or where they are failing nor does it provide pointers toward improvement. Perhaps this is the reason program chairpersons and faculty find it difficult to use outcome data for improvement. Continuous improvement in the absence of an understanding of the process producing the results limits one to trial-and-error improvement.

2.2 The Design Process

The processes one must understand to enhance instruction of open-ended design are the cognitive processes designers use when designing. Discovering the nature of design process is extremely difficult for several reasons. First, experts’ knowledge is largely intuitive and subconscious, making it difficult to describe. Second, research designed to uncover the processes suggests experienced designers do not all share a common design process. Third, textbooks often present a prescriptive analysis of the design process, sometimes rooted in research on effective practices, but often based on opinion. For example, many textbooks present the waterfall model of design [8], presenting the phases of design in a lockstep fashion. It assumes, for example, that requirements are set, stable, and fully evolved before analysis begins. The reality of engineering design is change is unavoidable and must therefore be explicitly accommodated in the life cycle.

There is now a fairly sizeable body of research in which experienced designers are asked to think aloud while designing. This research resulted in a variety of claims about what designers actually do, but very little analysis of whether what experienced designers do is sound practice. The many examples of massive failures in large software products suggest that standard practice may not be an adequate model of design process on which to base curricular change. A few studies have studied exceptional designers, but a solid picture has not yet emerged. A final problem to address occurs when experienced designers design in familiar areas, their process is quite different from when in unfamiliar areas [9,10].

Our search for a conceptualization of design process to build instructional interventions upon was based on the following principles. First, we argue that student designers are always faced with unfamiliar content and so should learn the design process that works best for expert designers when designing unfamiliar products. In addition expert designers in familiar contexts tend to skip steps that are important to learn. Second, we focused on the practices of extraordinary designers when possible. Third, when relying on research on experienced, but not
necessarily great designers, we selected design principles where research confirmed effectiveness. Fourth, we wanted our principles to include innovative design, occurring primarily during conceptual design.

From this large, diverse body of work, we developed a number of principles of design process skill that seemed both solid and important for students to learn. First, expert designers take time to understand the problem [1,11,12,13]. For example, Sonnentag found that excellent designers, compared to moderate performers, stopped after first reading each line of a requirements document and sketched out related scenarios [13]. Goel [14] found open-ended design task statements (approximately 25% - range 18 to 30%) were problem structuring, and occurred at the beginning, but also occur later as needed. Such practices, including taking time to develop structured requirements, resulted in a 55 percent decrease in defects, a 50 percent reduction in total estimation error due to increased understanding of the features, and a 45 percent decrease in deficiencies in a case study of one software company [15].

The second principle is to consider alternative solutions or maintain flexibility in considering a solution until the problem is well thought out. This is admittedly not always the way designers design. A number of studies show designers tend to latch onto a solution early in the design process and explore the problem in terms of specific solutions. This is particularly true of expert designers with extensive experience in the problem domain they work in [10, 16, 17]. Even when substantial difficulties are encountered, designers tend to make patches rather than changing solutions [17,18]. This trend has also been found with senior students [19]. Such strategies seem to be part of what Simon called “satisficing” strategies, requiring less cognitive space than considering alternatives, is not optimal.

Generating alternative conceptual designs from the beginning [12,20] results in better designs than considering either too few or too many. It can be effective to consider one design at the beginning so long as one is willing to consider alternatives at the first sign of trouble [21]. Outstanding designers tend to use one of these strategies [22,23,24]. In general, they have an ability to maintain openness and to tolerate ambiguity until the situation is fully explored.

A third characteristic of effective design process is early and repeated reviews. Sonnentag [13] found high performing designers “more often evaluated their design solution and started early with these evaluations.” They also were much more likely to solicit feedback from team leaders and coworkers. Other studies found experienced designers are likely to review and reflect on design in terms of requirements [25,26,27]. Research supports this principle. Peer reviews catch over half the defects in requirements, design, and code [28].

Catching defects early is also cost-effective. Shull [28] report finding and fixing a severe software problem after delivery is more expensive than finding and fixing it during requirements and design phase—often 100 times more expensive for severe problems.

Fourth, a principle appearing in prescriptive articles is top-down design, in which designers systematically consider components at a global level, and refine all components to similar levels. Experienced designers use top-down design, but also deviate from it by exploring one component to a detailed level (opportunistic design) [20,27,29,30]. Although opportunistic behaviors sometimes represent a breakdown in top-down design or incomplete exploration of the problem [30], they are an intelligent exploration of design feasibility [29] and way designers provide self-feedback. A related advantage of top-down design is it facilitates the examination of interrelations among components [31].

2.3 The Promise of using the Quality Model to Teach Design Process

Although there have been many projects addressing assessment of ABET a-k [32], only a few projects address continuous improvement at the program level [32,33] and at the course level [34,35,36,37]. To our knowledge there have been no attempts to use continuous improvement based on assessment of student cognitive processes to foster the development of students’ higher-order skills rather than of declarative knowledge.

2.4 How the Quality Model can Help Students Become Reflective Practitioners

In the long run students need to learn to apply the continuous improvement model to their own cognitive growth. They need to be able to reflect upon their own cognitive processes, find ways of objectively assessing those processes, and figure out how to improve. This model of personal lifelong learning was identified decades ago by Argyris [38] as double-loop learning and by Schön as reflective practice [39]. To explain this notion, Argyris used the analogy of a thermostat. Single-loop learning is a traditional thermostat, set at a particular temperature turning the furnace on when the room temperature goes below that value and turns the furnace off when the
temperature goes above it. Double-loop learning would occur if the thermostat were able to analyze whether the thermostat setting was the best way to control the room temperature and change itself if needed.

To be reflective practitioners students must learn to focus on what about learning is working and what is not. The lowest level is active learning, in which students ask themselves whether they understand what they are learning and what they need to do to fill in their gaps. Chi [40,41] discovered several decades ago that good students are able to do this. It is clearly still a challenge to figure out how to help students develop more effective learning processes.

By teaching students to engage in continuous improvement we may help them use the quality model in their learning. We are also interested in whether this intervention promotes acquisition of expertise—motivation for practice, concentration, use of metacognitive processes, and self-efficacy. Ericsson demonstrated that it takes at least ten years of dedicated practice to become high-level experts [42,43]. For example, in one study the best students in a music conservatory spent four hours per day practicing [44]. He also showed that it takes deliberate practice to achieve in college [45]. Only highly motivated individuals will spend ten years deliberately practicing their craft. We wish to discover whether our curricular revisions increase the time students spend in deliberate practice.

We believe one necessary factor for dedicated practice is that design tasks put designers into flow, a state of intense concentration on the task at hand resulting in intense enjoyment and a desire to repeat the experience [46,47]. Thus, we expect as students develop design process skills, it will be easier to enter a state of flow.

Another individual difference construct related to learning is orientation towards learning. Dweck and her colleagues [48,49] have identified two types of orientation—performance and learning. When people approach a task with a learning orientation, they focus on what they can learn from the task and view errors as opportunities to learn. When people approach a task from a performance orientation, they focus on whether they are performing well and view errors as a negative judgment. Button and Mathieu [50] developed a measure appropriate for adults. A related construct has been developed by Biggs [51,52,53] which distinguishes between surface and deep approaches to learning. Those who approach learning with a deep approach strive to understand the material, while those who approach with a shallow approach focus on learning the facts necessary and do not strive to make theoretical connections. We predict that as students learn reflective practice they will become more learning oriented, deep processors.

Metacognitive processing, [54], involves shifting one’s focus from aspects of the problem to how to solve the problem, in other words, one’s own processes. Most undergraduate students do not spontaneously engage in metacognitive processing, but when required, improved their problem solving, both in the original task and in transfer tasks. We have developed a measure of metacognitive processing and predict that having students learn reflective practice will lead to increased scores.

Bandura [55] argues that the feeling of self-efficacy is key to skill improvement and lead to increased persistence on a task. Instructional treatments, in contrast, have relatively small effects. We have developed a self efficacy scale appropriate for engineers [56,57], predict as students develop design process skills, self-efficacy will increase.

In summary, to be effective and innovative designers engineers must be motivated to spend long hours, focused on figuring out the task rather than on performance, be able to step outside the task to monitor processes, and feel confident in their ability to tolerate ambiguity and not foreclose deliberations.

3. Studio-based Open-ended Design

For a continuous improvement strategy to work, courses must focus on developing open-ended design skills. To improve design skills, an open-ended digital design course using concepts from extreme programming (XP) [58] was developed. XP’s pedagogical strategy and its constituent practices such as test-driven development and pair design can have significantly positive effects in academic settings [59].

XP emphasizes teamwork, focused on the target design that is continuously tested and developed in a series of small, fully integrated releases [60]. XP promotes design-oriented learning [61] through (1) individuals and interactions over processes and tools, (2) creating working designs over creating comprehensive documentation, (3) collaboration with customer, keeping the focus on how the product should work, over contract negotiation, (4) responding to change over following a plan [62].

XP practices introduced in academic settings [58, 63] such as the planning game significantly increased the commitment of the students to the project completion and the development process. Erdogmus [64] found that students implementing the “Test-first” XP practice had higher productivity than students who implemented traditional testing. Edwards [65] found
that test-driven development (TDD) resulted in a 28% reduction of defects in students’ programming assignments. Williams and Upchurch [66] reported favorable results from the introduction of the pair programming practice in computer science courses.

The characteristics of design expertise found from experts provide further justification for the use of XP to help students develop skill at open-ended design. Many of these innate design processes can be directly mapped to XP practices. For example, taking time to understand the problem, maps to the XP planning game, testing frequently, maps to XP test driven design, and considering alternative designs, maps to XP refactoring.

### 3.1 Open-ended Studio Course Design

The guide for bringing open-ended design practices using XP into the digital design studio classroom includes examining XP practices and converting them for use in a hardware design course. These XP practices include:

- **Small releases**: structure development into separate periodic releases. After each release students write assessment-based continuous improvement plans to improve their design.
- **Test-driven design**: tests are written before design commences, forcing designers to think on a higher plane.
- **Pair programming**: two designers develop applications, side by side. One is the driver, and the other the observer.
- **Refactoring**: redesigning applications hardware and software abstractions, supporting evolution (response to changing requirements) and improvement (fix design flaws).
- **Planning game**: constitutes an excellent feedback device (reflective experience) where students evaluate design activities in terms of self estimates.
- **Sustainable pace**: teaches students the important concept of time management based on estimates of individual effort.

### 3.2 Studio Classroom Design

The course design focused around the concept of an art studio model, where students are given minimal instruction on a creative task then let loose within a blocked time slot to develop a creative interpretation of an open-ended task. The design course is run as one lecture hour and two integrated three hour studio sessions in a classroom configured for integrated lecture – laboratory sessions. The lecture focuses on providing students with the tools required to develop skills necessary for the sessions open-ended design problem.

Our technology enhanced studio design classroom is organized around individual design, modeling, fabrication and testing stations. Each of these stations is configured for a team of two students to work comfortably. Each design station has relevant software and hardware design tools, simulation, test generation, circuit routing and placement tools. In addition each design team station has AC/DC power supplies, signal generators, wave form generators, digital oscilloscopes, FPGA prototype boards, breadboards, a variety of input and output devices and configurable data collection devices.

### 3.3 Design Course Improvements

To test our hypothesis that the design process can be taught and improved upon, we altered a junior computer engineering digital design course. This course is focused around the concept of open ended problem solving using semester thematic elements. Each element is aimed at providing students with examples of and practice in problem definition, solution discovery, requirements generation and refinement, design considerations and tradeoff analysis, prototype construction, customer interaction, acceptance test development, generation and customer acceptance, through final test and delivery. The course uses Field Programmable Gate Arrays (FPGA) as design components allowing students to experiment quickly with analog, digital and mixed signal designs using state of the art hardware, software and firmware specification, simulation, and synthesis tools. Additionally, since the tools support embedded core processors, students can also experiment with mixed hardware and software implementations for a given design to examine space and performance tradeoffs.

To enhance student design skills the course utilizes a variety of design assessment tools (DPKT, CMAP and Simulations discussed in the next section).

Each individual “design module” within the course consists of a real hardware, software or mixed design (e.g. a controller and data path for a video display screen integrating multiple display planes into a composite video signal), followed by a design assessment generated for the class in total, looking at defining generic design process problems that most students exhibited, and individual team design assessments focused on team design flaws and errors.

This “real” design is followed by a simulation based design generated using a template design tool, that allows for different simulations with variable
flows to be generated. The simulations are designed from the perspective of a design team and its members interacting on a design with customers and upper tier company personnel. The simulations either start with a given initial description (e.g. problem statement, desired product properties, client partial description of requirements, etc.) From the initial statement, either an open ended question is asked (e.g. what would the team do next?) or a set of possible next steps to choose from is given. If the choice is an open ended question, the system will perform a phrase and word analysis to see if they have chosen a correct or plausible path. If the path is possible, they are brought down another path in the simulation that would be derived from the given response. If they provided a statement that was incorrect or not plausible, they would be given some additional details concerning the design or be given a set of alternatives to chose based on direction from a senior engineer or the customer for example.

Using this process of open ended or selective questioning, the simulation flows through the entire simulated design task ultimately leading to a final ending point. Students paths through the simulation are stored as are all open ended answers. Using the traversal through the simulation students design process is “scored” based on the degree of correctness of a given step and the number of steps taken to achieve a give design or partial design. The end result is a directed graph structure that can be categorized (e.g linear, cats cradle, web, etc.) with each having a numeric score per section of the design process (e.g. requirements generation, design tradeoffs, initial design, structure of design, test and acceptance, etc). The generated scores associated with each task in the simulation are used to assess how well students have performed on the given simulation. The results of the simulations along with the results of the matched real hardware design task are discussed with the class in general and specifically with each design team in the next class session after the simulation were performed.

Students are then directed to write a design process self improvement plan using these assessments. The plan is used to provide students with an opportunity to reflect on how they performed in their real design task and in their simulated design task and to describe how they intend to work on improving their performance in future designs. For example, students may find they have trouble in defining the problem sufficiently and succinctly such that they cannot seem to get the design started. To rectify this problem, students may look to get earlier interactions with the target customers to refine the problem statement further and therefore the requirements. These reflections are important elements of the maturation process in developing and improving upon design skills.

We use three distinct designs to provide students with three opportunities to refine their design skills. This is not to say there are only three designs performed during the semester. There are still many early traditional laboratory sequences (3 to 5) where students learn to use design tools and specialized measurement and testing equipment for the course. The three open ended design sequences have been developed in such a way that in the end, each element is used as a component in a larger semester design which is the course theme for the term (e.g. CISC, RISC computers, medical sensors, process control systems, etc.).

For the last term we used three major sequences. The first was a design focused on component development for use in a hierarchical design (sensor data path elements). The second was a multiple plane input and output subsystem using touch screen and composite video generation. The third element was an analog input and digital controlled output that provided the integration of the two top elements into a working.

3.4 Analysis of Student Design Process

A central claim of this paper is that assessing the quality of students’ designs cannot provide the basis for improvement since assessments do not reveal what processes were used or failed to be use and therefore does not provide pointers toward improvement. To provide such design process assessments, we developed the DPKT [67], the structure of knowledge by concept map [68] and relatedness rating tasks, procedural knowledge by simulation tasks [5] and by reflective design task postmortem [69].

The DPKT is a 31-question multiple choice task. It has a Cronbach’s alpha of .89 [67]. It can be scored as a simple sum of correct answers and can also yield a report of areas of lack of understanding (clusters of items) and specific misconceptions through analysis of false alternatives.

In the concept map task students are asked to create a concept map from 10 nouns and 6 verbs using the freeware CMAP Tools from the Institute for Human and Machine Cognition [68]. The resulting concept maps can be scored against an expert map developed through consensus by a team of three engineers (computer, electrical, and mechanical), a computer scientist and a psychologist. The overall pattern can also be classified by a coding system on which reliability overall is 96%.
One of the common patterns is a linear pattern that looks like a waterfall model. Differences between majors in shape of pattern were found. Finally, subsections of the maps can identify particular areas of misunderstanding. For example, in the expert map Feasibility evaluates Requirements and drives or influences Preliminary Design. None of the 166 ELE and computer science seniors got this full pattern although 18% got the link between feasibility and requirements and 28.5% got the link between design and feasibility. Thus, both an overall score and pointers toward improvement can be derived from student concept maps.

The simulation task [69] is a web-based program where respondents advise a design team creating a design. The respondent sees the design documents, starting with a problem statement, and is asked what he thinks the team should do at critical junctures (e. g., how to understand the problem, testing requirements, refactoring). The respondent first gives a narrative answer, and is asked to choose among alternatives the design team has generated. Students who give poor answers to the multiple choice questions are asked further questions until they make an effective choice. A single score can be computed (giving greater weight to students who made an effective choice the first time). In addition pointers toward improvement can be derived from students’ responses to each multiple choice question.

The design task process postmortem [69] asks six questions (e. g., “What design alternatives did you consider?”, “How did you address feasibility?” and “How did you check your design or prototype?”) immediately after students complete a design task. A rubric was developed and proved feasible when applied to the responses of 50 software engineering students, but performance was so low that reliability analysis was meaningless (albeit very high).

The Learning Process Questionnaire [52] and the Goal Orientation Survey [50] assess students’ learning orientation and have been validated in other contexts. The post-design task attitudes survey assesses students’ flow, self-efficacy, and metacognitive processes with respect to a specific task. It has been adapted from versions used to predict performance on the Force Concepts Inventory [56] and developed as part of the Foundation Coalition [57].

Our assessments addressed both what students know about the design process and how students use that knowledge. For each assessment a single score can be calculated and used as a summative assessment for curricular evaluation or as classroom assignments. Each assessment can also be analyzed to give pointers for improvement. For example, the concept map task can identify the proportion of students whose model of the design process is the waterfall. The simulation identifies both the points of the design process where students lack understanding of good process and what their specific misconceptions are.

We demonstrated reliability for the final assessments and established that there were neither ceilings nor floor effects for the final set of assessments.

4. Preliminary Assessment

Assessing the quality of a design is difficult. The traditional criterion in engineering and computer science is whether the system, program or device runs correctly and meets specification. This, however, is often inadequate since minor errors may cause failure of an otherwise excellent design. Likewise a design that runs can be poorly designed and thereby inadequate for maintenance or reuse.

In ASIC and FPGA hardware designs specified using a hardware design language such as VHDL [70], effective metrics have been developed to measure characteristics promoting quality, reuse, completeness, time and space minimization [71,72] implemented using CAD tools.

Particular emphasis will be placed on application of hierarchical system and component designs using concepts and metrics from computational thinking theories [73], design for testability concepts [74,75], grouped generally into a concept called test driven design [76]. In addition to these hardware and software metrics, students’ final designs are assessed by the instructor, computer engineering graduate student and student designer or team independently using a modified twenty-question design assessment scoring tool [77]. Disagreements will be resolved by consensus. Since the final designs are team projects, each member is responsible for a specific subsystem or component within the total design. Both the individual contributions and final team designs are scored.

In the first year of analysis we conducted a summative evaluation of the electrical (ELE) and computer (CE) engineering majors using the DPKT, concept map tasks, and design simulations. In addition to the department wide assessment, students in the four CE specific design courses were given the design assessments (DPKT and additional simulation) at the end of their course. They were also given the design task postmortem at the end of their last design task to assess how well they used their design process knowledge in an actual task.
Initial assessments indicate that students in the studio design course have an improved understanding of the design process and the impact of following good design process compared to not following good practice (Figure 1). Students indicate that they felt more engaged when they were empowered with impacting their own learning. They felt design came alive to them when they were required to think out of the box, be more creative and to draw on their aggregated background knowledge concerning the task. It will have to be seen if seniors who had passed through the redesigned digital design course continue to perform superior to peers.

After expanding open-ended design and assessments into all four years of our Computer Engineering curriculum we hope to be able to test a variety of hypothesis. Some of these are listed below:

- Will students in freshman and sophomore computer engineering design courses centered on teaching open-ended design and assessment-based continuous improvement (1) improve in their ability to design and use effective design process, (2) increase in their meta-cognitive processing, (3) evince more positive attitudes toward engineering and design (4) are retained at higher level compared to traditional curriculums?
- Do students show cumulative sustainable effects?
- Does assessment-based continuous improvement provide instructors with the kind of information that actually results in better courses?
- Do students who initially use metacognitive processes improve their open-ended design skills more than other students?
- Do students who write assessment-based improvement reports become more metacognitive compared to students who do not?
- Do students who show the most improvement in design process skills and metacognitive processes exhibit higher self-efficacy and flow and evince more satisfaction with engineering as a career?

5. Summary and Conclusions

In this paper we presented the process for redesigning a junior level hardware design course around an open-ended studio based design thread using extreme programming and test driven design concepts embedded with self assessment and continuous improvement loops built in. The focus of the redesign activity and assessment is on developing students who have an appreciation for and the tools necessary to be engaged in creative open-ended design throughout their careers.

The redesigned course ran for the first time in the spring semester of 2011. Initial results; though high level indicate that student’s design skills improved beyond those in traditional settings. The initial assessment process also led us to developing numerous other follow on questions we will attempt to answer over the coming years as we expand on this concept and refine our pedagogical and assessment processes.

Acknowledgement

The authors of this paper like to thank the National Science Foundation for their support under grant #DUE-0941233.

References


