2012 IEEE 24th International Conference on Tools with Artificial Intelligence

A Novel Prototype Tool for Intelligent Software Project Scheduling and Staffing Enhanced with Personality Factors

Constantinos Stylianou
Department of Computer Science
University of Cyprus
Lefkosia, Cyprus
cstylianou@cs.ucy.ac.cy

Simos Gerasimou, Andreas S. Andreou
Department of Computer Engineering and Informatics
Cyprus University of Technology
Lemesos, Cyprus
{simos.gerasimou, andreas.andreou}@cut.ac.cy

Abstract—**Software project managers are often faced with challenges when trying to effectively staff and schedule projects. Incorrectly planning and estimating the execution of tasks frequently causes software projects to be delivered late and/or over budget, whereas not selecting the appropriate developers to carry out tasks may produce lower-quality, defective software products. To combat these challenges, this paper presents IntelliSPM – a tool aiming to support software project management activities consisting of several optimization mechanisms borrowed from the area of Computational Intelligence. The tool takes into account technical aspects but also significant human factors, which have been found to play a crucial role in software quality and developer productivity. The purpose of IntelliSPM is to offer suggestions to project managers containing a set of possible project schedules and staffing strategies that minimizes duration and maximizes resource usage. Several simulated and real-world projects were used during the validation process, with results showing that IntelliSPM is capable of providing that much-needed practical benefit to software companies to improve various aspects of development, such as performance and job satisfaction, whilst keeping within the general objectives and particular constraints of each software project.**

Keywords-Software project staffing and scheduling; genetic algorithms; particle swarm optimization; personality factors

I. Introduction

Software development companies have struggled for many years to adequately deal with the various issues complicating project management activities, such as the accuracy of planning estimates, the suitability of staffing methods and the sufficiency of control mechanisms. These difficulties contribute significantly to the fact that the majority of software projects are still either challenged (due to late delivery, overrunning costs and reduced functionalities), or considered to have failed (by being cancelled or delivered but never used) [1]. In addition, the ever-growing size and complexity of modern software systems has also added to the difficulty in managing today's software development projects and, in turn, has increased the rate of unsuccessful projects. These issues show a possible sign that existing software project management techniques may not be efficient or effective enough to provide practical benefits to software development companies and their project managers.

Researchers in the area of Software Engineering, therefore, have focused their attention on finding ways to provide software development companies with various tools and technologies, in the hope of mitigating the risks of a software project failing or being challenged. One particular area of interest involves the improvement of scheduling and staffing processes so that project managers are able to plan and assign developers to tasks in the best possible way. This area is considered extremely important because, on the one hand, inaccurate scheduling may cause significant delays in delivery and budget overruns and, on the other hand, improper staffing can lead to an undesired low level of quality in software products.

Despite many research efforts, scheduling and staffing of software projects remain a major challenge for software project managers due to the inherent complexity and uncertainty involved in these activities. Furthermore, management decisions regarding scheduling and staffing can suffer from the subjectivity of managers, whose criteria are often based on past experiences and perceptions, possibly leading to wrong decisions being made. Thus, the approach described in this paper targets helping software project managers to make scheduling and staffing decisions by employing optimization techniques from the field of Computational Intelligence. The techniques are incorporated into a dedicated decision support tool, namely IntelliSPM, developed using Matlab and Java, and use information concerning technical aspects (project duration and developer skill levels) as well as human aspects (personality traits of developers) of software development. The aim of the tool is to suggest optimal solutions to different scheduling and staffing queries based on various preference criteria. For example, certain project managers may want to evaluate whether their available developers have the adequate skills to undertake a project. Thus, IntelliSPM provides a way for software project managers to assess their resources, for instance, when deciding to submit a request for tender/proposal. Alternatively, a software project manager may wish to simply assign developers to tasks of a project whose schedule is known beforehand. Furthermore, the tool is able to notify managers whether or not the available developers are adequate enough to satisfy all the skills required by the tasks of a project and, thus, assist in deciding whether to improve skills further within the capacity of the available developers or consider the need to hire additional resources. Observations made after a brief run of experiments provide indications that IntelliSPM, as an intelligent project management tool for software development companies, has the potential to contribute significantly to mitigating the risks of software project failures. The results obtained through executions of the various optimization algorithms show that the tool has the ability to provide feasible and optimal scheduling and staffing strategies if and when these exist, but also has the ability to identify and notify project managers of possible problems, such as the lack of adequate resources, so that appropriate actions can be taken promptly to ensure that software products are delivered on time, within budget and with the appropriate levels of quality.

The remainder of the paper is organized as follows. Section II provides a summary of related research work carried out in the area of software project scheduling and staffing using Computational Intelligence techniques. The third section gives a detailed description of IntelliSPM and how it can be used by project managers. This section also gives an overview of the objective and constraint functions implemented in the tool's functionalities. Next, section IV describes the experiments carried out for validation purposes and discusses the results obtained from testing the optimization algorithms in real-world and simulated settings. Finally, the last section provides a synopsis and examines possible future work and enhancements to IntelliSPM.

II. Related Work

Research attempts related to software project scheduling and staffing have employed a number of different techniques and methods focusing mainly on providing solutions with respect to specific technical aspects of development projects, for example, duration, cost and effort. These techniques and methods are derived from various disciplines and are generally used as a means of optimizing goals in order to best schedule the execution of tasks and assign developers to work on a project. The most popular methods used are found in the area of Computational Intelligence since these methods are capable of modelling complex problems where mathematical analysis is limited. Also, they are more effective in handling the NP-hard nature of such problems and are able to explore problem search spaces more efficiently, thus providing solutions faster and with fewer computations [2, 3]. For these reasons, Computational Intelligence methods and, in particular, optimization algorithms have also been widely adopted in a broad range of areas of Software Engineering.

One of the most common Computational Intelligence techniques involves the use of evolutionary algorithms. For instance, single-objective genetic algorithms (GAs) were employed in various project scheduling attempts aiming to minimize software project duration and costs, taking into account developer reassignments [4], developer capabilities [5], and developer skills and global complexity [6, 7]. In addition, several attempts also targeted the improvement of software quality by minimizing product defects [8, 9]. A single-objective GA was also implemented in [10] as a means of minimizing project schedules and also maximizing utilization of developers by assigning them to tasks based on experience and team size. This approach to scheduling and staffing was also implemented using a single-objective particle swarm optimization (PSO) algorithm [11] and also developed into a multi-objective GA [12]. This type of algorithm is utilized mainly due to its ability to handle competing objectives, allowing for trade-offs between various problem criteria. For instance, the approach described in [13] attempts to simultaneously minimize project duration and cost and includes personnel risks that are assessed using information from previous assignments. Also, in [14] a framework for project scheduling and rescheduling was proposed taking into account the capabilities and skills of developers to help minimize cost and duration as well as to minimize disruption effects.

With respect to resource allocation, various attempts aim to assign software teams to work-packages in order to minimize project duration. Examples include a bin-packing representation and queuing theory approach within a single-objective GA, simulated annealing and hill-climbing [2, 15]. In addition, a multi-objective GA was employed to provide optimal solutions that minimize project overruns and duration, and at the same time maximize developer usage [16], later extended to incorporate developer specialization and task dependencies [17]. More recently, a cooperative co-evolutionary GA was proposed to assign developers and schedule the order of work-packages simultaneously [18].

Other Computational Intelligence methods used include constraint satisfaction to form teams based on different project management criteria [19, 20], fuzzy logic to handle time parameter uncertainties in scheduling [21], and multi-valued logic to rank available developers according to how suitable they are to certain tasks [22].

Apart from the use of Computational Intelligence and optimization techniques, several other approaches have been proposed regarding software project staffing, which concentrate more on the human factors present in software development practices and processes. Specifically, these attempts aim to build efficient and effective software teams by incorporating a number of non-technical aspects, such as personality types and traits [23] and capabilities [24], both of which are unique for each developer. Moreover, some of the proposed staffing methods also integrate Computational Intelligence techniques and take advantage of their benefits for assigning the most suitable developers to tasks. For example, in [25] the best allocation of developers is recommended to software project managers using an adaptive network-based fuzzy inference system (ANFIS) learning approach and utilizes developers' personality types to assign roles in Software Engineering teams through a role assignment methodology, RAMSET. An alternative method described in [26] uses a rule-based approach that aims to select the best software project roles (formed from sets of generic and technical competencies) matching the personality types of developers. A number of objective functions and constraints are then extracted from the rules and implemented in simulated annealing, hill-climbing and Tabu search algorithms for defining the optimal developer-role assignments. More recently, a multi-objective GA approach was employed for project staffing using several objective functions and constraints that evaluated the levels of technical skills of assigned developers as well as the level to which their personality traits were suited to the personality traits required by individual tasks [27]. IntelliSPM is largely based on this approach but also extends to integrate previous work of [10-12] by borrowing the encoding scheme, objective and constraint equations, described in section III.

One of the most important contributions of this paper is the design and implementation of a prototype intelligent tool for software project managers that integrates several optimization techniques for scheduling and staffing using both technical aspects (i.e., developer skill levels, duration and team size) and human factors (specifically, developer personality traits and required personality traits of tasks). This is an important aspect because developers are not interchangeable resources, which is an assumption encountered in several previous approaches. Instead, each developer is unique and possesses different technical skills with varying levels of skills and his/her own set of personality traits. Furthermore, each task is considered to have a set of personality traits that are more suited for its execution. What's more the integration of these techniques allows software project managers to perform scheduling and staffing in any way that suits their approach. For instance, they are able to perform staffing on a predefined schedule, or they can perform scheduling on a set of optimized staffing strategies and then choose the most preferable from the resulting schedules.

III. Description of IntelliSPM

In order to provide software project managers with solutions to the optimal scheduling of tasks and the most suitable developer assignments (staffing strategies), IntelliSPM makes use of single- and multi-objective GAs as well as a single-objective PSO algorithm. The implementation of these optimization algorithms provide the back-bone of two separate functionalities of IntelliSPM so that project managers have the option of using either one or both whenever they are required to carry out these activities. The two functionalities essentially contain an input process to obtain the required project information from the project manager, the customized optimization mechanism, and an output process for displaying various results to the project manager. In both cases, IntelliSPM requires information regarding project tasks, such as the effort required (in person-days) for each task, the dependencies between tasks, as well as the skills required by the activities carried out in each task. Innovatively, it also uses the personality traits of developers and the desired personality traits that are required to carry out tasks. Finally, the implemented algorithms have predefined parameter settings but project managers also have the option of changing them depending on how familiar they are with the optimization algorithms. The two functionalities are described in the subsequent sub-sections.

A. Schedule-based Optimal Staffing

The first functionality assigns developers to tasks and targets the best possible utilization of skills but also the allocation of the most suitable developers in terms of personality traits. This significant staffing criterion is used primarily as an objective in making sure that developers have the "right" personality to carry out a task and not just the appropriate technical skills. Furthermore, this functionality assumes that a project manager has already constructed the project's schedule and thus has the additional purpose of making certain that developers are allocated to tasks without assignment conflicts. This allows a project manager to use this functionality to answer important development questions, such as whether the available developers are capable of and suited to carrying out the project within a given time frame; which developers are utilized the most; and whether extra resources will be required for any tasks.

An important aspect of the approach is the possible competing nature of the optimization's objectives. Specifically, there may be cases where a developer has high levels of the skills required by a task, but is less suitable to carry out the task based on his/her personality traits, or vice-versa. For this reason, the functionality uses an implementation of the constrained Non-dominated Sorting Algorithm II (NSGA-II) [28], which is a multi-objective genetic algorithm that allows the generation of multiple solutions to a problem, each with varying objective values. Each individual is encoded using a binary string, whose length depends on the number of tasks in the project and the number of available developers, and each bit denotes whether or not a developer has been assigned to work on a specific task. This representation easily allows the evaluation of an individual, which uses the following objective functions adopted from [27]:

$$f_1 = \begin{array}{l}\text{max\_ratio}(skill\ levels\ of\ assigned\ devs) + \\ \text{avg\_ratio}(skill\ levels\ of\ assigned\ devs)\end{array} \tag{1}$$

$$f_2 = {}^{1}\!/_{number\ of\ assigned\ devs} \tag{2}$$

$$f_3 = \left| \begin{array}{c}desired\ personality\\ traits\ of\ each\ task\end{array} - \begin{array}{c}personality\ traits\\ of\ assigned\ devs\end{array} \right| \tag{3}$$

The objective function in (1) evaluates solutions with respect to the technical skill levels required by each task that are possessed by the assigned developers. The higher the skill levels, the more fit an individual will be. Likewise, the objective function shown in (2) is incorporated into the optimization approach so as to minimize the number of developers assigned to each task. By doing so, on one hand, resources are not "wasted" on tasks that can be executed with fewer developers and, on the other hand, the overhead in communication between developers is reduced. Finally, the objective function in (3) assesses the distance between the personality traits desired for the execution of a task and the personality traits possessed by the task's assigned developers. In this way, the more suited a developer's personality traits are to a task, the more likely he/she will be assigned to carry it out. An important note here is that in order to identify the personality traits of developers, a personality measuring instrument is required. In the current implementation of IntelliSPM, developers' personality traits are identified using the NEO-FFI-3 instrument [29] which scores developers based on the domains of the Five-Factor Model (FFM) – Neuroticism (N), Extraversion (E), Openness to Experience (O), Agreeableness (A), and Conscientiousness (C). In addition, for each task, the desired level – Low (L), Average (A), or High (H) – of each FFM domain is determined, therefore allowing the objective function to evaluate the level to which these preferred personality traits are satisfied by the personality scores of the assigned developers through a simple distance measure. For example, software analysis tasks call for high extraversion as a personality trait. Therefore, developers possessing high extraversion as a trait, as well as the required skills for the specific analysis task, will be more suited to carry it out.

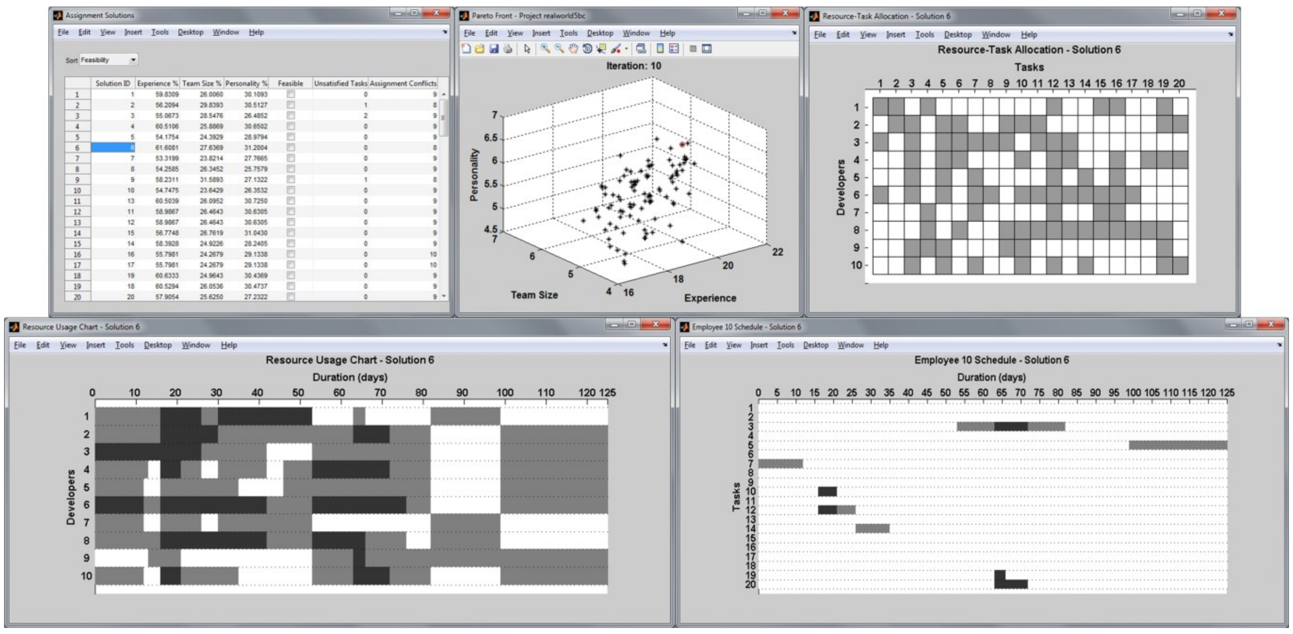
Since the purpose of this functionality is to assign developers to tasks using an existing project schedule, it is imperative that the algorithm penalizes solutions that assign developers on simultaneously executing tasks. Also, it is important that all skills required by tasks are satisfied by the assigned developers. These constraints are taken into consideration in the following functions:

$$c_1 = \frac{number\ of\ days\ with\ conflicts}{total\ number\ of\ working\ days} \tag{4}$$

$$c_2 = \frac{number\ of\ unsatisfied\ skills}{total\ number\ of\ project\ skills\ required} \tag{5}$$

The purpose of the constraint function shown in (4) is to help remove any instances where a developer is assigned to work on more than one task at any time during the project. If such an instance does occur, this is considered as a conflict in assignment, since an assumption is made that a developer is not able to suspend and then resume work on a task. Additionally, in order to make sure that the developers assigned to tasks actually possess the required skills, the constraint function in (5) is used to calculate the degree to which any skills are unsatisfied based on the current assignments represented in an individual. Together, these two constraints dictate whether an individual represents a feasible solution.

Once the multi-objective algorithm executes and terminates, several results are displayed in the form of tables and charts with which a software project manager is able to compare solutions and determine which staffing strategy is best to adopt. Fig. 1 shows an assortment illustrating several examples of these results. First, a plot of the final Pareto front is displayed showing the final non-dominated solutions in the respective objective space. The objective values are also presented in tabular form together with information regarding the feasibility of each solution. If a solution is infeasible, then number of skills unsatisfied and/or the number of developers that have assignment conflicts are also shown in the table. In order to view further results, a project manager simply selects a solution from the table and the corresponding resource usage chart is presented alongside the developer-task allocation chart. The resource usage chart helps project managers identify developer usage levels in the project, as well as the days in the project where a developer may have assignment conflicts. The developer-task allocation chart displays the tasks to which each developer is assigned, but also highlights tasks where the developers assigned to it do not possess any or all of the skills it requires to be completed. Furthermore, each developer's schedule is generated, allowing project managers to examine the exact points in the project where assignment conflicts have arisen and which tasks are affected by these conflicts. By providing


Assignment Solutions
File Edit View Insert Tools Desktop Window Help
Sort Feasibility
Pareto Front - Project realworld5bc
Iteration: 10
Personality
Team Size
Experience
Resource-Task Allocation - Solution 6
Tasks
Developers
Resource Usage Chart - Solution 6
Duration (days)
Developers
Employee 10 Schedule - Solution 6
Duration (days)
Tasks

	Solution ID	Experience %	Team Size %	Personality %	Feasible	Unsatisfied Tasks	Assignment Conflicts
1	1	59.8309	26.0060	30.1093	☐	0	9
2	2	56.2094	29.8393	30.5127	☐	1	8
3	3	55.0673	28.5476	26.4852	☐	2	9
4	4	60.5106	25.8869	30.6502	☐	0	9
5	5	54.1754	24.3929	28.9794	☐	0	9
6	6	61.6081	27.6369	31.2004	☐	0	8
7	7	53.3199	23.8214	27.7665	☐	0	9
8	8	54.2585	26.3452	25.7579	☐	0	9
9	9	58.2311	31.5893	27.1322	☐	1	8
10	10	54.7475	23.6429	26.3532	☐	0	9
11	13	60.5039	26.0952	30.7250	☐	0	9
12	11	58.9867	26.4643	30.6305	☐	0	9
13	12	58.9867	26.4643	30.6305	☐	0	9
14	15	56.7748	26.7619	31.0430	☐	0	9
15	14	58.3928	24.9226	28.2405	☐	0	9
16	16	55.7981	24.2679	29.1338	☐	0	10
17	17	55.7981	24.2679	29.1338	☐	0	10
18	19	60.6333	24.9643	30.4369	☐	0	9
19	18	60.5294	26.0536	30.4737	☐	0	9
20	20	57.9054	25.6250	27.2322	☐	0	9

Figure 1. Visual results displayed by IntelliSPM. Top row (left to right): Objective function and constraint values table, Pareto front plot, Resource-task allocation matrix. Bottom row (left to right): Resource usage chart, Individual developer schedule.

these results to project managers, they will be in a better position to compare and assess which staffing strategy is more favourable, depending on which objective carries more weight in the particular circumstance. In the case where solutions are infeasible, the results can provide valuable information, such as whether the schedule of tasks determined beforehand is reasonable or achievable given the current personnel or whether additional resources will be required in order to complete the project on time.

B. *Staff-based Optimal Project Scheduling*

The second functionality of IntelliSPM performs two basic operations: (1) provides a set of staffing strategies represented by multiple solutions of developer assignments, and (2) arranges tasks to form a project schedule using one of the staffing strategies chosen by a project manager. Specifically, the second functionality begins by suggesting a set of possible staffing strategies in a similar manner as in the previous functionality, but without the restriction of a project schedule. Thus, software project managers can use this to identify whether the available personnel is adequate enough to carry out a project based on skill levels and personality traits. Consequently, they can compare several feasible solutions and select the most preferable staffing strategy. For this operation, which uses an identical NSGA-II implementation, the same objectives and constraint functions are used as before in functionality *A*. However, it does not apply the function denoted in (4) regarding the assignment conflicts constraint (c_1), since in this stage it does not deal with the assignment of developers in specific time-slots.

If a project manager then wishes to create a project schedule using a specific staffing strategy, IntelliSPM uses optimization to provide a schedule with minimum project duration. Here, project managers are able to select between a GA approach and a PSO approach, which both make use of task durations and dependencies, as well as the staffing strategy selected. Both algorithms implemented use the objective function in (6) to devise a series of task executions so that the amount of idle days between tasks is kept to a minimum, thus generating a project schedule with the shortest duration possible.

$$f_4 = {}^{1}\!/_{1 + number\ of\ idle\ days} \tag{6}$$

Specifically, number of idle days represents the number of days between the starting time of a specific task and the finishing time of its predecessor. If a task depends on more than one task, then the predecessor that ends the latest is considered only for the computation of the number of idle days. Therefore, if no idle days exist, this will mean that there are no unnecessary delays between tasks (leading f_4 to a maximum value of 1). However, one of the assumptions made is that dependencies between tasks have a finish-to-start relationship, so that a task cannot start before any of its predecessors. In this case, for the task being evaluated, a value of 0 is assigned to f_4. It is also necessary to penalize solutions that violate such a constraint. The value of this violation is computed using (7), which evaluates the number of dependencies violated over the total number of dependencies present in the software project.

$$c_3 = \frac{number\ of\ violated\ dependencies}{total\ number\ of\ project\ dependencies} \tag{7}$$

Because this optimization entails the scheduling of tasks based on already-assigned developers, it is still necessary to ensure that none of them participating in the project is scheduled to work on more than one task at any time. Hence, the constraint function (4) is applied to evaluate if any of the tasks are scheduled in a way that cause assignment conflicts.

The optimization algorithm (GA or PSO, depending on the method chosen by the project manager) is executed in order to find a solution where the best scheduling of tasks, given the staffing strategy selected previously, generates the project with minimum duration. Once the execution terminates, IntelliSPM displays this solution to the project manager in the form of a Gantt chart. Alongside this, for each developer, his/her involvement in the project is displayed showing when he/she is scheduled to work on each task. If a non-feasible solution is found, then this will be indicated in the schedules, warning the project manager that a schedule is unachievable with the specific staffing strategy chosen. Moreover, this functionality allows managers to examine different staffing strategies and the subsequent schedules they generate, enabling comparisons to be made before selecting the most appropriate.

IV. Experiments and Results

A. *Design and Execution of Experiments*

Two types of experiments were carried out in order to assess the effectiveness and efficiency of IntelliSPM. Firstly, several hypothetical software projects of varying size and complexity were generated based on information provided by several project managers of SME software development companies, with respect to the type and nature of the software projects they usually undertake. This information included the number of tasks, the number of dependencies, in addition to the average number of required skills per task and average number of skills possessed per developer. For all project instances, the skill levels and personality traits of available developers were defined so as to represent the best-case and worst-case scenarios for generating optimal developer assignments. In the best-case scenarios, specific developers were chosen to possess both the highest skill levels and best personality traits for certain tasks, while other developers were less skilled and less suitable personality-wise. In contrast, in the worst-case scenarios, there was a direct conflict between the developers who possessed the highest skill levels and those who possessed the most suitable personality traits.

The second set of experiments considered a variety of real-world projects, one of which is selected as a case study in the following sub-section to discuss the results obtained after application of the proposed functionalities of IntelliSPM. This particular software project was provided by a local IT company and concerns the implementation of a vessel policies management system for a large insurance broker company. The project consisted of 4 management type tasks, 2 database design type tasks, 21 programming type tasks and 4 testing type tasks. Tasks belonging to the same type share the same skill requirements and desired personality traits. Additionally, four developers were available, each having unique personality traits and possessing a variety of skills. The information of this project is summarized in Table I.

B. *Discussion of Results*

To begin with, functionality *A* was performed in order to examine the algorithm's behaviour when attempting to produce solutions optimizing the assignment of developers using a predefined project schedule. The schedule, shown in Fig. 2(a), together with the other necessary project information was provided by the manager responsible for the project. Once inputted into IntelliSPM, the respective multi-objective GA used this information to guide the evolution of the population and evaluate the fitness and feasibility of individuals using the various objective and constraint functions. The final population of individuals consisted of a number of different solutions, from which several were selected for further analysis. The results of these selected solutions included, as mentioned previously, the developer-task allocation chart and the resource usage chart, which is presented in Fig. 2(b). This chart specifically displays the days when each developer is assigned throughout the duration of the project for the solution which scored the highest objective value for f_1, which favours developer skill levels over team size and personality traits. Interestingly, the specific solution contains several developer assignment conflicts (darkest-shaded time-boxes). This chart correctly indicates that there are not sufficient resources to carry out the software project given the current project schedule. This can be verified by examining the project data provided in the design of experiments. The project schedule consists of several tasks that are set to be executed simultaneously, but there are not enough developers possessing the necessary skills to fulfil this arrangement of tasks.

For assessing the effectiveness of the second functionality, the same project data was used, this time to first generate a set of optimal staffing strategies (without taking into consideration assignment conflicts) and then to generate an optimal sequence of task executions based on a selected staffing strategy so that the project's duration is minimized. The corresponding multi-objective GA was executed and, once again, the final population of individuals each represented a different staffing strategy favouring either one of the objectives. Following this, the optimal project schedule of several of these strategies was constructed using, in this case, the single-objective PSO approach. Figs. 2(c) and (d) depict the resulting project schedule and resource usage chart, respectively, of the staffing strategy that also scored the highest objective value regarding developer skill levels. With this functionality, there are no assignment conflicts existing, since the scheduling of tasks is carried out after the staffing strategy is selected, in which case, task starting days are repositioned (while developer assignments are kept fixed) so as to satisfy both the dependency violations and assignment conflicts constraints. As a result, the duration of the optimized project schedule in Fig. 2(c) is also longer than the project schedule constructed by the project manager in Fig. 2(a), so as to accommodate the scheduling of tasks. When these results were shown to the project manager, the manager agreed that the original project schedule was not consistent with the available developers and that it was necessary in the end to reschedule some of the tasks. Specifically, the original plan was devised in anticipation that the developers would actually work overtime (around 1½ times their normal workload). Unfortunately, this was too ambitious for the developers to follow, causing a much lower productivity rate than expected. Accordingly, the manger decided to prolong the schedule so that the product could be delivered, even if late.

C. *Validation of IntelliSPM*

An empirical validation was conducted with the help of project managers from local software development companies to examine IntelliSPM in terms of applicability, usability and scalability. Also, the validation process aimed to investigate whether this tool could be adopted by project managers in order to help them successfully perform the challenging software project management activities of scheduling and staffing.

The tool was initially presented to projects managers and the functionalities were explained and demonstrated using the data from the real-world project (Table I). The first

TABLE I. SUMMARY OF DATA FROM REAL-WORLD SOFTWARE PROJECT

(A) REQUIRED SKILLS PER TASK TYPE

	S1	S2	S3	S4
Management	✓	✓	✓	
DB Design	✓			
Programming		✓	✓	
Testing				✓

(B) SKILL LEVELS OF AVAILABLE DEVELOPERS

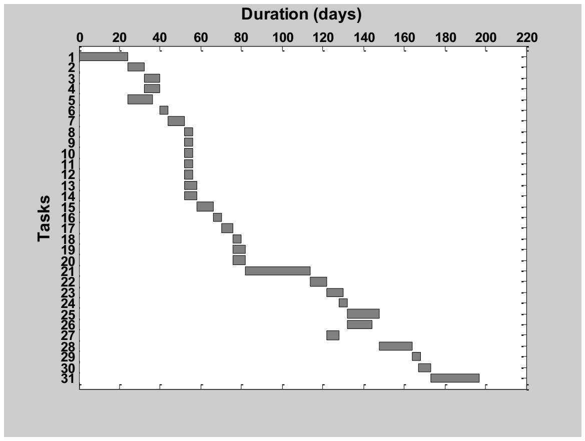
	S1	S2	S3	S4
Developer 1	0.9	1.0	1.0	0.0
Developer 2	0.2	0.8	0.7	0.0
Developer 3	0.5	0.6	0.5	0.0
Developer 4	0.0	0.0	0.0	0.8

(C) DESIRED PERSONALITY TRAITS FOR EACH TASK TYPE

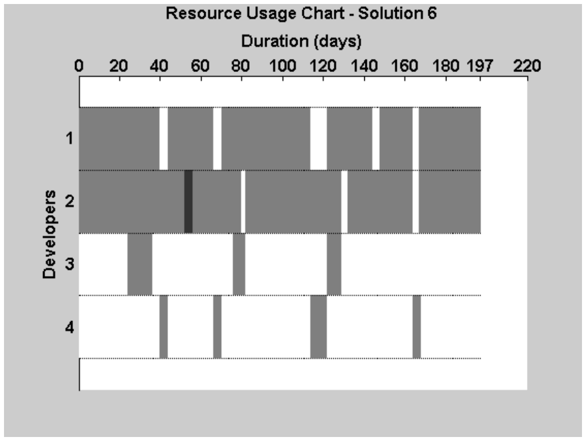
	N	E	O	A	C
Management	L	H	A	L	H
DB Design	L	L	H	L	L
Programming	L	L	A	L	A
Testing	A	A	L	L	L

(D) PERSONALITY TRAITS OF AVAILABLE DEVELOPERS

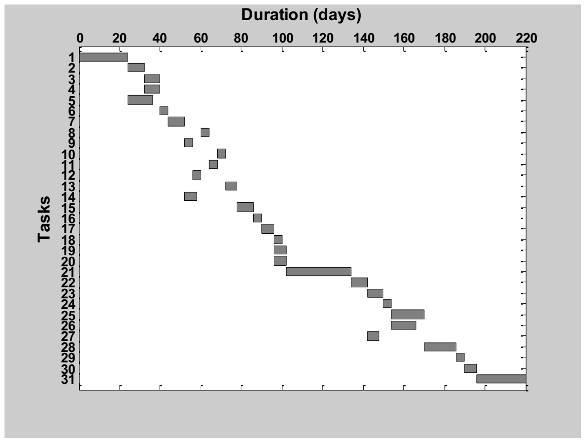
	N	E	O	A	C
Developer 1	A	H	H	H	L
Developer 2	A	A	A	L	H
Developer 3	L	L	A	L	A
Developer 4	H	A	A	H	H



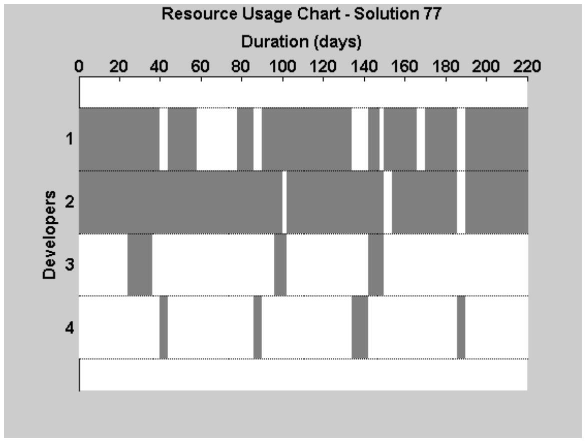
(a) Schedule constructed by project manager.



(b) Resource usage chart generated by IntelliSPM (functionality *A*).



(c) Schedule constructed by IntelliSPM (functionality *B*).



(d) Resource usage chart generated by IntelliSPM (functionality *B*).

Figure 2. Project schedules and resource allocation charts of real-world software project.

impression derived after spending some time with IntelliSPM was that the tool is quite easy to grasp and use, and also it offers the ability to display a variety of useful project and resource data, such as the developers' personality traits, which are not currently incorporated in any way in their own staffing approaches. Notably, positive comments were expressed regarding the helpful information presented in various figures, which allowed them to answer important, operational questions, such as whether the development team has the capacity to successfully carry out a project within a strict deadline or if additional developers should be employed, and whether any team members are over-/under-utilized. Additionally, the fact that they are presented with a range of possible staffing strategies was considered as a great advantage since this would allow the project managers to select the appropriate team formation based on their preference – skill levels, team size or personality traits.

Although the projects managers commented that IntelliSPM is a tool they would definitely like to employ for performing their activities, a request for more interaction between the user and the tool was expressed. For instance, one of the desired interactive functionalities suggested was to allow users to be able to make drag-and-drop actions on the resource usage chart so as to adjust the utilization of specific team members in order to balance their workload. Such comments for improvements and enhancements were noted so as to be addressed in the future.

V. Concluding Remarks

This paper proposes a tool that makes innovative use of several Computational Intelligence techniques in order to assist software project managers in completing the challenging activities of project scheduling and staffing. Specifically, IntelliSPM supports schedule-based optimal staffing using a multi-objective GA approach, in addition to staff-based optimal project scheduling where a hybrid optimization approach is employed using a multi-objective GA followed by a single-objective GA or PSO. The purpose of these functionalities is to allow project managers to generate the best staffing strategies in terms of developer skill levels and personality traits, as well the optimal project schedules with the shortest make-span. The inclusion of personality traits of developers and personality traits required by tasks as part of an objective function innovatively provides an extra dimension to scheduling and staffing, attempting to integrate human aspects that are known to critically impact software development. Certain constraints were also imposed to ensure that in both cases the solutions generated were in effect feasible based on several assumptions regarding task dependencies, developer availability and skill requirements. Experiments using simulated and real-world projects have demonstrated that the tool is capable of providing feasible and optimal solutions, but also an empirical validation taking into account expert opinions has shown that IntelliSPM can offer valuable assistance to software project managers in the form of scheduling and staffing strategies that can be examined and compared against each other.

Several future enhancements to IntelliSPM have been planned. Firstly, the implementation of additional optimization approaches, such as a multi-objective PSO algorithm, as well as the incorporation of additional objectives, such as project cost, can provide further benefits in the tool. Furthermore, offering the ability for project managers to have more control over the objectives he/she uses during optimization shall also be examined. Moreover, the results currently displayed will be improved, providing more first-level information, such as which developer has assignment conflicts and for how long, as well as enabling more user interaction. Finally, a web-based version of the tool is currently being designed, to allow software project managers to access project data and to carry out their scheduling and staffing activities remotely.

REFERENCES

[1] Standish Group, Standish Group CHAOS Report. Boston, MA: Standish Group International Inc., 2010.

[2] G. Antoniol, M. Di Penta and M. Harman, “Search-based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project,” Proc. 21st IEEE International Conference on Software Maintenance, IEEE CS Press, Sep. 2005, pp. 240-249.

[3] N. Pan, P. Hsaio and K. Chen, “A Study of Project Scheduling Optimization Using Tabu Search Algorithm,” Eng. Appl. Artif. Intel., vol. 21, no. 7, Oct. 2008, pp. 1101-1112.

[4] C. K. Chang, H. Jiang, Y. Di, D. Zhu and Y. Ge, “Time-Line Based Model for Software Project Scheduling with Genetic Algorithms,” Inform. Software Tech., vol. 50, no. 11, Oct. 2008, pp. 1142-1154.

[5] Y. Ge and C. K. Chang, “Capability-based Project Scheduling with Genetic Algorithms,” Proc. 2006 International Conference on Computational Intelligence for Modelling, Control and Automation, IEEE CS Press, Nov.-Dec. 2006, pp. 161.

[6] E. Alba and J. F. Chicano, “Management of Software Projects with GAs,” Proc. 6th Metaheuristics International Conference, Aug. 2005, pp. 13-18.

[7] E. Alba and J. F. Chicano, “Software Project Management with GAs,” Inf. Sci., vol. 177, no. 11, Jun. 2007, pp. 2380-2401.

[8] J. Duggan, J. Byrne and G. J. Lyons, “A Task Allocation Optimizer for Software Construction,” IEEE Softw., vol. 21, no. 3, May/Jun. 2004, pp. 76-82.

[9] T. Hanne and S. Nickel, “A Multiobjective Evolutionary Algorithm for Scheduling and Inspection Planning in Software Development Projects,” Eur. J. Oper. Res., vol. 167, no. 3, Dec. 2005, pp. 663-678.

[10] C. Stylianou and A. S. Andreou, “Intelligent Software Project Scheduling and Team Staffing with Genetic Algorithms,” Proc. 7th IFIP Conference on Artificial Intelligence Applications and Innovations, Springer, Sep. 2011, pp. 169-178.

[11] S. Gerasimou, C. Stylianou and A. S. Andreou, “An Investigation of Optimal Project Scheduling and Team Staffing in Software Development using Particle Swarm Optimization,” Proc. 14th International Conference on Enterprise Information Systems, SciTePress, Jun.-Jul. 2012, p. 88.

[12] C. Stylianou and A. S. Andreou, “A Multi-Objective Genetic Algorithm for Intelligent Software Project Scheduling and Team Staffing,” Intelligent Decision Technologies, 2012, in press.

[13] H. Jiang, C. K. Chang, J. Xia and S. Cheng, “A History-based Automatic Scheduling Model for Personnel Risk Management,” Proc. 31st Annual International Computer Software and Applications Conference, IEEE CS Press, Jul. 2007, pp. 361-366.

[14] Y. Ge, “Software Project Rescheduling with Genetic Algorithms,” Proc. 2009 International Conference on Artificial Intelligence and Computational Intelligence, IEEE CS Press, Nov. 2009, pp. 439-443.

[15] G. Antoniol, M. Di Penta and M. Harman, “Search-Based Techniques for Optimizing Software Project Resource Allocation,” Proc. 2004 Genetic and Evolutionary Computation Conference, Springer, Jun. 2004, pp. 1425-1426.

[16] S. Gueorguiev, M. Harman and G. Antoniol, “Software Project Planning for Robustness and Completion Time in the Presence of Uncertainty using Multi-Objective Search-based Software Engineering,” Proc. 11th Annual Conference on Genetic and Evolutionary Computation, ACM, Jul. 2009, pp. 1673-1680.

[17] M. Di Penta, M. Harman and G. Antoniol, “The Use of Search-based Optimization Techniques to Schedule and Staff Software Projects: An Approach and an Empirical Study,” Software Pract. Exper., vol. 41, no. 5, Apr. 2011, pp. 495-519.

[18] J. Ren, M. Harman and M. Di Penta, “Cooperative Co-evolutionary Optimization of Software Project Staff Assignments and Job Scheduling,” Proc. 3rd International Symposium on Search-based Software Engineering, Springer, Sep. 2011, pp. 127-141.

[19] A. Barreto, M. Barros and C. Werner, “Staffing a Software Project: A Constraint Satisfaction Approach,” ACM SIGSOFT,” vol. 30, no. 4, Jul. 2005, pp. 1-5.

[20] A. Barreto, M. Barros and C. Werner, “Staffing a Software Project: A Constraint Satisfaction and Optimization Based Approach,” Comput. Oper. Res., vol. 35, no.10, Oct. 2008, pp. 3073-3089.

[21] M. Hapke, A. Jaszkiewicz and R. Slowinski, “Fuzzy Project Scheduling System for Software Development,” Fuzzy Set Syst., vol. 67, no, 1, Oct. 1994, pp. 101-117.

[22] D. A. Callegari and R. M. Bastos, “A Multi-Criteria Resource Selection Method for Software Projects using Fuzzy Logic,” Proc. 11th International Conference on Enterprise Information Systems, Springer, May 2009, pp. 376-388.

[23] R. H. Rutherfoord, “Using Personality Inventories to Help Form Teams for Software Engineering Class Projects,” Proc. 6th Annual Conference on Innovation and Technology in Computer Science Education, ACM, Jun. 2001, pp. 73-76.

[24] S. T. Acuña and N. Juristo, “Assigning People to Roles in Software Projects,” Softw. Pract. Exper., vol. 34, no. 7, Jun. 2004, pp. 675-696.

[25] L. Martínez, A. Rodríguez-Díaz, G. Licea and J. Castro, “Big Five Patterns for Software Engineering Roles using an ANFIS Learning Approach with RAMSET,” Proc. 9th Mexican International Conference on Artificial Intelligence, Springer, Nov. 2010, pp. 428-439.

[26] M. André, M. G. Baldoquín and S. T. Acuña, “Formal Model for Assigning Human Resources to Teams in Software Projects,” Inform. Software Tech., vol. 53, no. 3, Mar. 2011, pp. 259-275.

[27] C. Stylianou and A. S. Andreou, “A Multi-Objective Genetic Algorithm for Software Development Team Staffing based on Personality Types,” Proc. 8th IFIP Conference on Artificial Intelligence Applications and Innovations, Springer, Sep. 2012, pp. 37-47.

[28] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, “A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II,” IEEE Trans. Evol. Comput., vol. 6, no. 2, Apr. 2002, pp. 181-197.

[29] R. R. McCrae and P. T. Costa, Jr., NEO Inventories: Professional Manual. Lutz, FL: Psychological Assessment Resources Inc., 2010.