A Multi-Objective Genetic Algorithm for Software Development Team Staffing Based on Personality Types

Constantinos Stylianou[1] and Andreas S. Andreou[2]

[1]Department of Computer Science, University of Cyprus,
cstylianou@cs.ucy.ac.cy
[2]Department of Computer Engineering and Informatics, Cyprus University of Technology,
andreas.andreou@cut.ac.cy

Abstract. This paper proposes a multi-objective genetic algorithm for software project team staffing that focuses on optimizing human resource usage based on technical skills and personality traits of software developers. Human factors are recognized as critical aspects affecting the rate of success of software projects, as well as other properties, such as productivity, software quality, performance, and job satisfaction. However, managers often rely solely on technical criteria to staff their projects, which risks overlooking these important aspects of software development, such as the abilities and work styles of developers. The behaviour and scalability of the algorithm was validated against a series of hypothetical projects of varying size and complexity, and also through a real-world project of an SME in the local IT industry. The approach demonstrated a sufficient ability to generate both feasible and optimal staffing solutions by assigning developers most technically competent and suited personality-wise for each project task.

Keywords: Software Project Management, Team Staffing, Genetic Algorithms Personality Type Matching, Five-Factor Model.

1 Introduction

For many years, researchers in the area of software engineering have argued that human factors should be taken into consideration when developing software since human resources are the most, if not only, crucial resource available for software development companies. Hence, most research targets software project managers since it is their responsibility to assign these resources to tasks and their selection directly influences the success of a software project, especially with respect to critical software development issues such as performance, productivity, quality, and job satisfaction.

Assigning software developers to tasks is not a simple process as each developer has their own strengths and weaknesses, which often extend beyond the "academic" knowledge acquired from a university degree, for example, or the experience gained from using a specific tool or technology over a number of years. They also concern traits and behaviours in the form of abilities and competencies that develop from each individual's personality and psychological processes. For this reason, it is argued that

software project managers need to look beyond technical skill-based and experience-based methods when selecting developers, as it is equally important to deal with interpersonal relationships and social aspects present in software development processes and organizations [1]. The goal, therefore, of the proposed approach is to support software project managers of SMEs in the selection and allocation of the most suitable developers to tasks, by attempting to optimize the assignments based on technical skills and personality traits of developers using a multi-objective genetic algorithm.

The remainder of this paper is organized as follows: Section 2 provides a brief overview of the most recent research attempts proposed regarding the inclusion of human factors in software project team staffing. Section 3 describes the method used to identify the various occupations found in the software development industry together with their characteristics and requirements in terms of technical skills and personality traits. In section 4, a description of the methodology is provided, in which various aspects of the multi-objective optimization approach are described. Next, section 5 presents the evaluation of the proposed approach, describing the projects used in the experiments carried out followed by a discussion on the results obtained. Finally, section 6 concludes the paper with a synopsis and comments on future work.

2 Related Work

One of the most common human factors affecting software development addressed in literature is the area of personality, with many different investigations carried out over the years. Some attempts have looked into studying the various types of personality of software development professionals. The results from such studies can provide helpful insight about the type of personalities attracted to the software development industry, and also can be used by companies looking to recruit or release personnel, or by software project managers attempting to assign developers to tasks. Some of the questions asked in these studies focus on whether software professionals share a common personality type [2] and whether they differ from the rest of the general population [3]. Some studies concentrate on specific professions, such as systems analysts [4], while others examine occupations from all development phases [5]. More recently, Varona, et al. carried out a survey of existing studies that attempted to profile IT-related professions in order to identify trends and changes, and to form a better understanding of the software industry's human resources [6]. Another area of research has focused on assessing the effects of personality on various properties of software development. For instance, personality has been explored with respect to team effectiveness and performance [7, 8], cohesion [9], software quality [10], as well as job satisfaction [11]. More recent research has also concentrated on how personality influences pair programming in agile methodologies [12, 13].

Research work relating to team staffing and formation have also been carried out, whereby attempts to build teams for software projects take into account human factors such as developers' personality types [14] and capabilities [15]. Some of the team staffing approaches proposed employ computational intelligence techniques to aid with assigning developers to tasks. For instance, Martínez et al. [16] proposed

RAMSET as a methodology for assigning roles in software engineering teams that adopted a learning approach based on an adaptive network-based fuzzy inference system to recommend the best resource allocation possible. Also, André, Baldoquín and Acuña [17] formalized a set of rules to match the personality types of developers to fixed project roles defined by a set of generic and technical competencies. They then transformed these rules into objective functions and constraints, and applied them in various heuristic algorithms (such as, random restart hill-climbing, simulated annealing and Tabu search) to carry out optimal assignment of developers to roles.

Despite much research being conducted concerning human factors, and in particular personality, most focuses on exploring and investigating their effects in various aspects of software development. There is still, however, a great need for tools to support software project managers incorporate these factors systematically in their staffing activities. Therefore, a major contribution of this work is providing such a tool, which carries out team staffing in an automated fashion and that, in addition, employs computational intelligence through the application of multi-objective optimization to handle the balancing of personality traits and technical skills and knowledge.

3 Personality Traits of Software Development Occupations

One aspect of the optimization approach implemented in this paper concerns the evaluation of selected developers based on the suitability of their personality traits with respect to the type of task they have been assigned to carry out. To do this, the software professions most commonly found in SMEs of today's software development industry were first identified using the 2010 Standard Occupational Classification (SOC), which serves as a systematized taxonomy of the majority of existing professions identified by the U.S. Bureau of Labor Statistics [18]. Then, detailed analysis of each profession was carried out using the Occupational Information Network (O*NET) Resource Center [19], which provides a content model and an online database defining standardized and occupation-specific descriptors of each profession using the SOC system coding. Each occupation's job-related and worker-related characteristics and requirements were retrieved containing information on: the abilities and work styles of workers, the skills required by workers, and the work activities of occupations. Once these key requirements and characteristics were identified, the most suitable personality traits required by developers to carry out activities of each profession were then were associated with corresponding personality traits. These are expressed using five basic domains of personality, which comprise the Five-Factor Model (FFM), originally identified in 1961 by Tupes and Cristal [20] and later operationalized by Costa and McCrae through the NEO-Personality Inventory (NEO-PI) [21]. The Five-Factor Model has been widely adopted in many academic and application disciplines where personality measures have been required, and is a common instrument in cases involving career and personnel assessment. Specifically, the five domains are described as follows:

- *Neuroticism* reflects the level to which an individual is predisposed to experiencing negative emotions, such as sadness, embarrassment, fear and anger.
- *Extraversion* refers to the level to which an individual engages with their external world through interpersonal interactions, as well as their energy and predisposition to experiencing positive emotions.
- *Openness* to experience concerns an individual's tendencies regarding intellectual curiosity, creativity and variety in interests and experiences.
- *Agreeableness* involves interpersonal orientation with regards issues, such as compassion, social harmony, cooperation, and trust.
- *Conscientiousness* relates to the degree of self-discipline and control of impulses, and also ambition and organization.

Finally, the desired level of each of the domain was determined so as to ascertain whether a profession requires either a {1:low, 2:medium or 3:high} level of that particular domain. The same was applied to personality traits possessed by developers so that comparisons between the two can take place using a simple distance measure.

An important issue here is that of the validation of each profession's associated personality traits that were selected as desirable. A large number of studies were used for this purpose, such as [2, 3, 4, 5, 6], as well as other related material, for example, career handbooks suggesting the best occupation based on personality types. However, the validation process is not currently in the scope of this paper, since the desired personality types can be easily modified and, furthermore, the study's focus is on how well the chosen encoding performs in optimizing developer assignments.

4 Description of Methodology

The goal of the proposed approach is to allow project managers of SMEs in the software industry to staff their projects with the most suitable teams taking into account the technical knowledge and skills of available developers as well as their personality traits and abilities. These, however, may be viewed as conflicting in some cases, where a developer may be technically capable but does not hold the appropriate traits required by activities of a task, or vice-versa. For example, if a programming task requires skills in a specific programming language, a developer possessing a high level of such skills may not necessarily possess a low level of extraversion, which is one of the desired traits of programming tasks. On this basis, a software project manager would encounter difficulties in trying to make the best selection and assignment of resources whilst trying to balance the two. Also, due to the fact that there are many different possible combinations to examine, a software project manager will be required to perform an exhaustive assessment of all possible permutations, which only becomes more difficult when the number of tasks to be performed and the number of available developers increases. Therefore, in order to decrease the search space and handle the NP-hard nature of such a problem [22], a multi-objective optimization approach was adopted. In particular, two objective functions were modelled in an implementation of the Non-dominated Sorting Genetic Algorithm II (NSGA-II), introduced by Deb in 2002 [23]. With this technique, a set of optimal solutions will be

produced suggesting a collection of possible assignments of developers to project tasks aiming to satisfy the two aforementioned considerations. Furthermore, due to the involvement of constraints influencing the feasibility of solutions, the Constrained NSGA-II algorithm was applied. The algorithm promotes the solution diversity using a crowded comparison operator during its selection procedure and population reduction process. In addition, because parent and offspring populations are combined before non-dominated sorting takes place, elitism (i.e., preservation of the best solutions) is always ensured. Further details of the NSGA-II can be found in [23].

4.1 Encoding and Representation

Each software project consists of a number of tasks that need to be carried out. For each task, developers need to be assigned to perform the activities involved. Therefore, since it is the selection of developers that forms the basis of evaluation, each project task is denoted by a string of bits, and each bit represents one specific developer. If a bit in the string has a value of '1', then this signifies that the specific developer is assigned to work on the task, whereas a value of '0' indicates that the developer has not been selected for the task. Overall, if a software project consists of T tasks and there are E available developers, then each solution would be represented by an individual in the algorithm using (TxE) bits.

Since it is possible that a development company follows its own method to evaluate technical skills and knowledge, each developer is simply required to be rated based on each of the skills required by task activities in a normalized form in the range [0, 1], meaning that low possession of skill will be denoted by a value closer to zero, and high possession of a skill will be denoted by a value closer to one. Project managers can use different metrics, such as experience or IT-related aptitude tests, or can even use the experience requirements and occupation-specific information set suggested by the O*NET Content Model. As part of on-going research efforts in this area, a method to rate and match developers' skills and knowledge is currently in progress. On the other hand, regarding personality traits, developers are assessed specifically using the NEO-Five-Factor Inventory and their five domain scores are used.

4.2 Objective Functions and Constraints

A total of three objective functions were created for the evaluation of each solution along with two constraints to measure the degree of feasibility of each solution.

Technical skills objective function (f_1). This maximization function is responsible for evaluating a solution based on the assigned developers' levels of technical skills and knowledge required by the activities of each task. The objective function's value is calculated for each task by adding the maximum skill level possessed by the developers assigned to the average skill level possessed by the developers assigned, as shown in Eq. (1).

$$f_1 = \max(skill\_level\_of\_assigned\_developers) + \\ + avg(skill\_level\_of\_assigned\_developers) \quad (1)$$

Personality traits objective function (f_2). This maximization function is responsible for evaluating a solution based on developers' personality traits. Using Eq. (2), a distance is computed between the desired levels of the FFM domains for the profession and the selected developer's levels of the FFM domains. If more than one developer is assigned to work on a task, then the average of the distances is used.

$$f_2 = \sum_{i=1}^{5} \left| desired\_level\_in\_domain_i - developer\_level\_in\_domain_i \right| \quad (2)$$

Team size objective function (f_3). This minimization function is responsible for evaluating a solution based on the number of developers assigned to each task. The inclusion of such a fitness measure is important in order to maximize resource utilization and avoid unnecessary assignments. For each task, the inverse of the number of team members is calculated, as shown in Eq. (3).

$$f_3 = \frac{1}{number\_of\_assigned\_developers} \quad (3)$$

Skills satisfied constraint (c_1). The purpose of this constraint function is to measure the feasibility of a solution with respect to the technical skills required by activities of a task. Using Eq. (4), a solution is feasible only if for all activities, there is at least one developer selected to carry it out who possesses the necessary technical skills.

$$c_1 = \frac{number\_of\_unsatisfied\_skills}{total\_number\_of\_project\_skills\_required} \quad (4)$$

Developer availability constraint (c_2). The second constraint implemented involves checking the feasibility of a solution regarding the availability of developers when assigned to tasks that are carried out simultaneously. An assumption made in this approach is that no developer can work on more than one task at any given point in time. Furthermore, it is also assumed that the project's schedule is fixed beforehand and, thus, each task has already been allocated a specific "time slot" to be carried out. Eq. (5) determines this constraint value by calculating the number of days a developer has been assigned to more than one task throughout the duration of the project.

$$c_2 = \frac{number\_of\_days\_assigned\_with\_conflicts}{total\_number\_of\_days\_assigned\_in\_the\_project} \quad (5)$$

4.3 Algorithm Parameters and Settings

For the execution of the multi-objective genetic algorithm, a population of 100 individuals were used. The fast non-dominated sorting procedure was applied to rank the individuals in terms of their fitness and feasibility, after which a tournament selection of size 4 was used to select which parents were to enter the mating pool. The best two

parents were then set to produce offspring by the application of a recombination operator (one-point crossover) with a likelihood of 0.80, and a mutation operator (single bit-flip mutation) with a $(1/number\_of\_project\_tasks)$ probability. The population evolves by repeating the steps from the selection of individuals until the termination criteria are met or the maximum number of iterations have been reached (set at 2000).

5 Results of Experiments

5.1 Design of Experiments

Two experiments were carried out to evaluate the proposed methodology. First, in order to evaluate the validity and scalability of the multi-objective optimization algorithm, two hypothetical software projects consisting of 20 and 30 tasks each were created based on the input of several project managers of SME software development companies as to the basic structure, size and complexity of the type of software projects they usually undertake. For both projects, the skill levels and personality traits of the available software developers were selected so as to represent the best-case and worst-case scenarios for the proposed team staffing approach. For the best-case scenario, all available developers possessing the highest skill levels also possessed the most suitable personality traits. On the other hand, for the worst-case scenario, all the available developers possessing the highest skill levels possessed the least suitable personality traits, and vice-versa. Through these two extreme cases, both the behaviour and correctness of the optimization approach could be observed and analyzed, and also the competitive nature of the objective functions could be investigated.

In the second experiment, a real-world software project developed by a local IT

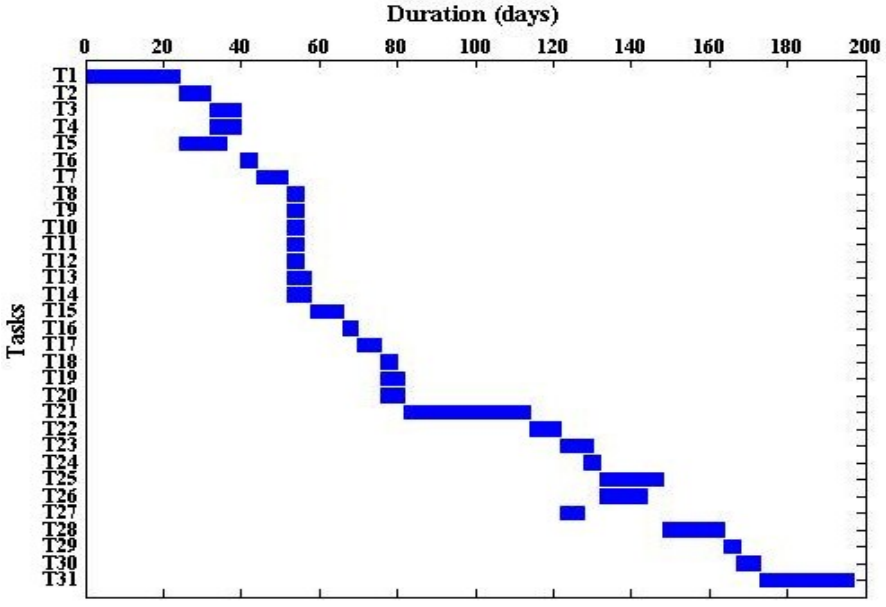

Fig. 1. Real-world software project schedule (vessel policies management system).

company was used that related to the implementation of a vessel policies management system for a large insurance brokers company. The project, whose schedule is shown in Fig. 1, consisted of 31 tasks involving project management, design, programming and testing activities. A total of four developers were available to carry out the project, each possessing a varied set of skills at different levels, as well as unique personality traits, which were determined by administering the NEO-FFI-3 [21].

5.2 Results and Discussion

In the first experiment, the algorithm was executed 10 times for both hypothetical projects using the best-case and worst scenarios. The results are shown in Table 1.

Table 1. Results obtain from the first experiment using hypothetical software projects.

Experiment	Average Number of Unique Solutions	Execution Time (min.)
Best-case scenario (20 tasks)	1	22.56
Worst-case scenario (20 tasks)	94	22.86
Best-case scenario (30 tasks)	1	23.90
Worst-case scenario (30 tasks)	95	26.08

For both hypothetical projects, the algorithm managed to provide both feasible and optimal solutions when performing team staffing in the best-case scenario. Specifically, the algorithm always managed to assign the most suitable developer with respect to both the technical skill levels and personality traits possessed, and never assigned a developer who was less suited in either aspect. Furthermore, the Pareto front consisted of individuals representing the same optimal solution. This was expected since in the "optimistic" case there would always be only one possible ideal assignment existing for each task. In the worst-case scenario, the algorithm's job was to try to balance the two objective functions, since no developers possessed both the highest skill levels and most suitable personality traits for any task. For both hypothetical projects, it was observed that this time the individuals of the Pareto front represented a number of different solutions, as seen in Fig. 2. Such behaviour again was anticipated since for each task either skill levels or personality traits could be given preference – but not both due to the nature of the characteristics of the available developers. The general behaviour of the algorithm was, thus, validated as correct.

In the second experiment, the algorithm was also executed 10 times. However, the results obtained from these executions showed that the algorithm did not actually produce any optimal solutions but, in fact, consistently generated the same infeasible developer assignments with respect to their availability (constraint c_2). This observation is mainly attributed to the small number of available developers in combination with a relatively high number of concurrent tasks within the project. Specifically, in cases where several tasks requiring the same skills and personality traits were set to execute simultaneously (e.g., tasks T8-T14 in Fig. 1), the algorithm would encounter difficulties in finding an optimal assignment of developers possessing these to the

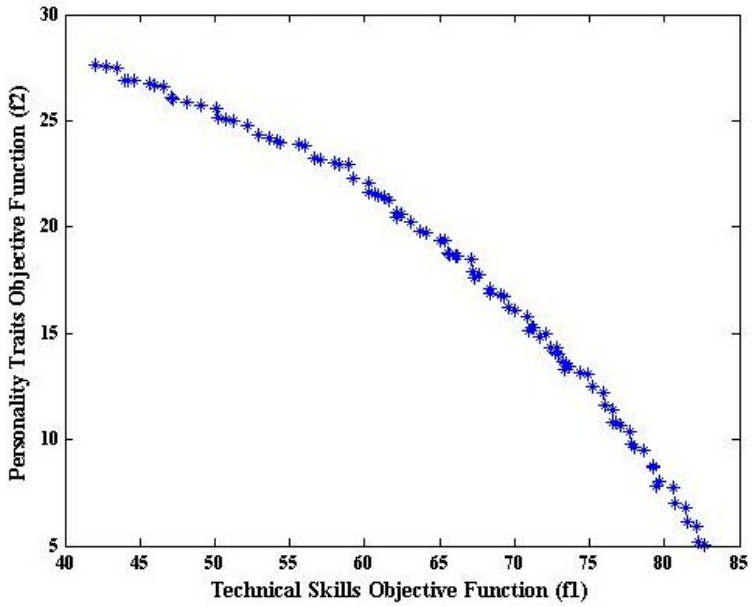

Fig. 2. Pareto front of hypothetical project (30 tasks) worst-case scenario (f_1 vs. f_2).

tasks, simply because the available resources were insufficient for such concurrent scheduling of tasks.

What's interesting, however, is that whilst consulting with the project's manager to understand the method that was used to allocate human resources to tasks, it was established that the project suffered from schedule overruns due to the lack of available resources. This shows the potential of such approach as it can be used as means to pinpoint possible staffing caveats for project managers, who would then be able to seek a solution by either revising their original schedule of tasks or even consider hiring or buying the services of developers for tasks that could not be optimally allocated resources. To further investigate this, the experiment was repeated without inclusion of the developer availability constraint. The results this time showed that the algorithm was able to produce both feasible and optimal assignments, averaging around 95 unique Pareto front solutions over 10 executions. In some cases developers possessing high levels of skills required for tasks but less suitable personality traits were chosen and in other cases developers most suited with respect to personality traits were preferred even if they possessed lower levels of skills for the tasks they were assigned to. This further enhanced the initial belief that the two objectives can indeed be competing. In such a case, it is up to the project manager to decide which of the two options has higher priority and, therefore, should be followed.

6 Concluding Remarks

The approach described in this paper proposes the innovative use of computational intelligence as a means to help software project managers solve the problem of team

staffing. In particular the approach suggests the use of a multi-objective genetic algorithm for assigning software developers to project tasks based on technical skills and personality traits. By taking into consideration these factors, it allows software project managers to view team staffing and human resource allocation from an alternative perspective, since software project success is considered to be largely influenced by the human factors present in software development. The results obtained from several experiments indicate that the algorithm is capable of generating adequate and feasible solutions, balancing the two objectives where necessary, and has the potential to constitute a decision support tool for software project team staffing.

One of the major contributions of this approach is that it can allow project managers to foresee possible resource issues arising during development. With respect to either or both constraint functions, if the algorithm is unable to find feasible solutions when applied to a specific project whose schedule is fixed, this could indicate that the available resources are not sufficient or adequate enough to carry out the software project. This is very useful for project managers since it would allow them to revise their project schedule and attempt to staff their team in a slackened timeframe. Alternatively, without modifying the project's schedule, a project manage may use the results to recruit extra resources (possessing either higher skill levels or more suitable personality traits or both). In a similar way, the approach can be used to examine whether the development company has the required capacity in terms of human resources before bidding for a software project.

As part of future work, improvements can be made to the way in which developer skill levels are measured and evaluated Also, additional objective functions can be introduced, such as the minimization of a project's cost based on developers' salaries. Attempts have also been made to collect data from development companies for the purposes of further evaluation using real-world projects, in addition to comparison of the approach with other computational intelligence techniques. Another possible future enhancement can involve integrating the proposed approach with other techniques for software project management activities taking into account the solutions generated from the current optimization method. This could allow for a comparative analysis of the effectiveness of the approach when combined and used in parallel with other models. Possible applications include feeding the solutions generated by the algorithm into an intelligent scheduling mechanism or, alternatively, using the output of assigned developers to help predict the cost of a software project (per phase or as a whole).

References

1. Amrit, C.: Coordination in Software Development: The Problem of Task Allocation. In: 27th International Conference on Software Engineering, pp. 1--7. ACM, New York (2005)
2. Moore, J.E.: Personality Characteristics of Information Systems Professionals. In: 1991 Conference on Computer Personnel Research, pp. 140--155. ACM, New York (1991)
3. Wynekoop, J.L., Walz, D.B.: Revisiting the Perennial Question: Are IS People Different? ACM Database, 29, 62-72 (1998)

4. Smith, D.C.: The Personality of the Systems Analysts: An Investigation. ACM SIGCPR Computer Personnel, 12, 12--14 (1989)
5. Capretz, L.F., Ahmed, F.: Making Sense of Software Development and Personality Types. IT Prof., 12, 6--13 (2001)
6. Varona, D., Capretz L.F., Pinero, Y., Raza, A.: Evolution of Software Engineers' Personality Profile, ACM SIGSOFT Soft. Eng. Notes, 37, 1--5 (2012)
7. Peeters, M.A.G., van Tuijl, H.F.J.M., Rutte, C.G., Reymen, I.M.M.J.: Personality and Team Performance: A Meta-Analysis. Eur. J. Personality, 20, 377--396 (2006)
8. Capretz, L.F., Ahmed, F.: Why do we Need Personality Diversity in Software Engineering? ACM SIGSOFT Soft. Eng. Notes, 35, 1--11 (2010)
9. Karn, J.S., Syed-Abdullah, S., Cowling, A.J., Holcombe, M.: A Study into the Effects of Personality Type and Methodology on Cohesion in Software Engineering Teams. Behav. Inf. Technol., 26, 99--111 (2007)
10. Fernández-Sanz, L., Misra, S.: Influence of Human Factors in Software Quality and Productivity. Lect. Notes Comput. Sci. 6786, 257--269 (2011)
11. Acuña, S.T., Gómez, M., Juristo, N.: How do Personality, Team Processes and Task Characteristics Relate to Job Satisfaction and Software Quality? Inform. Software Tech., 51, 627--639 (2009)
12. Sfetsos, P., Stamelos, I., Angelis, L., Deligiannis, I.: An Experimental Investigation of Personality Types Impact on Pair Effectiveness in Pair Programming. Empir. Softw. Eng., 14, 187--226 (2009)
13. Salleh, N., Mendes, E., Grundy, J., St. J. Burch, G.: An Empirical Study of the Effects of Conscientiousness in Pair Programming using the Five-Factor Personality Model. In: 32nd ACM/IEEE International Conference on Software Engineering, pp. 577--586, ACM, New York (2010)
14. Rutherfoord, R.H.: Using Personality Inventories to Help Form Teams for Software Engineering Class Projects. ACM SIGCSE Bulletin, 33, 73--76 (2001)
15. Acuña, S.T., Juristo, N.: Assigning People to Roles in Software Projects. Softw. Pract. Exper., 34, 675--696 (2004)
16. Martínez, L., Rodríguez-Díaz, A., Licea, G., Castro, J.: Big Five Patterns for Software Engineering Roles using an ANFIS Learning Approach with RAMSET. Lect. Notes Comput. Sc., 6438, 428--439 (2010)
17. André, M., Baldoquín, M.G., Acuña, S.T.: Formal Model for Assigning Human Resources to Teams in Software Projects. Inform. Software Tech., 53, 259--275 (2011)
18. Standard Occupation Classification, Bureau of Labor Statistics, U.S. Department of Labor, http://www.bls.gov/soc
19. Occupational Information Network, Employment and Training Administration, U.S. Department of Labour, http://www.onetcenter.org
20. Tupes, E.C., Christal, R.E.: Recurrent Personality Factors Based on Trait Ratings. Technical Report ASD-TR-61-97, Lackland Air Force Base, Personnel Laboratory, Air Force Systems Division (1961)
21. McCrae, R.R., Costa, P.T.: NEO Inventories Professional Manual. PAR Inc., Florida (1992)
22. Pan, N., Hsaio, P., Chen, K.: A Study of Project Scheduling Optimization using Tabu Search Algorithm. Eng. Appl. Artif. Intel., 21, 1101--1112 (2008)
23. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. IEEE Trans. on Evol. Comput., 6, 181--197 (2002)