Effectively Assign Student Groups by Applying Multiple User-prioritized Academic and Demographic Factors Using a New Open Source Program, GroupEng

Thomas G. Dimiduk, Kathryn C. Dimiduk
Harvard University/Cornell University

Keywords: groups, demographics, academics, multi-factor, open-source, isolation, teamwork, balanced groups, automation

Conference Summary

We created an open-source program, GroupEng, which assigns groups according to guidelines from education research. Guidelines include avoiding isolating women or minorities and assigning multi-disciplinary groups of mixed abilities. The program operates on a set of simple, flexible, faculty defined rules, keeps data local, and ensures "fairness" of group strengths.

Abstract

Isolation of women and minority students increases their likelihood of dropping out of engineering[1,2,3]. Groupwork can either aggravate or reduce their isolation. The literature recommends [1,2] clustering two or more of these students per group rather than the common practice of spreading them out evenly and singly. Faculty must also meet other pedagogical and course specific goals in selecting groups. Furthermore, large variability in groups' strength invites wide-spread student complaints. Several orthogonal grouping rules can be addressed simultaneously, but this highly constrained problem is very time consuming to solve by hand for large classes. We have created an open-source program, GroupEng, which assigns groups from a class spreadsheet according to a prioritized list of rules chosen by the instructor. GroupEng differs from existing web-based programs[4-9] by keeping student data local, increased emphasis on fair groups, and permitting user-defined rules. The instructor defines group size, then specifies and prioritizes group selection rules. Rules have one of four functions: balance expected group performance, distribute students with particular attributes across groups, cluster 2 or more students with a particular attribute, and aggregate students by an attribute to make homogenous groups. For example, one can disperse majors in an interdisciplinary class, freshmen in a multi-age class, or students with skills from a particular prior course. Isolation of women and/or minorities can be avoided. Students with similar interests can be grouped together. Using student performance data (eg. pre-test or early test scores, pre-requisite courses, homework scores, GPA, …), GroupEng can create "fair", mixed ability groups. Multiple rules can be prioritized and applied; if the problem is over-constrained, the program meets the highest priority constraints first and then lower priority ones where possible. The program returns a set of groups that are a good fit to the requirements, comparable to one produced by hand with many hours of effort. Freed from the mechanics of group selection, faculty can take advantage of multiple research-based recommendations to form groups and tailor them to their particular institution and instructional style.

Introduction

Group work is an important part of engineering education; indeed use of group work to develop students' skills in working on multi-disciplinary teams is an ABET requirement[10]. Details of how these groups are assigned or chosen can have a significant effect on student learning and success. Isolation of women and minority students increases their likelihood of dropping out of engineering[1,2,3]. Group work can either aggravate or reduce isolation for women and minority students[1,2]. In addition, the self-selection of groups can lead to students feeling isolated or awkward[11]. There is considerable literature covering recommendations for forming groups; this literature is surveyed from an engineering perspective by Layton et al.[12] In particular, the literature recommends assigning groups and clustering two or more women or URM students per group rather than the common practice of spreading these students out evenly and thus singly[1,2]. Mohanty[13] and Tuitt[14] discuss the need for creating an atmosphere of social trust as part of creating an atmosphere conducive to learning[13]. Both cite experiments by Steele[15] that show social trust or mistrust can affect minority student achievement. Thus, a set of "ideal" groups should avoid isolating women students, avoid isolating minority students, be multi-disciplinary, and help develop social trust in the classroom setting. Groups that students perceive as unfair can hurt social trust while balanced groups can enhance students' social trust. In the context of group assignments, balanced, "fair" groups have an equivalently valuable mix of skills and academic strengths. In addition to these common goals, individual faculty frequently have specific pedagogical or logistical goals they wish to consider. It is usually possible to meet many such constraints simultaneously because the variables under consideration are largely independent. However, this problem is very time consuming to solve by hand due to the large number of constraints. This problem is, however, amenable to automation. A fast computer program would allow faculty to explore various combinations and priorities of grouping rules to choose a good set of groups for their course.

There are several existing programs that assign groups. Several simple programs available on-line assign groups randomly[4-6], but they are don't meet the educational goals as described above. There is one existing program, TEAM-MAKER [7-9], that meets many of these goals. This program was first created as an open-source project at Rose-Hulman University [7,8] ,but this version is no longer supported and has been replaced by CATME/TEAM-MAKER[9]. The CATME project hosts the new CATME/TEAM-MAKER program which is free to use and has a clear, easy to use web-based interface, but the source code is no longer available. All further references in this paper to TEAM-MAKER refer to the active CATME/TEAM-MAKER. TEAM-MAKER uses a multi-start greedy algorithm and is described in more detail in several papers[12,16,17] . It includes the opportunity to incorporate an automatic student survey to collect data. However, TEAM-MAKER does not meet our needs for several reasons. First, sensitive FERPA protected data and survey data are sent to an external website, and it may not be clear to students completing a TEAM-MAKER survey that the data, while confidential from the outside world, is released to their professor. Second, instructors cannot create criteria specific to their course. Third, in our understanding, the TEAM-MAKER algorithm for group value contains a

presumably unintended consequence of allowing groups that meet demographic or other rules to be weaker on average strength than other groups. We compared TEAM-MAKER selected groups with the set of groups assigned by hand for an 87 student class; the standard deviation of total group strength, as determined by GPAs, was significantly larger for TEAM-Maker selected groups (with maximum weighting for group "fairness") as for those assigned by hand.

To address these concerns, we have implemented a new program for assigning groups which we have named GroupEng. GroupEng contains a flexible grouping algorithm that can enforce very strong fairness constraints. We have open sourced GroupEng so that others can use and improve it. GroupEng is available from www.GroupEng.org[18] under the Affero General Public License version 3 (AGPLv3).

We also introduce a set of four generic operators (distribute, cluster, aggregate and balance) for describing goals in group formation. These operators, acting on different types of student data, can describe all of the group selection criteria we have encountered through the Cornell Engineering Teaching Excellence Institute and our own teaching experience, so we believe they are quite general.

Description of the GroupEng Program

GoupEng is an open-source (AGPLv3) program written in Python that assigns groups to meet an arbitrary set of instructor specified and prioritized grouping rules. GroupEng is designed to be run locally by an instructor or other university official and thus keeps all student data local. The program is customizable on several levels. The grouping rules are generic and modular, allowing instructors to determine groups based on any factors they desire. Further, the source code is available and is organized to facilitate adding entirely new grouping rules. Instructors supply data about their students to GroupEng as a .csv text file (usually saved from an excel file). This data can come from class data, institution data or from a student survey. Instructors specify group size and create a set of rules or criteria that they want GroupEng to use in assigning students to groups. A rule is defined by one of 4 operations: distribute, cluster, aggregate, and balance and some student attribute on which it acts: e.g. gender, race, major, or gpa. Each of these operations is defined below.

Distribute: Spread students with a particular attribute out across groups. For example, distribute can form multi-disciplinary groups by distributing out students by major.

Cluster: Ensure that students with this attribute are in a group with at least one other person with the same attribute. For example, cluster avoids isolating students with a particular attribute (e.g. minority) in a group.

Aggregate: Form groups whose members are alike in a particular property. This forms homogeneous groups. For example, aggregating on recitation section would allow groups to work together during recitation time.

Balance: Create "fair" groups. For example, balance forms groups with similar total strength based on a measure of each student's academic strength such as GPA.

(Note: Balance is applied to an attribute with a numeric value, either a continuous variable such as GPA or to a discrete variable such as a multiple choice survey question on skill level. Distribute, cluster and aggregate are all applied to attributes that have discrete, values which do not need to be numeric.)

Table 1 lists some sample rules and Fig. 1 shows a schematic of group selection using GroupEng. Because GroupEng accepts arbitrarily many rules, the group assignment problem can become over constrained. In this case GroupEng will meet high priority rules, and then lower priority rules to the greatest extent possible without violating the higher priority rules.

Table 1. Sample grouping rules

Sample rule	**Operation**	**Student Attribute**
Make groups interdisciplinary	Distribute	Major
Spread out students by year		Year in school
Each group has the necessary background		Prerequisite skills or courses
Spread out students with weak English		English proficiency
Each group has a self-identified leader, writer, and content specialist		Self-identified contribution to previous groups
Separate certain students		Common flag for these students #
Don't isolate women	Cluster	Gender
Don't isolate URMs		Ethnicity
Keep disabled student with note-taker		Flag these students, #
Group by project choice	Aggregate	Project choice
Group students by major		Major
Group students by recitation section		Recitation section
Group grad and ugrad separately		Grad or ugrad status
Group students by how much effort they want to put into the project		Survey data on expected effort^
Balance academic strength of groups	Balance	GPA
Make groups fair based on prior skills or knowledge		Pre-test score
Make groups fair based on how students are performing in the class		Test 1 scores
Make groups fair based on prior skills		Survey data on skill level *
Create fair mixed ability groups		GPA, test score or pre-test score

^ Survey must contain several specific choices (not a fill in the blank). # different flag for different sets of students

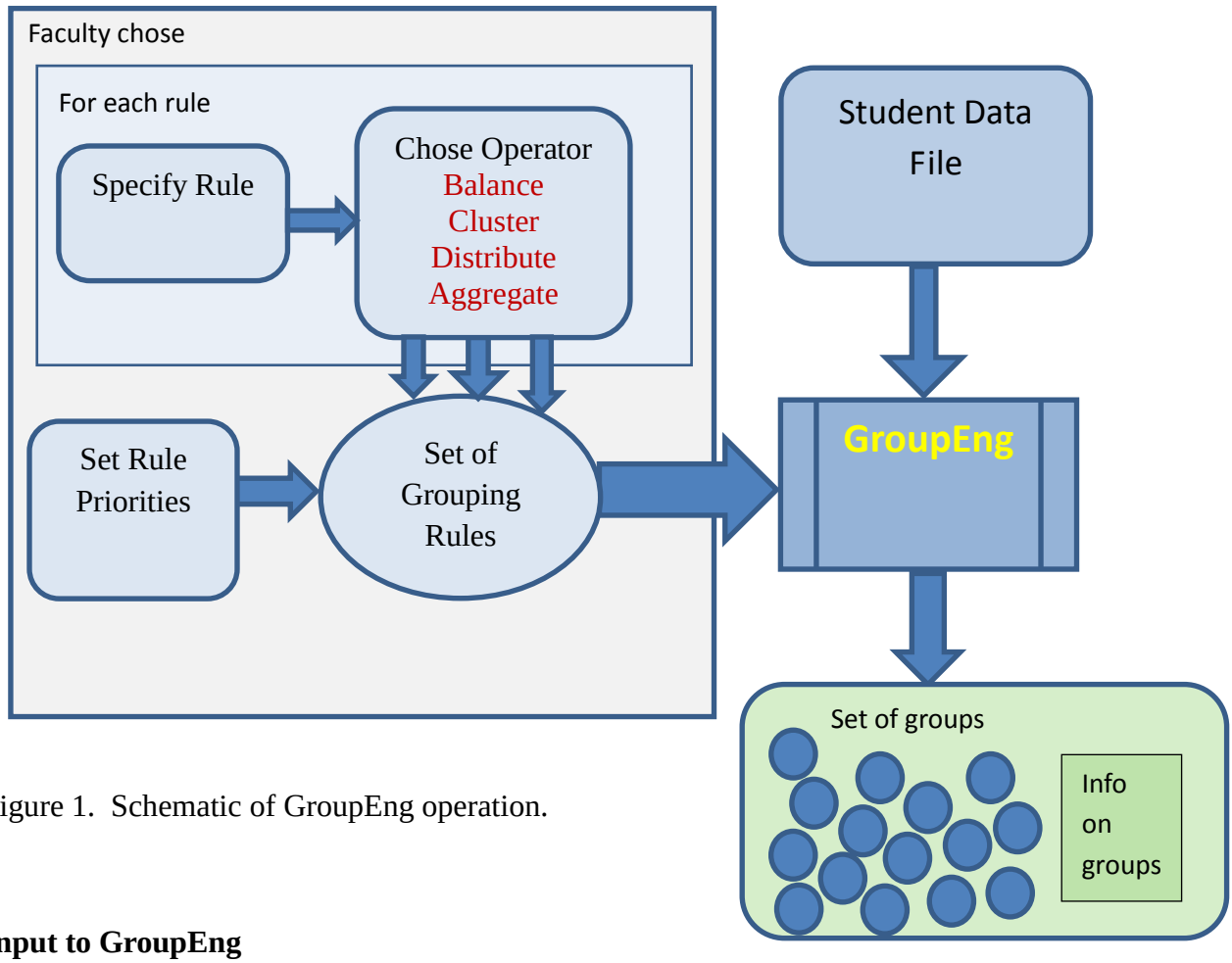

Figure 1. Schematic of GroupEng operation.

Input to GroupEng

There are two types of input to GroupEng: student data and grouping rules. Student data is supplied as a .csv text file. Each student is represented by one line/row of the file. Columns contain attributes on which the grouping rules operate. The first row of the file is a header and contains labels for each attribute. The grouping rules and desired group size are supplied to GroupEng in an input deck (see Fig. 2). This file is a simple structured text file (yaml[19] format). Group size must include how to handle the extra students when the class size does not divide evenly by group size; this can be to either allow a few groups to be high or low one member. Each rule contains the data label, such as gender and an operator such as cluster. Rules are prioritized by their order of appearance in the file (first highest). A future version of GroupEng may allow specification of numeric weights. Even high priority rules are subject to the reality of how students can be divided up; for example, 17 students cannot be perfectly aggregated into groups of 3. There can be as many or as few rules as desired. Keep in mind that a large number of rules can easily over constrain the group assignment and leave the program little ability to meet the low priority rules. More detailed user instructions for writing these files are available at GroupEng.org.

```
# The file that contains the student data.
classlist : mae3240.csv
# Identifier (name, id number, email, ...)
identifier : Name
# Student strength, e.g. gpa, grade in prereq/test, ...
strength : GPA
group_size : 3
# choose some groups low or high if class not evenly divisible by group size
uneven_size : low
rules:
- type : balance
  tol : .2 # standard deviations
- type : cluster
  flag : Gender
  value : F
# What kinds of output do we want?
output :
- type : full_report
  outfile : mae3240_grouped.csv
- type : group_blocks
  outfile : post_me.txt
```

Figure 2. Sample input deck. # indicates a comment (red). Line 2 specifies the data file. Lines 6-9 specify the group size and preference for when this cannot be exactly met. Lines 11-15 specify grouping criteria by grouping operator and column header for the data on which it operates, and lines 17-21 specify output files and format.

Algorithm

The GroupEng results presented here use a heuristic guided stochastic greedy algorithm. This algorithm is simple, fast, and easily extensible. It iterates through rules in priority order fixing "breaks" by swapping students between groups. The following criteria are used to determine if groups "break" a rule:

Cluster: broken by groups which contain exactly 1 student with the given value.

Aggregate: broken by groups which mix values for the given attribute. Usually 1 group will break this rule for each value under consideration because the number of students is not evenly divisible by the group size. This is considered normal operation.

Distribute: broken by groups which have more or less than their "share" of members with any attribute value under consideration. This "share" is (# students with attribute value)/(# groups), or the two integers on either side of this value for fractions.

Balance: broken by groups for which the standard deviation of group member's values for the attribute is larger than a tolerance times the class's standard deviation for that attribute: StDev(group) > tol*StDev(students)

Once a high priority rule is finished, swaps which would break that rule are not allowed while processing lower priority rules. Uneven group sizes are handled by filling out the smaller groups with placeholder students which are considered to have the lowest value in the class when computing deviations for balance rules and otherwise ignored. This causes the desired effect that groups short a student have a slightly higher average strength. See www.GroupEng.org and the GroupEng source code for further detail on the algorithms.

In assigning groups by hand we used excel with a nested sort to list students by a values for several attributes. We then started assigning groups, usually starting with a cluster rule and carefully picking students with high, medium and low GPAs for groups of 3. After each group was formed we filtered those students out of the pool, and repeated. Though still very time consuming, this approach made it possible to consider 4-5 criteria.

Output

GroupEng creates a set of groups based on the group size, prioritized grouping rules and student data supplied. The program output includes a list for posting that only includes group member information (by name or other ID), and a .csv file which includes of the input data and with their group numbers added to each student's row. GroupEng also provides a report that gives information on the overall set of groups (average group strength and standard deviation of strength) and on the individual groups (group strength and specifically which, if any, of the rules the group fails to meet). Strength of groups is measured as both average group strength, which is not sensitive to group size, and as total group strength, which is sensitive to group size.

Results

We have tested GroupEng on classes for which we had previously assigned groups by hand. For one of these classes we also anonymized the student data and then ran it through the CATME/TEAM-MAKER program. Table 2 gives the grouping criteria for example classes and compares how the set of groups produced by each approach met the grouping criteria. The table gives class size, group size and whether a minimum number of larger (high) or smaller (low) groups were used if the class size didn't divide evenly by the group size. In each, we applied a balance rule and lightly tuned the tolerance parameter to achieve good strength statistics. The average GPA for each group was calculated and then group strength statistics (average, standard deviation and range) were determined for the set of group averages. The slight differences in the average of the averages come from the difference in weighting of individual GPAs in the small number of large or small groups. For each rule besides balancing groups, we calculated what percentage of the groups met the rule.

Table 2. Comparisons of how groups formed by GroupEng, by hand and, by Team-Maker meet various grouping rules.

Class Class Size (group size)	Assigned by	Group Strength Statistics			% of Teams Meeting Grouping Rule			
		1st Rule Balance GPA			2nd rule	3rd rule	4th rule	5th rule
A 88(3, high)		Ave	StDev	Range	Disperse major	Disperse year	Cluster gender	Cluster URM
	By hand	3.22	0.069	3.05-3.32	69%	76%	83%	72%
	GroupEng	3.21	0.06	3.11-3.27-3.48	100%	100%	100%	100%
	TeamMaker Run 1	3.21	0.200	2.69-3.56	*	*	100%	*
	TeamMaker Run 2	3.22	0.217	2.82-3.61	*	*	100%	*

B 110			Ave	StDev	Range	Cluster gender			
	(3, low)	By hand	3.23	0.069	3.075-3.43	81%			
		GroupEng	3.24	0.02	3.16-3.26	100%			
	(3, high)	By hand	3.23	0.075	3.075-3.35	81%			
		GroupEng	3.24	0.02	3.20-3.27	100%			

C 114		Ave	StDev	Range	Disperse Freshmen	Disperse Women[o]	Aggregate majors	Disperse language
	By hand	3.12	0.133	2.57-3.25	100%	100%	76%-87%^	95%[#]
	GroupEng	3.11	0.03	3.07-3.18	100%	100%	100%	

*TEAM-MAKER cannot address rules except GPA and gender data from a student survey.

^missing data on some students, for hand filtering at least 76% meet the rule and up to 87% could meet rule, in GroupEng students with missing data don't count towards fails

[#]: disperse language was done with manual guesses, GroupEng can disperse langage if given that data

[o] instructor specifically requested dispersing women

We have found that 4-5 orthogonal criteria is a practical maximum for hand sorting even requiring only ~75% rule adherence. Hand sorting created balanced groups that were a significant improvement over random grouping and is the standard we set out to exceed with GroupEng. Because GroupEng uses a stochastic algorithm, there is some variation between runs, See Table 3 for the results of 10 consecutive runs with the same rules and settings on Class A. The best result is highlighted and is the result shown in Table 2. Each run took less than 1 minute on a 2.5 GHz Pentium® Dual-Core E5300 with 4 GB of RAM. TEAM-MAKER run 1 had "not isolating by gender" weighted at 3 out of 5 and group dissimilar GPAs weighted at 4 out of 5. For run 2 gender was weighted 2 and disperse by GPA at the maximum value 5.

Table 3. Data from 10 runs of GroupEng on class A. The best run is highlighted. Note that several runs are nearly as good.

Run	StDev of group strength	Disperse Major	Disperse Year	Cluster Gender	Cluster Race
1	0.06	97%	100%	97%	97%
2	0.06	100%	100%	100%	100%
3	0.07	93%	100%	97%	97%
4	0.08	97%	100%	97%	97%
5	0.14	100%	100%	100%	100%
6	0.08	100%	100%	100%	100%
7	0.16	97%	100%	100%	100%
8	0.08	97%	100%	97%	100%
9	0.09	93%	100%	100%	97%
10	0.06	97%	100%	100%	97%

We compared sets of groups using the standard deviation of group strength to measure balance (fairness) and the percent of the groups that met the other rules. GroupEng results (using the best run of up to 10 runs) were comparable to hand sorting results for fairness and considerably better for adherence to other rules. TEAM-MAKER groups were significantly less balanced (fair), but did as well as GroupEng and better than hand sorting in meeting the gender rule. We were not able to assess TEAM-MAKER adherence to the other rules without creating a survey and having students to input the data.

Discussion

We have defined four operators (balance, cluster, distribute, and aggregate) that can be used to define most grouping criteria. We have created a program, GroupEng, which applies these operators to create balanced groups that also meet several additional goals. Testing of GroupEng on real class data shows that GroupEng can create fair groups that meet several additional rules. How well additional rules are met depends on the overall constraints of the rules and student attributes. Table 4 compares the range in average GPAs for the groups created

by each method using a GPA scale of A = 4.0 with ±0.3 for a + or – increment on a grade. Overall, GroupEng does a better job of forming groups than hand sorting, with for drastically less effort. GroupEng and Team-Maker both meet other rules well, but GroupEng does significantly better at using GPA data to form balanced groups. Setting up a GroupEng run takes only a few minutes and the program runs in less than a minute, while even with practice hand sorting groups many hours for a large class.

Table 4. Range of average group strengths in terms of letter grades for Class A

Approach	group strengths	in grade increments (such as B to B^+)	As letter
By hand	3.05-3.32	0.9 increment	B to B+
GroupEng	3.11 – 3.27	0.53 increment	Mid B to nearly B+
Team-Maker run 1	2.69-3.56	2.6 increments = 0.87 full grade	B^- to halfway between B^+ and A^-
Team-Maker run 2	2.82-3.61	2.3 increments = 0.79 full grade	Mid B^- to nearly A^-

Since the program can quickly assign groups according to rules and priorities, faculty can experiment with several variations of rules and priorities to find a set that works well for their class. Group assignments can connect rather than isolate women and minorities while simultaneously meeting other educational goals. Fairness of groups is considered a high priority as it is part of establishing "social trust" in the classroom. Because it takes very little time to create a set of teams with GroupEng, our teaching center offers group creation based on faculty supplied rules as a service. This allows centralization of sensitive data and removes any concern of perceived grading bias based on information supplied to form groups. GroupEng groups can be imported into CATME[20] for faculty who would like to use CATME's peer evaluation tools.

Future work on GroupEng will focus on improving ease of use and improving algorithms for large, restrictive rule sets. The current sorting algorithm in GroupEng works well for all cases considered, however, group quality (number rule breaks and strength deviation) starts to drop if many more rules are added. More advanced deterministic (linear programming) or stochastic (genetic algorithms or simulated annealing) algorithms could allow better performance in difficult cases and allow more flexibility on prioritizing rules. We welcome feedback from users on how to make the overall process of using and learning to use GroupEng as efficient as possible. We hope to see GroupEng widely used to improve the quality of group selection and, through that, student learning.

Acknowledgements

Jami Joyner, Associate Director of the Diversity Programs at Cornell University, provided valuable insight and was an integral part of forming groups by hand for the first class. Jeffrey Dimiduk contributed insight in analyzing Team-Maker's algorithm, statistical analysis, and editing of the paper. Thomas Dimiduk is supported by an NSF Graduate Fellowship.

References

1. Felder, R. M., Felder, G.N., Mauney, M., Hamrin, C. E. Jr, and E. J. Dietz. 1995. A longitudinal study of engineering student performance and retention. III. Gender Differences in Student Performance and Attributes. *Journal of Engineering Education* 84 (2): 369: 151-63.

2. Rosser, S. V. 1998. Group Work in Science, Engineering, and Mathematics: Consequences of Ignoring Gender and Race. *College Teaching* 46(3) 82-8

3. Light, R. 1986. Strengthening Colleges and Universities: The Harvard Assessment Seminars. http://net.educause.edu/ir/library/pdf/ffp0604.pdf

4. http://www.makeuseof.com/dir/team-maker-random-team-generator/

5. http://www.acljohn.com/eadministration/team-maker

6. http://chir.ag/projects/team-maker/

7. http://sourceforge.net/projects/team-maker/

8. http://www.ostatic.org/team-maker

9. https://www.catme.org/login/help

10. http://www.abet.org/Linked%20Documents-UPDATE/Criteria%20and%20PP/E001%2009-10%20EAC%20Criteria%2012-01-08.pdf

11. Cornell University Faculty Institute for Diversity workshop led by Sue Rosser and Frank Tuitt, June 2009, Cornell University

12. Layton , R.A., Loughry, M. L., Ohland, M.W., and Ricco, G.D., "Design and Validation of a Web-Based System for Assigning Members to Teams Using Instructor-Specified Criteria", Advances in Engineering Education, Spring 2010

13. Mohandy, S.P. "Diversity's Next Challenges", Inside Higher Ed., March 4, 2011

14. Tuitt, F., " Afterward, Realizing a More Inclusive Pedagogy"in Race and Higher Education by Howell, A., and Tuitt, F., Harvard Educational Review pp. 254

15. Steele, C. M. , "Thin ice: 'Stereotype threat' and Black college students", Atlanctic Monthly, August 1999, pp44-54

16. Layton, R., Ohland, M., and Pomeranz, H. R., "Software for Student Team Formation and Peer Evaluation: CATME Incorporates Team-Maker" AC 2007-1565, American Society for Engineering Education, 2007

17. Cavanaugh, R., Ellis, M., Layton, R., and Ardis, M., “Automating the Process of Assigning Students to Cooperative-Learning Teams”, Proc. 2004 ASEE Annual Conf. & Exposition, 2004

18. www.GroupEng.org

19. http://yaml.org/

20. https://engineering.purdue.edu/CATME