A Project Team Building Model for Term Projects of Software Engineering Courses

Yasar Guneri SAHIN
Izmir University of Economics, Department of Software Engineering
yasar.sahin@ieu.edu.tr

Abstract
This paper proposes a new model for team building, which enables teachers to build coherent teams rapidly and fairly for the term projects of software engineering courses. Moreover, the model can also be used to build teams for any type of project, if the team member candidates are students, or if they are inexperienced on a certain subject. The proposed model takes students' preferences and the teacher's considerations into account when a team building process is required for any type of course. In addition, this paper investigates how team building models (*RandomM*: teams are built with randomly selected students; *TeacherM*: teacher selects the members for each team; *StudentsM*: students build their own teams and the proposed model) affect team performance and how gender differences affect project activities and team performance. A three-year (five semesters) teaching experiment was performed with the participation of 248 male and 79 female university students and a total of 67 software project teams. Two different One-way ANOVA tests were applied on the experimental data, and the results indicated that the proposed model was better than RandomM, TeacherM and StudentsM models in terms of project grades, and the effect of gender differences on the teams' performance and project activities was negligible.

Keywords: *architectures for educational technology system; evaluation methodologies; gender studies; improving classroom teaching*

1. Introduction

Teams are a primary mechanism for accomplishing organizational work, especially for software projects (Faraj and Sproull 2000), hence team building, team size and cooperation between team members are critical factors in providing a quality software project. Team building process has been regularly investigated over a long period, and a wide range of methods have been implemented to improve team cohesion and quality.

1.1. Literature review

The first contextual study in this field, called "Tuckman's model of team development" (Tuckman 1965), proposed a four-stage model, consisting of form, storm, norm and perform sequence. These stages are regarded as the best and idealized forms (Buchanan and Huczynski, 1997). However, Rickards and Moger (2000) proposed a new contextual framework that modifies Tuckman's well known model. They stated that "although the stages of Tuckman model may have considerable face validity as a general sequence, empirical observations of specific teams reveal complexities that cannot be explained as a simple stage sequence". They focused on two questions: "what if the storm stage never ends" and "what is needed to exceed performance norm", and proposed to replace all stages with two barriers: weak behavioral barrier and strong performance barrier. They have proposed an effective model, and have successfully tested their hypothesis with projects teams of business graduates and multiple teams within multidimensional industrial organizations. In addition, a number of modified and complementary alternative models to Tuckman's original have been proposed (Rickards and Moger, 2000; Mcgrew et al, 1999; Miller 2003; Hope et al, 2005; Tuckman and Jensen, 1977; Tuckman, 2010).

Furthermore, team coherency and its effects on team performance have also been studied by many researchers. Chung and Guninan investigated the relationship between team members' participative style and team performance in software development, and they stated that team size and the professional experience of team members moderate the relationship between participation and performance. The most important finding is that participation is significantly related to team performance in teams with inexperienced members (Chung and Guinan 1994). In addition, there are many other studies about factors that affect team performance, such as the problems caused by the distribution of projects to teams (in terms of workloads and size of the projects), difficulties in project management and collaboration problems

within groups (Beck, 2008; Humphrey et al, 2008; Mead, 2009; Jenkins, 2008; Favela and Pena-Mora, 2001; Mitchell and Delaney, 2004; Pereira et al, 2008; Robillard and Robillard, 2000).

Bardach (2004) offered a solution to these problems, his method, "elimination auction", presents a computer program for matching students to particular projects. In this approach, students are asked to vote or bid for the projects from a project pool according to several rules. Finally, project groups (teams) are assembled by students with an interest in the particular project subjects. Although the method is ingenious clever, it is a top-down approach: first, the subject is decided, and then subject coherent teams are built according to the project subject. However, there are a number of disadvantages to this approach: well-adjusted teams may not be built, and project subjects have to be defined in advance. Thus, either a very limited project pool is offered to students, or they cannot freely select any of the real life project subjects they may be interested in. However, in a bottom-up approach, teams are built first, and then a subject is decided upon by team members. Thus, this approach seems to be more advantageous in terms of project subjects.

1.2. Motivation

While these studies state some kind of team problems, they propose several solutions to team building and team management problems for industrial and business projects. However, in education, especially in software engineering education, students as team member candidates are inexperienced, and this software project might be their first encounter with project issues. Students can be sensitive, and they need to be guided very carefully through their first experience in project tasks and management. In addition to students' project skills, group participation and relationships are crucial and need to be taken into account during the team building process.

Inexperience is not the only problem in building a new team for software engineering (SE) courses' term projects. The list below highlights a number of limitations and restrictions evident in the process:

1. Time limitation: there is very limited time for building the project teams. A semester is about 15 weeks, including exams and presentations; hence team building processes should be completed in a very short time. When the teacher lets the students build their project teams, the students cannot finalize building the teams in two weeks because of problems caused by relationships between students, emotionality, avoiding responsibility, etc. These delays result in very limited time to complete the project work. Therefore, team performance and project quality may suffer.

2. Behavioral tendency problems: since all team member candidates are students, they are likely to behave in ways considered unprofessional, such as gravitating toward their best friends or academically successful students. Empirical observations reveal that sometimes, students can be unwilling to be in the same project team with certain students because of hidden conflicts between them, but they may not express this directly, causing unnecessary tension or strife. This situation decreases the motivation and total effectiveness of the team.
3. Limited project subjects: forcing students (teams) to select a project subject from a limited project pool may narrow the students' visions. Therefore, a project team should be allowed to freely select a project subject from the real world, not only from a project pool. Furthermore, it is important that they discuss their availability and interests, such as interviewing and negotiating abilities, travel restrictions etc, before deciding what the proper project can be. This process is crucial to improving project skills and creativity.

In this paper, four team building models – *RandomM (*teams are built with the randomly selected students); *TeacherM (*teacher selects the members for each team); *StudentsM (*students build their own teams); and *ProposedM (*the proposed model) are examined. When the restrictions and limitations listed above are considered, RandomM, TeacherM and StudentsM models appear not to be the best choices for building project teams for SE courses' term projects.

This paper proposes a bottom-up, straightforward algorithm that is a combination of the improved model of Gale & Shapley and Prim's minimum spanning tree algorithm to solve team building problems for SE

courses (Gale and Shapley, 1962; Prim, 1957). Gale and Shapley proposed a model to solve roommate matching problems (similar to team building for a software project). In addition, there is a recent proposal to solve roommate matching problems by Morrill (2010); however, he focused only on 2N students into N roommates pairing. Although these coupling methods need improvements in order to be used in the problem of N students pairing into M teams, their approaches are very useful for initial pairing of students into subgroups.

The model proposed in this paper is an improvement of roommate coupling method. By taking into account students' preferences, it improves team collaboration and team success, and limits team problems.

2. Team Building for Term Projects

In this study, four different team building models –*RandomM; TeacherM; StudentsM; ProposedM*- are examined as previously mentioned for term projects of software engineering courses.

2.1. Team building models

RandomM: this model is based on random selection of *n* (team size) students from a student list for each team. The teacher makes a random selection algorithm to build teams at the beginning of the semester using a variety of options, ranging from maximum team size to gender (i.e. the teacher may decide to include at least one female in each team). If there is an undesired structure in any team, the teacher finalizes manually. This is the fastest model, however it is unfair, and the one that least considers the students as individuals.
TeacherM: in this model, the teacher selects *n* (team size) members from a student list for each team using his/her knowledge of individual students. Because this model is based on manual selection of students, it requires very precise knowledge of students profile and characteristics of all students, and consequently has the potential for student objection. Thus it is very difficult to implement.
StudentsM: at the beginning of the semester, the teacher determines *n* (team size) and instructs the students to build their own teams of this size. Because many of the students know each other, most of them build very coherent teams. However, this process usually requires more than 2 weeks, and students recently joining the class, or those repeating the course may be left out of teams.

2.2. ProposedM: The Proposed model

The algorithm of the proposed model is based on agglomerative clustering and Prim's shortest path algorithm. The initial step of the proposed team building model has the students competing preference forms at the beginning of the semester. Next, the forms are transferred to a database table, and then a preference matrix is created with a number of adjustments. A computer program designed for the proposed model is then executed with several parameters, such as max group size and merging function selection. Then a team list is created and shown on the screen. If the teacher needs to revise or modify a team's structure or replace members, s/he would manually finalize the list, finally the team list is announced to students.

2.2.1. The preference form

At the beginning of the semester, the students (proposers) are asked to complete a form (Fig.1.a) for ranking their friends (acceptors) using a preference scale from 1 to n-1 (1 being highest and n-1 being lowest) according to their willingness to be in the same project group as other students. This process should be finalized within short period (i.e. in five days). Here, a proposer is a student who declares the list of preferences, an acceptor is a student who accepts a proposer as a teammate, and n is the number of students in the list. Since the proposers cannot prefer themselves, they must fill the form for n-1 acceptors. The preference forms are kept completely confidential; thus, proposers fill out the forms without anxiety. While proposers are completing the forms, they should agree to the following rules:

1. They cannot give the same preference order for more than one acceptor.

2. They have to prioritize at least k acceptors with a $P(i)$ preference level, where $P(i)$ is the preference level of the i^{th} acceptor, k is the maximum number of the students, and i=1..k. Here maximum team size (k) is previously determined by the teacher according to class size. If a proposer prioritizes i acceptors and if $i<k$, all the acceptors who are not prioritized by a proposer will be regarded as having the same preference level with $P(j) = P(i)+1$ for $j=i+1$ to n-1.
3. All students must submit a form at the beginning of a semester; otherwise the proposer agrees that all the acceptors will be regarded as having the same priority with the highest level 1.

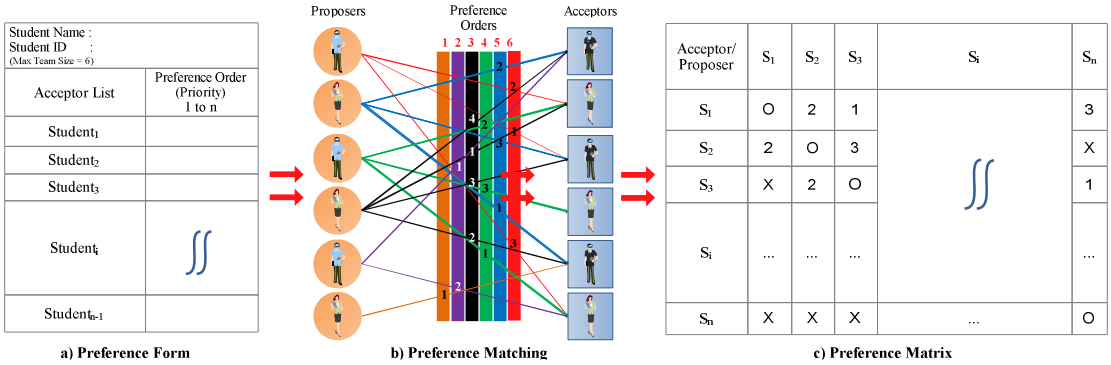

Figure 1. Transfer sequence of the preference forms to a preference matrix

2.2.2. Constructing the preference matrix

All of the completed forms are transferred into a preference matrix M (see Fig.1.c), and several revisions are made on the matrix according to assumptions 1 and 2 (assumptions are stated according to preference form filling rules). Fig.1 shows how the preference forms can be transferred to a preference matrix. In Fig.1.c, "X" denotes that no preference level is given to that acceptor.

Assumption 1: If a proposer student does not complete a form, the proposer would accept being a member of any team without complaint. Thus, the rows of these proposers in the matrix are filled by 1 (highest preference level) for each acceptor (column). $P(j)$=1, for j=1 to n-1.

Assumption 2: If a proposer does not prioritize a sufficient number of acceptors, the proposer would accept that all the acceptors who are not prioritized have the same priority level. Thus, priority level for these acceptors (columns) is set to the lowest priority given by the proposer plus 1. $P(j) = P(i)+1$ for $j=i+1$ to n-1.

When the necessary revisions are completed on the matrix, it is decomposed into a lower triangular matrix by adding the weights of each cross proposer and acceptor ($M_{ij} = M_{ij} + M_{ji}$). Figure 2 demonstrates a sample construction sequence of a preference matrix for six students.

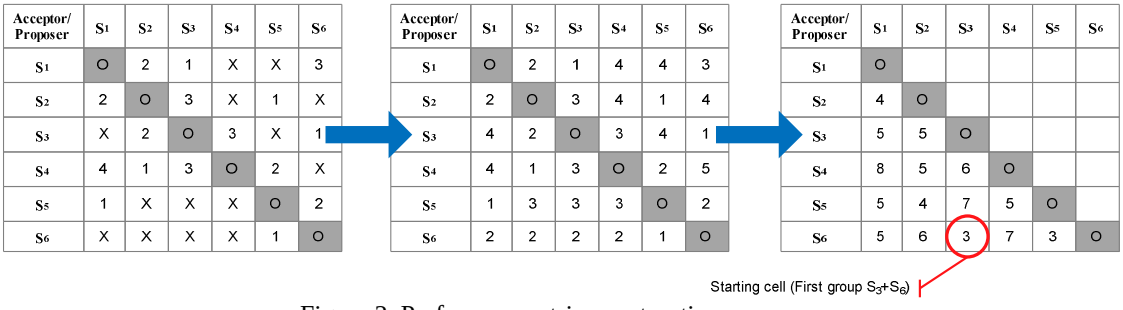

Figure 2. Preference matrix construction sequence

2.2.3. The team building algorithm

The algorithm of the proposed model is straightforward, and it is similar to Prim's minimum spanning tree algorithm. All proposers and acceptors (after triangular decomposition, only one node denotes both) are referenced by nodes and given preferences referenced by edges (weights). A sample node structure is demonstrated in Fig.3 for maximum team size two and three.

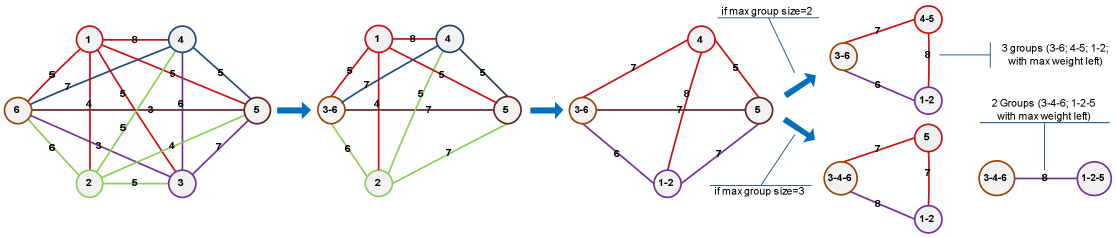

Figure 3. Sample steps for the team building algorithm

The first teams are created by combining the nodes with minimum weights (highest preference levels). When two nodes are combined in a single node, the new weights to others are then recalculated using various functions. Three different recalculation functions are considered; these are RF_{total}, RF_{max} and RF_{min} where RF_{total}, RF_{max} and RF_{min} assume that the new weight to a particular node is the sum of the weights, maximum weight and minimum weight of combined nodes to that particular node respectively. In Figure 3, the recalculation function forces that (RF_{max}) maximum weight of the combined nodes will be the new weight. Here, the model hypothesizes that when all the team building processes are completed, the total weight left (total weight between groups of proposers) is maximum, maximum weight left means that the teams have minimum weight inside, and there is a maximum challenge between groups.

The pseudo code of the model is shown in Algorithm 1. In this code, the preference matrix construction is completed between lines 1 and 7.

Algorithm 1 : Team building algorithm

Input: Students' Preference matrix M_{ij}; number of students →*n* ; max. groups size→ *maxgm*
Output: Groups

```
1  for j:=1..n do                                   // Find max preference value and set the
2      maxpref = max(preference in tuple M[j]);     // empty columns to maxpref+1 for each
3      for i := 1..n do                             // tuple (Student)
4          if M[i,j] = 0 then Set M[i,j] = maxpref + 1;
5      end {for}
6  end {for}
7  Transform(M);                                    // Get the total of the edges (Transform M
8  StudentsLeft = n;                                // to // a lower triangular matrix)
9  while LeftStudents > 0 do
10     FindMinRelationShip(StudentX,StudentY,Weight)   // Find the students having min weight
11     if CheckMaxGroupSize(StudentX,StudentY) then    // X and Y are student or groups of
                                                       students
12         MergeStudentsYtoX(StudentX,StudentY,Weight); // Create group and get total group
                                                        weight
13         RemoveStudents(StudentY);                   // Construct the group and remove the
14         StudentsLeft := StudentsLeft – Sizeof(StudentY); // group Y from Preference Matrix
15         SetupAGroup;
16     end {if}
17     else SolveMaxGroupSizeProblem(X,Y);          // if any problem with group size
18 end {while}
```

Next, until no student is left alone, the algorithm continues to build teams. Meanwhile, some group matching problems arise (i.e. maximum group size might be exceeded because of the nature of the algorithm). In these problematic cases, SolveGroupSizeProblem procedure is executed to rearrange the

teams' members. In this model, different functions can be considered to improve the efficiency of the algorithm or to maximize total weight left. Improvements in the algorithm can be investigated in a future study.

3. Experimental Results and Data Analysis

A three-year (five semesters) teaching experiment was performed to test the team-building models, and to study how gender differences affect project activities and team performance.

3.1. Data collection

The research was conducted at a university with the participation of 248 male and 79 female 3rd year students studying in the Software and Computer Engineering departments (a total of 67 project teams). In this study, two major software engineering courses - Software Specification and Design and Principles of Software Engineering - were selected. Table 1 shows the demographic information of the participants with respect to RandomM, TeacherM, StudentsM and ProposedM team building models, and team compositions based on gender. In the table, "Only M", "Only F", "#M>#F" and "#F>=#M" denote the teams that include only male members, only female members, more males than females, and more or equal females than males, respectively. The data were gathered from five semesters of the SE courses taught in 2007-2008, 2008-2009 and 2009-2010 academic years for the models RandomM, TeacherM, StudentsM and ProposedM, respectively.

Table 1. Demographic information of the participants

Team building model	Students			Project Teams (with male and female distributions)				
	Male	*Female*	*Total*	*Only M*	*Only F*	*#M>#F*	*#F>=#M*	*Total*
RandomM	73	23	96	7	1	7	3	18
TeacherM	35	16	51	4	0	4	5	13
StudentsM	102	24	126	13	1	9	4	27
ProposedM	38	16	54	2	0	4	3	9
Total	248	79	327	26	2	24	15	67

3.2. Analysis of the results

During five semesters, five SE course term project grades were collected to evaluate the success of the proposed model, effects of gender difference on the project activities, and team performance identified by Test1 and Test2 respectively. The project grades were the combination of several project activities, documentation, and team performance. A project grade is calculated using the following components:

1. 30% of Meeting reports: Average score of a total of ten meeting reports which should be submitted every week after the project subjects are assigned.
2. 50% of Documentation: Project documentation score is calculated using feasibility report, UML diagrams, the document format and the document completeness and consistency.
3. 20% of Presentation: Each team must make an oral presentation at the end of the semester on its project. The presentation score is calculated using presentation style, presentation documents, and questions-answers about the project.

In addition, each team member must grade all team members, including herself/himself individually immediately after the presentations. Meanwhile, if a student fails to grade teammates, a score of 100% is given to each team member, and the student who fails to grade is given 0%. Using these self grading results and the teacher's observations, a contribution ratio is calculated for each team member. To finalize the project grade of a student, the raw project score of the team (30% of meeting reports + 50% of documentation + 20% of presentation) is multiplied by the student's contribution ratio. Therefore, a project grade reflects both the project activities and the team's performance impartially.

To analyze (Test1) the difference between the proposed team building model and the other models, one-way variance analysis (ANOVA) and a comparison were performed, and significance level was taken as

5%. Table 2 demonstrates the ANOVA test results of the first test, the achievement test of the proposed model. Here, hypothesis H_0: There is an insignificant difference between models.

Table 2. Descriptive statistics of the results for the first test (Test1)

Models	*N*	*total*	*avg.*	*variance*
RandomM	18	1244.613	69.145	318.252
TeacherM	13	880.302	67.716	234.194
StudentsM	27	2213.411	81.978	257.655
ProposedM	9	788.482	87.609	34.388

Groups	*SS*	*df*	*MS*	*F*	*P*	*F (criteria)*
Between groups	3897.825	3	1299.275	5.387	0.002	2.751
Group inside	15194.730	63	241.186			
Total	19092.550	66				

The ANOVA results (F=5.387, p=0.002 < 0.05) show that H_0 is rejected, as it is obvious that there are significant differences between models. Therefore, in terms of project grades ($avg_{ProposedM}$>$avg_{StudentsM}$>$avg_{RandomM}$>$avg_{TeacherM}$), the proposed model is a better model than the others, and it can be said that the proposed model is successful in team building.

Furthermore, Table 3 demonstrates the one-way ANOVA test results of the second test (Test2). Here, hypothesis H_0 is that gender differences have little or no effect on project grades, project activities and team performance.

Table 3. Descriptive statistics of the results for the second test (Test2)

Groups	*N*	*total*	*avg*	*variance*
Only M	26	2008.569	77.253	290.783
Only F	2	177.740	88.870	18.850
#M>#F	24	1748.923	72.872	357.512
#F>=#M	15	1191.576	79.438	201.083

Groups	*SS*	*df*	*MS*	*F*	*P*	*F (criteria)*
Between groups	766.187	3	255.396	0.878	0.457	2.751
Group inside	18326.370	63	290.895			
Total	19092.550	66				

The one-way ANOVA results for testing the effect of gender differences on project grades (F=0.878, p=0.457 > 0.05) shows that H_0 is accepted, and gender differences do not impact on project grades, project activities and team performance. In the experimental study, the percentage of female students was 24.15% (79/327). Because of this low percentage, there were less "#F>=#M" teams than "#M>#F" teams, and only two "Only F" teams. Therefore, the results would be different if the percentage of females were higher. In fact, low percentage of females in engineering education is a common problem of almost all countries, especially in developing and underdeveloped countries (Bouville, 2008; Zengin-Arslan, 2002).

4. Discussions and Conclusion

Experimental results show that the proposed model is a gratifying model for team building processes for term projects of fundamental software engineering courses. Another meaningful result of the study is that the gender difference is insignificant on project activities and team performance.

This study considered not only a measurable factor (i.e. projects' grades), but also a number of different factors which may affect the performance of the teams, project activities, and students' skills. All of these factors and their effects on the related subjects are given in Table 4, and the models are compared according to several factors using a five-level scale of appropriateness (1 for very good and 5 for very bad).

Table 4. The factors and their effects on the team building processes (1 very good to 5 very bad)

Factor description	*RandomM*	*TeacherM*	*StudentsM*	*ProposedM*
Objections to team members	5	4	1	1
Restiveness in teams	4	3	2	2
Time requirement for the team building process	1	3	5	2
Correspondence with meeting schedule	3	2	2	2
Project completion on time	2	2	2	1
Project appropriateness to course goal	3	3	2	2
Total	18	17	14	10

The observations show that *ProposedM* and *StudentsM* models are the best models for minimizing of students' objections about team structures. If the teacher builds the teams or teams are built randomly, this can lead to continuous student complaints. Therefore, this may lead to an uncomfortable project environment, and it may decrease the team performance. Building teams randomly would be the best choice for a teacher with very limited time to build project teams; (actually it takes approximately five minutes using student list). Remaining factors' scores are similar to all models with small differences. Finally, the results highlight the fair that the proposed model is the most successful model in terms of many factors that may affect the team performance, project activities and students' skills.

Although test results show that gender differences have no overall effect on team performance during the semesters that the experimental studies were conducted, a number of distinct effects of gender differences have been observed as follows;

1. The teams with more female members had less competition between team members.
2. Homogenous teams were the problem free teams in terms of gender structure, thus no group problems were noticed.
3. While the females were very successful in negotiations and interviews with clients, the males were more successful in the completion of the project documents and related UML diagrams.

In addition, there are a number of lessons learned during the experimental study period. Although unlikely, it is possible for teams to have floater students. Although several teams with a homogeneous structure mentioned nothing about group problems during the semester, the team members awarded each other very low scores; perhaps because rather than directly criticize their friends, they prefer to express their feelings by awarding them a low score. In fact, contribution ratio implementation appears to be a solution for determining which students were floaters. Nevertheless, building heterogeneous teams is strongly recommended, and gender should be considered as a new parameter for team building algorithms as a future study.

Furthermore, highly consistent teams may show very high performance in term project studies. However, students should be strongly motivated and highly skilled at problematic circumstances, and they have to learn how to handle difficulties they will face in their future; thus teams should not necessarily be comprised of completely compatible team members. Actually, it is observed that the proposed model helped to build these kinds of teams, since the students (proposers) could not prioritize all of their friends with the same preference level, and the teams did not include more than two students at the highest preference level.

The experimental results show that for team building the proposed model presented in this paper is a more effective model than the other models (RandomM, StudentsM and TeacherM). Although it may not be the most suitable team building model for all types of project, it enables very fast, reliable and relatively fair team building for software engineering course term projects. In addition, the model can be easily implemented for all types of courses in education due to its simple and understandable design, and it can be improved by discovering new merging functions and algorithms, which can be studied in the future. Team building is not the only issue for the SE course projects, so a number of complementary tools, technologies or education styles can be considered to improve the teams' performance and project activities, such as web-based project monitoring and a comprehensive pool for project subjects.

References

Beck, J. (2008). Fair Division as a Means of Apportioning Software Engineering Class Projects. In *39th ACM Technical Symposium on Computer Science Education* Portland, OR.

Bouville, M. (2008). On Enrolling More Female Students in Science and Engineering. *Science and Engineering Ethics, 14(2),* 279-290.

Buchanan, D. & Huczynski, A. (1997). *Organizational Behaviour: An Introductory Text*. (3rd ed.) Prentice-Hall, London.

Chung, W. Y. & Guinan, P. J. (1994). Effects of participative management on the performance of software development teams. In *Proceedings of the 1994 computer personnel research conference on Reinventing IS : managing information technology in changing organizations* (pp. 252-260). Alexandria, Virginia, United States.

Faraj, S. & Sproull, L. (2000). Coordinating expertise in software development teams. *Management Science, 46,* 1554-1568.

Favela, J. & Pena-Mora, F. (2001). An experience in collaborative software engineering education. *IEEE Software, 18,* 47-53.

Gale, D. & Shapley, L. S. (1962). College Admissions and the Stability of Marriage. *American Mathematical Monthly, 69,* 9-14.

Hope, J. M., Lugassy, D., Meyer, R., Jeanty, F., Myers, S., Jones, S. et al. (2005). Bringing interdisciplinary and multicultural team building to health care education: The Downstate Team-Building Initiative. *Academic Medicine, 80,* 74-83.

Humphrey, W., Musson, R., & Salazar, R. (2008). Preparing Students for Industry's Software Engineering Needs. In *39th ACM Technical Symposium on Computer Science Education* Portland, OR.

Jenkins, M. (2008). Teaching Computer Aided Software Engineering at the Graduate Level. In *13th Annual Conference on Innovation and Technology in Computer Science Education* Madrid, SPAIN.

Mcgrew, J. E., Bilotta, J. G., & Deeney, J. M. (1999). Software team formation and decay - Extending the standard model for small groups. *Small Group Research, 30,* 209-234.

Mead, N. R. (2009). Software engineering education: How far we've come and how far we have to go. *Journal of Systems and Software, 82,* 571-575.

Miller, D. L. (2003). The stages of group development: A retrospective study of dynamic team processes. *Canadian Journal of Administrative Sciences-Revue Canadienne des Sciences de l Administration, 20,* 121-134.

Mitchell, G. G. & Delaney, J. D. (2004). An assessment strategy to determine learning outcomes in a software engineering Problem-based learning course. *International Journal of Engineering Education, 20,* 494-502.

Morrill, T. (2010). The Roommates Problem Revisited. *Journal of Economic Theory, 145,* 1739-1756

Pereira, J., Cerpa, N., Verner, J., Rivas, M., & Procaccino, J. D. (2008). What do software practitioners really think about project success: A cross-cultural comparison. *Journal of Systems and Software, 81,* 897-907.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal, 36,* 1389-1401.

Rickards, T. & Moger, S. (2000). Creative leadership processes in project team development: An alternative to Tuckman's stage model. *British Journal of Management, 11,* 273-283.

Robillard, P. N. & Robillard, M. P. (2000). Types of collaborative work in software engineering. *Journal of Systems and Software, 53,* 219-224.

Tuckman, B. (1965). Developmental sequence in small groups. *Psychological Bulletin, 63,* 384-399.

Tuckman, B. W. & Jensen, M. A. C. (1977). Stages of Small-Group Development Revisited. *Group & Organization Studies, 2,* 419-427.

Tuckman, B. W. (2010). Developmental Sequence in Small Groups. *Psychological Bulletin, 63,* 384-399.

Zengin-Arslan, B. (2002). Women in engineering education in Turkey: Understanding the gendered distribution. *International Journal of Engineering Education, 18(4),* 400-408.