E pluribus, plurima: Interdisciplinary class groups yield exceptional results

Debra S. Goldberg
University of Colorado Boulder
430 UCB
Boulder, CO 80309, USA
+1-303-492-7514
debra@colorado.edu

Elizabeth K. White
University of Colorado Boulder
430 UCB
Boulder, CO 80309, USA
+1-303-492-7514
Elizabeth.white@colorado.edu

ABSTRACT
Computer science is increasingly becoming interdisciplinary, with applications not only in scientific disciplines, but also in the arts, humanities, and social sciences. Training computer scientists to work in diverse application disciplines is imperative for modern departments. We have had success using interdisciplinary groups for this purpose in a computational biology class, *Algorithms for Molecular Biology*. In this class, carefully-balanced interdisciplinary groups learn to take advantage of each other's abilities, and to communicate effectively with students with a much different background.

From this diversity, we get much more (*e pluribus, plurima*) than would be possible if we tried to train all students to have a more homogeneous blend of multiple disciplinary knowledge. Within a single semester, students go from virtually no understanding of one discipline to completing research projects on a relevant problem that they have defined themselves.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – [*computer science education]*.

General Terms: Algorithms, Design, Human Factors.

Keywords: Active learning, cooperative learning, project-based learning, communication skills, interdisciplinary education, computational biology.

1. INTRODUCTION
As the field of computer science becomes more interdisciplinary, we must consider how best to educate computer science students for this new reality. There is no single pedagogical method that will work best for all students, so we must have a variety of approaches available to them.

One way to train interdisciplinary computer scientists is to educate them with strong disciplinary knowledge in multiple domains. However, this not only takes more years, which many students are unwilling to invest, but distinct disciplines often require different ways to address problems, which is difficult to master for many students with specific learning strengths.

Rather than trying to cram the equivalent of two degrees-worth of knowledge and experience into a single degree program, an alternative is to train students strongly in one discipline, and give them sufficient knowledge of another discipline as well as the communication skills to allow them to effectively work together with similarly-trained scholars in the other discipline. This is the premise for our *Algorithms for Molecular Biology* course.

There are many reasons to introduce group work into coursework. Employers complain that engineering graduates lack the communication and social skills to function in a team environment, even though this skill is essential to the workplace [1]. Closer to home, a 2004 survey by Waite et al. [27] showed that companies who hire CU graduates perceived significant gaps in these students' preparation for the work force. Specifically, they reported that students had sufficient technical knowledge but lacked the social skills to apply it in groups. More dismally, they found that recent attempts in our department to include project courses have not solved the group problems; if anything, they have resulted in decreased technical skills. Accordingly, interest focused on reshaping the engineering curriculum to better engage and retain these students, without compromising the educational standards to which they are held.

Collaborative learning and active learning models have been proposed by many researchers to alleviate these problems [24][25], but these techniques are no panacea. Faculty members are usually attached to their traditional lecture-based courses [2] and are notoriously reluctant to reformulate them to include more group work, a phenomenon described as the academic version of "not in my backyard (NIMBY)" [12]. Another roadblock is that tenure decisions are generally influenced more by research work than by excellence in teaching [22]. Waite et al. [27] found high resistance to incorporating teamwork into lower-division coursework. Students who are accustomed to a passive learning model are understandably unhappy about learning to navigate a different system, and many are intimidated by the open-ended questions posed by actual research. Group work also presents students with logistical problems, like arranging meeting times [10] and new social dilemmas, like bullying or freeloading [7][20]. Team projects tend to lead to productive behaviors, like brainstorming, but students often have trouble sifting through the wealth of ideas they generate to find the most promising ones [10]. More generally, teams entail reliance on the other group members; student surveys argue that engineering students in particular find this proposition to be onerous [6][5][27]. Waite

makes a good case that this phenomenon is driven largely by ignorance; students tend to overvalue knowledge that they have already mastered, and to underrate concepts that are difficult to understand [27].

We believe that grouping together students with very different disciplinary backgrounds, but are compatible in other respects, may alleviate many of these problems. Students are either better able to or more willing to respect the knowledge that others bring to the group when they are in areas they themselves are not expected to know. We have had success with this approach in a computational biology class, *Algorithms for Molecular Biology*. In this class, students are assigned to carefully-balanced interdisciplinary groups. These groups work together on in-class exercises and homework designed to require the expertise of a biologist, a computer scientist, and a mathematician. In this way, students learn how to take advantage of each other's abilities, and more difficult material can be covered than most students could handle alone. Each interdisciplinary group develops an algorithm for a real biological problem, often based on data from the lab one of the group members works in.

2. THE COURSE

Since 2008, our department has offered a course in *Algorithms for Molecular Biology*. This course has drawn students from over ten departments (Computer Science, Physics, Chemical Engineering, Applied Mathematics, and Molecular Biology, among others). Both undergraduate and graduate students enroll in this course, giving an additional dimension to the diversity in this class, that when managed appropriately, adds to the learning experience. Few students come into these classes with all of the necessary knowledge to understand the algorithms at work; accordingly, the course must fill a variety of gaps. These courses offer an ideal laboratory for cooperative learning, and we have successfully integrated group projects as a central part of the coursework. This paper describes the techniques we have found to be most helpful and summarizes the results of cooperative learning in this interdisciplinary context.

We are mindful of Fenton and Pfleeger's dictum that "you cannot control what you cannot measure" [9]. To this end, we have used the Accreditation Board for Engineering and Technology (ABET) criteria [8] to define learning objectives, select course materials that allow students to meet these goals, and evaluate how well we succeeded and what areas require improvement. The ABET standards detail not only the technical skills students are expected to possess upon graduation, but also the ethical and communicative skills they will need to excel in the workplace [28]. In addition to objectives for understanding basic domain knowledge, this class also has these two objectives, for which the group projects are particularly useful:

- Students possess the skills required to collaborate with biologists, mathematicians, and computer scientists.
- Students have improved communication and presentation skills.

2.1 The Introductory Weeks

The first two weeks of the class are largely introductory material, to bring all students up to a minimum baseline of knowledge in both biology and computer science. Computer science topics include definitions of problems, algorithms, and correctness, pseudocode, asymptotic analysis, tractability (polynomial-time vs exponential-time algorithms), and a variety of algorithm design techniques. Molecular biology topics include a basic understanding of the structure and function of DNA, RNA, and proteins, the central dogma of molecular biology, gene structure, the genetic code, transcription of DNA to RNA, translation of RNA to proteins, gene expression, and the variety of RNA functions. During these weeks, students form ad-hoc groups for active learning exercises. We tell them that research shows that they learn better when they talk in class than when the instructor talks, and that they all have valuable perspectives to share with the class, and gradually through the first weeks there becomes increasing discussion within the classroom. This sets up the value of communication amongst students for their group projects.

Over the next few weeks, some of the most commonly used algorithms in molecular biology are covered. These are the most likely algorithms that will be adapted in group projects, but more importantly, they give concrete examples of abstractions that can be used for biological data, and show how an understanding of both the computer science and the molecular biology contribute towards useful algorithms. In these weeks, homework assignments have students work through and modify the basic algorithms, such as aligning two DNA sequences to compute a similarity score, analyze central phenomena in biology, and use online tools to gain an understanding of how computation is used to gain biological insights. Student groups are encouraged to tackle these assignments collaboratively, but we require them to write up their results independently. This strategy helps teams interact and bond on traditional types of homework problems, and help each other on assignments that require a variety of background knowledge, before they must tackle an open research problem. By week six, the students are required to come up with a research topic, as described below. During the rest of the semester, lectures continue on more advanced algorithms for problems in molecular biology, but homework assignments are largely replaced by milestones for project completion.

2.2 Group Assignment

The first challenge in this course model is to create balanced and diverse groups. Group size is limited to 3 or 4 students, and the groups persist throughout the semester, provided that no pathological group interactions surface. It is critical that each member of the group has an opportunity to contribute uniquely to the work. In forming groups, we take into account the different backgrounds of the students, as well as their maturity. For example, a sophomore in computer science will have some of the required knowledge of code writing, but may need more mentoring to estimate and meet deadlines; such a student may be placed in a group with a graduate student in chemistry, under the assumption that the more experienced member can assist with this process. Similarly, an older student who has already had a successful career in industry can provide the group with valuable grounding in staying on task. On the other hand, students like this can inadvertently dominate the group and intimidate younger or less experienced members; thus, we might counterbalance his group with another graduate student, to help ensure that group work is the result of dialogue between all the members.

Students fill out a student information sheet in the first week. This sheet includes a number of questions that are used to create balanced groups. Factors that strongly determine group placement include the educational level (undergraduate vs graduate students), extent of computer science background, knowledge of molecular biology, current affiliation with an experimental

biology lab, and general areas of primary interest. Each group must have at least one member with a strong CS background and one member with a strong molecular biology background. Each group generally includes at least one undergraduate and one graduate student. To the extent possible, exactly one student is placed in each group who can provide molecular biology data from their research lab.

Based on feedback from students in the first iterations of this class, we have found that a critical component of group composition is that work habits should be relatively homogeneous. A student who starts early, works steadily, and can't tolerate being in a group with last-minute participants should not be in a group with a student who, due to other commitments, cannot commit to completing group work early and steadily. Students are asked to rate their work habits on a scale with four options from one of these extremes to the other, and students within a group will differ by no more than one category.

Although gifted students are not generally considered at-risk, studies have shown they have special needs that are often not met in the classroom[3][4][16][26]. Many high-ability students get little challenge in their schools [4][11] and some are quite discouraged about the prospect of continued academics [21]. Some never learn how to work through a challenge, as the assignments they are given can be done too easily, and many resent being held back by their more typically performing classmates [3]. Many of these students approach collaborative work with dread, as they've done the lion's share of a group's work all too often. Such students need to be grouped together with other high-ability students so that they can truly be allowed to exercise their minds, a privilege for them that most students take for granted [15][19]. Although requiring differentiated assignments, this minority of students must also be taught the value of hard work and to push the limits of their abilities [4][21]. With this research in mind, we added a question that we termed "learning styles." Four choices are given for the question "Which best describes your learning style?":

- I often breeze through classes, and appreciate greater challenge (but not longer tasks).
- I often don't get good grades because I won't jump through unnecessary hoops, but often understand concepts more quickly than others.
- I generally earn good grades by working hard.
- Getting good grades is challenging (e.g., professors/books don't teach to my learning style).

To identify high-ability students, the first two answers, although seemingly very different, are considered equivalent. Although we don't indicate the commonality, these students are grouped together. When these groups meet with the instructor, they are guided towards more complex projects than other groups.

We also ask an open-ended question about what students have liked and/or disliked about previous group work. This can inform us further on the student's work habits, learning styles, and other attributes (e.g., leadership style) that will impact group dynamics.

We have found that the schedule constraints that students report is not a useful factor for forming groups. Students' schedules often change after the first week, and students often have more flexibility than they report. When students in a group share a common interest, have complementary knowledge, and have compatible working and learning styles, they usually manage to find times that they can meet together outside of class.

Groups are formed initially after the first two weeks of the course. These groups are given the opportunity to work together on in-class activities, as well as discuss schedules and common interests for possible group projects. Students are given the opportunity to privately inform the instructor of any perceived conflicts (without need to indicate if it is a schedule, personality, or other type of conflict), and a week later groups are reassigned. Students who have not requested a change do not know if their group composition has changed because someone asked to be removed from their group, removed from another group, or simply because groups needed to be rebalanced after other changes were made. After this, groups rarely change.

2.3 Defining a Research Problem

Many pedagogical studies have shown that students at the college level are adequately exposed to theoretical knowledge, but are unable to connect what they have learned to practical problems in an intuitive way (Hurst 1995). *Algorithms for Molecular Biology* differs from most other classes students have taken in that they get to work on open problems, and not problems with known solutions that can be perceived as toy assignments. While students certainly need to know certain facts, rather than memorizing lots of information, we help them decide which facts are relevant. For example, when aligning two DNA sequences it may be reasonable to consider all "letters" (nucleotides) in the DNA alphabet as being equally different from others, but when aligning two protein sequences, some pairs of amino acid differ much more strongly than others. We strive to focus on higher order thinking skills.

Defining an original problem, without a list of possible choices, is often a daunting task for students. Most have never been expected to do this before. Most groups will include a member who has some type of data that can be used for a group project, and this provides a good focal point to initiate discussion. Groups without a source of data must begin by brainstorming what questions or what type of publicly available data might be of interest.

After student groups have had a week or two to brainstorm amongst themselves, each group meets individually with the instructor. Most groups have not yet sketched out a specific problem to tackle, and the instructor directs a brainstorming session in which each student mentions some interests, and group members start building on the ideas of others in the group. These sessions usually end with each group member taking on something specific to research: a literature search to see what is already known about a particular topic; to learn what has already been done for a particular problem; to find existing software that can be modified; or, occasionally, contacting another researcher about gaining access to their data. Then a second meeting with the instructor is scheduled to complete the identification of a group problem. Generally, a suitable problem can be agreed on quickly at the beginning of the second meeting, if not before.

Some groups have sketched out a suitable problem before their first meeting with the instructor, and proceed immediately to the agenda for the second meeting. At these, students discuss the project's viability and potential pitfalls. Often students are too optimistic about what they can accomplish in the approximately ten weeks remaining in the semester, and the instructor guides them towards reducing the scope of the project. It is important to give groups a high probability of success, and open-ended

problems often take more time than planned as unexpected pitfalls are encountered, so we try to err on the side of smaller scope. However, we define possible extensions that can be worked on if the originally defined project is on track for early completion.

We require that each member have a clearly defined set of independent subtasks to complete, and that these jobs are tailored to the expertise of each student. Fong [10] argues that this approach reduces both freeloading and fear of making mistakes, which can otherwise paralyze a group. If a division of labor can't be found for which each group member feels they can contribute their expertise, and the division seems fair to all, another project must be found. This rarely happens, as by the time the group has agreed upon a project, all group members have considered the project and what their contribution might be.

2.4 From Groups to Teams

From the first days of the project, when student groups choose their problems, we emphasize the importance of implementing a small set of core functions, one at a time, so that their code remains reliable except for the new feature they are adding. We discuss potential pitfalls in these plans with the groups and help them to develop alternative plans, should these arise.

Also from the first days, students must think about how they will test their code, and this must be described in their initial group proposal. Often, a primary job for a group member with a strong CS background is to create a testbed, and a job for a group member with a firm grasp of the relevant biology could be to devise good test cases. The students write tests for their software concurrently with their code, which shows them how test-driven development can detect bugs early on, when there is still time to respond to them. These methods permit students to plan their project according to good software engineering practices.

Given the diversity of backgrounds in the groups, when the students have to prioritize project goals or weigh tradeoffs between them, they learn to pitch their case in language free of jargon, and to respect and engage with collaborators from other areas. When students in a group navigate the process of choosing a project, proposing a way to solve it, dividing programming tasks up between the members, planning deadlines with each other, integrating code from each member into the project, and writing tests to verify the results, not only do they gain a firsthand understanding of the software engineering process, but they also learn how to negotiate with stakeholders from completely different backgrounds to produce a working product.

Through this process, students learn collective responsibility—that others depend on them, and that they can depend on others. No one likes to be perceived as a shirker by their peers, and if they are helped to find a way they can contribute, most students will rise to the challenge. Our challenge is to help them find their way, despite other demands on them. Being the expert on one aspect of an interdisciplinary project, and being able to help others understand concepts that are second nature to you (even while struggling with concepts that others are helping you understand) can be a powerful motivator.

Groups meet with the instructor periodically throughout the remainder of the semester. Problems encountered are discussed, as well as changes to the original plan that result from these problems and/or a better understanding of the problem. These meetings help ensure that problems are dealt with early on, and group directions can be modified before the problems compound.

2.5 Scientific communication

Communication skills are frequently cited by employers as essential skills for success that young employees often need to learn [1]. When groups are first formed, students are given information about working in a group and group project planning. In addition to the informal communication that occurs within groups and during class discussions, *Algorithms for Molecular Biology* gives students experience with and feedback from several different forms of scientific communication.

The first required formal scientific communication is the project proposal, due in week six (of a 15 week semester). Students are given a list of required elements for this document. These include:

- A title describing the project
- The names of the people in the group
- Motivation: An introduction with enough background to explain why something new is needed.
- Problem description: Describe the topic you will research. What question are you trying to answer?
- What's new about your algorithm? After all, this is an algorithms class. You could use a new data encoding scheme (e.g. representing amino acids numerically in a way that makes "similar" amino acids closer numerically), create a new metric (e.g., distance measure), use an existing algorithm not previously used for molecular biology data, or modify an existing algorithm.
- Data: What data you will use, and where you will get it. Describe the input and output. Be specific. For example, instead of "microarray data" you could say "the relative expression level of genes known to be involved in early development at different times during development" (describing which genes will be considered and during which conditions or times they will be measured).
- Approach: Give an overview, then describe each step or task in your approach.
- A list of group members and responsibilities. Examples of some possible individual contributions are:
  - Biological domain expert: works with other members to ensure biological meaning, etc.
  - Background researcher: reviews the literature to find related work, available software, etc.
  - Data collector: finds and acquires data publicly available, from a lab on campus, etc.
  - Data formatter: formats data so it can be used in group's programs, etc.
  - Algorithmic designer: leads development of any mathematical formulae, etc.
  - Algorithmic developer: codes main algorithms, specific routines, etc.
  - Tester: develop test software, determine test data, etc.
- A preliminary literature survey

We have found that students often enter the class with so little experience with scientific writing that they neglect to include a title and/or names of the group members if not explicitly required to do so. Other required elements are discussed during the group's meetings with the instructor, which are completed one to two weeks before the project proposal is due.

Shortly after the project proposals are submitted, groups give a brief presentation of the problem they are addressing (but not the proposed methods) to the class. In doing this, they learn that most jargon is not understood by the entire class, and learn how to phrase things so that everyone can understand.

Around week ten, groups submit a written project update. This document includes an abstract with an overview of the information from the project proposal, assessment plans, changes to the original proposal, accomplishments and plans for each group member, optional or alternative tasks if there is time or if planned tasks cannot be completed, what the group needs from others (e.g., the instructor), and a complete list of references.

Students also present their final results in two venues; this is done first in class, and later at a poster session, to which science and engineering faculty are invited. This provides the students with a rare opportunity to practice communicating scientific ideas to other experts in their field.

Finally, students must submit a paper in a format accepted by a scientific journal. Initially, papers were not required, but this requirement was added in an attempt to encourage students to publish their results (when reasonable). Formats accepted by PLoS ONE are encouraged, but not required. PLoS ONE is a journal that will publish any paper that is scientifically sound, without the authors needing to convince reviewers of the significance of the results. This format has many advantages, and many significant results have been published therein, but it is also a plausible publishing venue for many group projects. Although we consider at least half of the group projects to have produced results that would be publishable with little or no additional work, and have offered assistance to any group members that want to pursue this, none have yet taken that final step to get the work published.

2.6 Grading

At the end of the course, the students in a group each evaluate the contributions of each member, including themselves. Studies have shown that students in teams resent being given one grade for the entire team, because they perceive it as giving freeloaders an unfair advantage [14]. Each student's grade for the group's products are modified by the feedback from these evaluations. To further validate these evaluations, each group is required to use a shared workspace (e.g., a wiki, blog, or google doc) that tracks modification history by author, and allow access to the instructor. The instructor looks through these periodically to ensure that all group members are contributing appropriately to the project.

2.7 Feedback

The course is now in its sixth iteration, and we have solicited student feedback via an anonymous survey and made changes each year. For instance, we have made homework assignments smaller and more frequent, to provide a more consistent workload. Group forming proceeds by a more sophisticated algorithm that takes student maturity into account as well as student background. The instructor now meets with each group more often to assess how the project is going, which provides more timely resolution of problems than waiting for students to recognize group issues on their own. We have also put more effort into reaching out to other faculty at CU to improve lecture content and assignments, to present guest lectures, and to supply groups with raw data for analysis. Last year we had a consultation by an associate for our campus teaching excellence program interview students in the absence of the instructor and provide valuable feedback on the strengths and weaknesses of the course.

While formal evaluations have been relatively low, as expected in project-based learning classes, many students have gone out of their way (even long after grades are due) to remark on the value of the class. Last year's evaluation report indicated that the greatest strength of the class was "good group composition (mixed bio & cs students, undergrad and graduate students", to which all 8 of the students interviewed agreed. Clearly the group portion of this course is working.

3. CONCLUSION/DISCUSSION

It is always difficult to develop a new course from scratch, particularly one that draws on as many disciplines as these classes do. Teaching these courses with a focus on collaboration presents additional risks: closer supervision of student groups is required to prevent students from dominating groups or hitchhiking with more active group members. Most importantly, student resistance to depending on their peers must be overcome. Even with these caveats, however, we see great promise in this approach. Over the past six years, the course has generated ongoing research collaborations between students from biology and computer science, and we have received many positive comments from students about the value of these courses over courses taught according to more traditional models. For example, the following unsolicited note of thanks was sent to the instructor well over a year after the student took the class:

> "…of all the classes I took as an undergraduate at CU, yours has proven to be among the most useful. While I was taking the class, I didn't really have much perspective on how useful it would be. For example, having never participated in de novo genome-sequence assembly or resequencing, I was only vaguely aware of what these fields were. So it didn't mean as much to learn the limitations of scaffold-assembly algorithms. As you well know, though, all of molecular biology today is inseparable from computer science and having delved deeper into the discipline, it amazes me when those of my colleagues who have only extensive biological background take the output of a computer program at face value—taking for granted that it is correct and relevant without much awareness of what the computer science means biologically. I've seen this for generating sequence alignments for microbial ecology, for RNA seq [sic] data, molecular phylogenetic, genomics, etc. I'm a long ways off in understanding much or any of the programming from the perspective of a computer scientist, but I can tell already that the perspective I garnered from your class has helped me as a biologist."

The rules we use to form and instruct groups, which include both experienced and less experienced members, yield unexpectedly productive collaborations. We foster a classroom culture in which all members contribute, and we advise (but do not manage) groups to help them circumvent problems. This model for interdisciplinary work allows our students to experience the pleasure of being local experts, and it brings out the best in them over the semester. Compared to cross-training students to expertise in two or more fields, this approach is nimble. We argue that this method is ideal for modeling genuine scientific collaboration at both undergraduate and graduate levels. As

collaborations involving computer scientists are becoming prominent in a wide variety of fields, we see interdisciplinary classes like *Algorithms for Molecular Biology* as a great way to prepare students to work happily and productively in this kind of future.

4. ACKNOWLEDGMENTS

Our thanks to our college and department for providing funding for initial course implementation, to Tim Dunn for his assistance with the class in recent years, to our colleagues for many fruitful discussions, and to all the students who have taken this course and thereby helping others learn and make the course a success.

5. REFERENCES

[1] Adams, S. G. and Laksumange, B. Building successful student teams in the engineering classroom. Journal of Science, Mathematics, Engineering, and Technology. 4 (3/4), 2003.

[2] Basken, P. (2009). Engineering Schools Prove Slow to Change. Chronicle of Higher Education, 55(21): A4, January 2009.

[3] Colangelo, N., S. Assouline, et al. (2004). A Nation Deceived: How Schools Hold Back America's Brightest Students, The Connie Belin & Jacqueline N. Blank International Center for Gifted Education and Talent Development, University of Iowa.

[4] Davidson, J., B. Davidson, et al. (2005). Genius Denied: How to Stop Wasting Our Brightest Young Minds, Simon & Schuster, New York, NY.

[5] Diwan, A.; Waite, W. M.; and Jackson, M. H., (2002). An infrastructure for teaching skills for group decision making and problem solving in programming projects. SIGCSE '02, February 27-March 3, 2002, Covington, Kentucky, USA, 276-280.

[6] Drury, H.; Kay, J. and Losberg, W. (2003) Student satisfaction with groupwork in undergraduate computer science: do things get better? In Proc. Fifth Australasian Computing Education Conference (ACE2003), Adelaide, Australia. CRPIT, 20. Greening, T. and Lister, R., Eds. ACS. 77-85.

[7] Felder, R. M. and Brent, R. (1996) Navigating the bumpy road to student-centered instruction. College Teaching 44 (2): 43-47.

[8] Felder, R. M. and Brent, R. (2003) Designing and teaching courses to satisfy the ABET engineering criteria. Journal of Engineering Education 92 (1): 7-25.

[9] Fenton, N. E. and Pfleeger, S. L. Software Metrics: A Rigorous and Practical Approach (2nd ed.), 1997, PWS Publishing Company, Boston, MA.

[10] Fong, P. (2010) Journal of Professional Issues in Education and Practice. 121-127.

[11] Gilman, B. (2008). Challenging Highly Gifted Learners. Prufrock Press, Waco, TX.

[12] Goldberg, D. E.; Cangellaris, A.; Loui, M.; Price, R.; and Litchfield, B. 2008. iFoundry: Engineering curriculum reform without tears. ASEE National Conference and Exposition.

[13] Hurst, K. D., (1995). Proceedings of the Frontiers in Education Conference, 1995, 1-4 November 1995, Atlanta, GA, vol.1, 2b3.8-2b310.

[14] Lejk, M. and Wyvill, M. (1997) Group learning and group assessment on undergraduate computing courses in higher education in the UK: Results of a survey. Assess. Eval. High. Educ. 22(1) 81-93.

[15] National Association for Gifted Children. (2009).Grouping [Position Paper]. Washington, DC. Retrieved from www.nagc.org.

[16] Purcell, J.H., (1994). The status of programs for high ability students. The National Research Center on the Gifted and Talented, University of Connecticut: Storrs, CT.

[17] Rimm, S.B. (2008). Why bright kids get poor grades and what you can do about it. Great Potential Press, Scottsdale, AZ.

[18] Rodger, S. H., (1995). An interactive lecture approach to teaching computer science. ACM SIGCSE Bulletin, 27(1): 278 - 282.

[19] Rogers, K. B. (1998). Using current research to make "good" decisions about grouping. NASSP Bulletin, 82(595), 38-46.

[20] Rosser, Sue V. (1998) Group work in science, engineering, and mathematics. College Teaching 46(3): 82-88.

[21] Ruf, D.L. (2005). Losing Our Minds: Gifted Children Left Behind. Great Potential Press, Scottsdale, AZ.

[22] Schertz, B. K.; Goldberg, D. E.; and Hyman, K. K. (2010) Engineering education reform in a large research university: Strategies and reflections on innovation. In Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments, April 6-9, 2010, Dublin, Ireland.

[23] Splitt, F. G. (2003). The Challenge to Change: On Realizing the New Paradigm for Engineering Education. Journal of Engineering Education, 92(2): 181-187.

[24] Springer, L.; Stanne, M. E.; and Donovan, S. S. (1999) Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis. Review of Educational Research 69 (1): 21-51.

[25] Terenzini, P. T.; Cabrera, A. F.; Colbeck, C. L.; Parente, J. M.; and Bjorklund, S. A. (2001) Collaborative learning vs. lecture/discussion: Students' reported learning gains. Journal of Engineering Education 90(1): 123-130.

[26] U.S. Department of Education. National excellence: A case for developing America's talent. 1993, Office of Educational Research and Improvement: Washington, DC.

[27] Waite, W. M.; Jackson, M. H.; Diwan, A.; and Leonardi, P. M., (2004). Student culture vs. group work in computer science. In Proceedings of the 35th ACM Technical Symposium for Computer Science Education (pp. 12-16). New York, NY.

[28] Williams, J. M. (2000). Transformations in Technical Communication Pedagogy: Engineering, Writing, and the ABET Engineering Criteria 2000. Technical Communication Quarterly, 10(2): 149-167.