Computers & Industrial Engineering 56 (2009) 1333–1339

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A systematic approach for resource allocation in software projects

Luis Daniel Otero [a], Grisselle Centeno [a,*], Alex J. Ruiz-Torres [b], Carlos E. Otero [c]

[a] Department of Industrial and Management Systems Engineering, University of South Florida, 4202 E. Fowler Avenue ENB 118, Tampa, FL 33620-5350, United States
[b] Information and Decision Sciences Department, College of Business Administration, Room 205, El Paso, TX 79968, United States
[c] Department of Electrical and Computer Engineering, Florida Institute of Technology, 150 W. University Boulevard, Melbourne, FL 32901, United States

ARTICLE INFO

Article history:
Received 1 March 2007
Received in revised form 3 June 2008
Accepted 6 August 2008
Available online 12 August 2008

Keywords:
Staff allocation
Personnel assignment
Resource allocation
Learning curve
Training time
Software projects

ABSTRACT

The completion of reliable software products within their expected time frame represents a major problem for companies that develop software applications. Today, the software industry continues to struggle with delivering products in a timely manner. A major cause for delays is the training time required for engineers and other personnel to acquire the necessary skills to complete software tasks. Therefore, it is important to develop systematic personnel assignment processes that consider complete skill sets of candidates to provide solutions that reduce training time. This paper presents a novel methodology to assign resources to tasks when optimum skill sets are not available. The methodology takes into account existing capabilities of candidates, required levels of expertise, and priorities of required skills for the task. A sample case is used to show the model capabilities, and the results are compared with the current resource assignment approach.

Published by Elsevier Ltd.

1. Introduction

The development of high-quality products continues to be a major struggle to software companies. This is evidenced by Linberg (1999), who stated that only about 16% of software projects are on time and within budget. In addition, the U.S. Department of Defense (DOD) spent nearly 8 billion dollars in 2004 to rework software because of quality-related issues (U.S. General Accounting Office Report, 2004).

Major contributors to the outcome of software projects are personnel assignment decisions (Abdel-Hamid, 1989; Acuña, Juristo, & Moreno, 2006; Tsai, Moskowitz, & Lee, 2003). An expected benefit from improving resource allocation processes is decreased tasks' durations, which will allow companies to be more productive (Kolisch, 1999).

Despite all the research and advances in the field, managing software personnel remains a very complicated endeavor. A major contributor to this complexity is the increased demand for specialized individual skills in the workforce, which results from high turnover rates and the fast pace at which new technologies and techniques are being developed. As a result of higher demands, candidates with exact required skills to work tasks are usually not available. Due to the lack of proper methods to assess personnel capabilities, decision makers are forced to assign resources to tasks based on subjective measures only. This results in excess training times that significantly affect the schedule of projects. Therefore, further studies of processes and techniques for personnel management are necessary to provide better solutions in terms of quality, cost, and schedule.

This paper proposes the Best-Fitted Resource (BFR) methodology, which is a systematic approach to determine the fit (i.e., suitability) between the complete set of skills available from candidates and skills required for tasks. This way, the model can be used effectively to assign resources to tasks even when the most desirable skills are not available from the existing workforce. The methodology considers capabilities of candidates in required skills, levels of expertise required, and relative priorities of required skills for tasks.

This paper is divided into seven sections. Section 2 describes literature related to resource allocation in software projects. Section 3 describes the current resource allocation process. Section 4 explains the proposed BFR methodology for personnel assignment. In Section 5, the methodology is expanded for the case of multiple tasks. Section 6 describes a survey analysis that was conducted to test the hypothesis that personnel assignment decisions are mostly based on the perceived capability of candidates in one main skill, without carefully considering capabilities of candidates in other skills. The results from the survey analysis were used to describe the current personnel assignment process. Section 7 presents the contributions and areas for future research.

2. Related work

Research in human-resource allocation techniques for software development is very limited. There are specific aspects that make

* Corresponding author. Tel.: +1 813 974 5587; fax: +1 813 974 5953.
E-mail addresses: ldotero@mail.usf.edu (L.D. Otero), gcenteno@eng.usf.edu (G. Centeno), aruiztor@utep.edu (A.J. Ruiz-Torres), cotero@fit.edu (C.E. Otero).

0360-8352/$ - see front matter Published by Elsevier Ltd.
doi:10.1016/j.cie.2008.08.002

the assignment of software engineers to tasks a unique situation (Kan, 1995). One specific characteristic in software projects is that tasks are usually not expedited by adding extra resources (Plekhanova, 1999), since there is significant overhead training and communication time required for resources to get familiarized with tasks (Abdel-Hamid, 1989). Another specific characteristic in software projects is that estimates of tasks' durations are very imprecise, since they depend on a set of factors that are hard to estimate such as capabilities of developers. (Plekhanova, 1999).

Acuña et al. (2006) stated that software managers typically make assignments based on "their experience, heuristic knowledge, subjective perception, and instinct". Duggan, Byrne, and Lyons (2004) developed a multi-objective optimization model for software task allocation based on genetic algorithms. Competencies of developers were modeled using a categorical variable with five levels. Competency levels were associated with expected productivities per day, and expected numbers of defects per unit of productivity. Other studies have developed procedures for allocating personnel to software tasks based on the assessment of behavioral competencies (Acuña & Juristo, 2004; Acuña et al., 2006).

Tsai et al. (2003) proposed selecting resources using the CRD method and the Taguchi's parameter design approach. The authors used the CRD method because it focuses on resource scheduling rather than activity scheduling to represent human-resource workflow and tasks' precedence. The Taguchi's parameter design method was used to optimize the selection of engineers for tasks under dynamic and stochastic conditions. The Taguchi's parameter design approach is based on the concept of target value (Roy, 2001). Deviations from target values result in additional costs. Deviations are attributed to controllable and uncontrollable factors. The aim is to achieve optimal levels of controllable factors while minimizing the variation caused by nuisance (uncontrollable) factors (Ross, 1996).

In Tsai et al. (2003), skill levels of resources were estimated as average numbers of software lines of code (SLOC) per day. The authors commented on the importance of including in their model stochastic factors affecting the selection of resources. Specifically, emphasis was placed on the stochastic behavior of tasks complexities, since they are very difficult to measure or even estimate, causing most of the variability of calculated project completion times.

Another methodology used to evaluate staffing alternatives is the Analytical Hierarchy Process (AHP). Ho-Leung (2001) used AHP to tackle the problem of human-resource substitution, considering several organization, client, and application attributes. The proposed model did not consider relationships between known and required skills. The author mentioned the importance of developing faster methods for human-resource substitution.

Antoniol, Cimitile, Di Lucca, Di Penta (2004) used queuing theory and stochastic simulation to study staffing needs for software maintenance problems. The authors proposed modeling maintenance processes as first-in-first-out (FIFO) queues in which incoming requests are passed to first available maintenance teams. Since most of the process was standardized, with some portion of it being automatic, different programming skills and experiences were not included in the decision making process.

Shaikh (1998) presented a model for project staff reallocation. The objective of the model was to increase the probability of projects finishing on time by avoiding delays caused by lack of needed workforce at scheduled start dates. The model, based on the Lead Time concept from Inventory Theory, first computes workforce sizes for each month from beginning to end of projects. Second, it trims peak workforces and recalculates staffing needs for the complete duration of projects, ensuring that due dates are met. Trimming peak workforces is accomplished by starting projects before their expected start date. This results in a decreased delay caused by unavailability of engineers to work the tasks that were started ahead of time. The disadvantage of this methodology is that companies take the risk of not being awarded the project; therefore, not getting compensated for the money and effort invested for the "early start" periods.

Abdel-Hamid (1989) proposed a model to study the dynamic implications of staffing policies to the cost and schedule of projects. The author defined staffing policies as the number of workers to be added during the different phases of a software development project. The author stated that adding new employees to projects causes delays that significantly impact productivity. Since levels of experience of workers are important in terms of training time estimation, hired workforce was classified as either "Newly Hired Workforce" or "Experienced Workforce". Each type of workforce was associated with a training delay based on training and communication overhead times. The author defined training time as the amount of time that it takes new employees to get familiarized with projects and come up to speed with technical skills, as well as the time that it takes experienced workers to train new employees. Based on the delays incurred by adding new employees, the author defined a weight factor called "Willingness to Change Workforce" (WCWF). This factor was used to determine changes to staffing policies and workforce levels needed. The author also commented on the high turnover rates and its serious implications to the management of software projects.

From the literature reviewed, it is evident that there is much room for improved personnel assignment methodologies in software projects. The literature shows that the most common measure of the ability of a software developer is an estimated value of SLOC per day. Furthermore, this estimated SLOC value is usually a function of whether the developer is considered experienced or not in a particular skill. To the best of our knowledge, a methodology that considers complete set of capabilities of candidates, levels of skills required, and priorities of required skills for tasks is nonexistent in the software development literature.

3. Resource allocation process

The process for resource allocation can be described as a two-step process. First, decision makers obtain information from projects' proposals on the number and types of software engineers required. The number of required engineers is a function of the necessary skills and the complexities of projects. Typically, complexities of projects are described as estimated numbers of SLOC. The idea is that SLOC values can be used as predictors of parameters such as effort and ease of maintenance. Expected productivities of resources are also estimated using SLOC values. Even though using them produce quick results, SLOC values are regarded as highly inaccurate measures for estimating important project parameters.

Second, decision makers query employee databases to assign available engineers to software tasks. The assignments are straight forward if perfect matches exist between tasks and available engineers. However, in most situations available resources do not possess the complete set of skills required for tasks. In these cases, managers must decide to either outsource or hire new staff, or assign engineers from the set of available ones. The latter option is usually preferred because it prepares engineers for similar tasks in the future. Furthermore, outsourcing can be very expensive and hiring processes can take considerable amounts of time. Today, most assignments of resources to software tasks are based on managers' judgment and not on methodologies to assist them to make better decisions (Acuña et al., 2006).

4. Best-Fitted Resource (BFR) methodology

The time that resources spend learning or improving skills in order to reach required levels of expertise is a function of the levels of knowledge of resources in other related skills. Training time is con-

sidered idle time in terms of productivity, as effort (i.e., man-hours) is not spent working on tasks and instead it is spent getting ready to work on tasks. Training time is considered one of the causes for Brooks' Law, which states that adding manpower to a late software project delays the project even more (Brooks, 1975). That is, reducing training time becomes even more critical at late stages of projects. Therefore, the importance of correctly matching resources with tasks increases as projects progress through their phases.

The BFR methodology incorporates skill-relationship tables to describe how previous knowledge on various related skills contributes to learning required skills. This approach integrates task complexities, expected use of skills, and capabilities of resources. The following notation is used in the BFR methodology:

H set of all skills
$H(t)$ set of required skills for task t
e_{jt} expected use of skill j on task t. Assigned values range from 0 to 1
1 = highly used
0 = not used
c_{jt} complexity of skill j on task t. Assigned values range from 0 to 1
1 = high level of complexity on the task for this skill
0 = no complexity
s_{jt} significance of skill j on task t = c_{jp} e_{jp}. Calculated values range from 0 to 1
1 = critically important
0 = not important
r_{jk} relationship between the level of knowledge of known skill j and the level of knowledge for required skill k. Assigned values range from 0 to 1
1 = strong relationship
0 = no relationship
if j = k and the level of knowledge of both j and k are equal, then $r_{jj} = 1$
l_{yj} level of knowledge of resource y for skill j. Assigned values range from 0 to 1
1 = expert
0 = no knowledge
b_{yj} relationship between resource y and its known skills and skill j calculated values range from 0 to 1
1 = strong relationship; resource y is an expert in the skill or a highly related skill
0 = no knowledge in the skill or in a related skill
f_{yt} fit of resource y to task t. Calculated values range from 0 to 1
1 = strong fit
0 = resource is a bad fit for the task

There are four steps in the BFR methodology. The objective of each step is to develop tabular information to establish a process for selecting the most qualified resources for tasks. The resulting tables for each of the steps are:

- Task Required Skills (TRS)
- Skill Relationships (SR)
- Resources' Skill Set (RSS)
- Best-Fitted Resource (BFR)

The following subsections discuss in detail the steps for the BFR methodology. The sample scenario shown in Fig. 1 is used to demonstrate the capability of the methodology.

4.1. Task required skills – TRS table

The first step of the process is to define levels of skills required for a task. Each skill level is specified in terms of its expected use (e_{jt}) and complexity (c_{jt}). For simplicity and flexibility, both e_{jt} and c_{jt} are defined subjectively by decision makers with discrete values ranging from 0 to 1. Default levels for e_{jt} and c_{jt} are as follows:

For e_{jt}, default values are:

- Little use = 0.3
- Significant use = 0.7
- Extensive use = 1.0

For c_{jt}, default values are:

- Simple = 0.2
- Complex = 0.5
- Very challenging = 1.0

The significance of each required skill for a task, represented as s_{jt}, is calculated as the product of the expected use index times the complexity index.

Table 1 shows a TRS Table for the sample problem. For example, assume that the XYZ Hardware will be used to develop a computer program that receives status information from it. The manual that explains how the equipment works is relatively simple, therefore an index of 0.2 is assigned for complexity. Since the XYZ Hardware equipment will be used only to receive input status, the expected use is considered little. Therefore, an index of 0.3 is assigned for expected use. The s_{jt} index calculated is 0.2 * 0.3 = 0.06, which indicates that knowledge of the functionality of the hardware equipment is not an essential skill, but rather a preferred one.

4.2. Skill relationships – SR Table

For proper assessment of the capabilities of resources, it is important to establish a measure to describe how different skills may help decrease the learning time to become proficient in other skills. If resources do not possess experience in required skills, perhaps they are proficient in other skills that are similar to the required skills and can accelerate the learning process. For example, knowledge in the C++ programming language can decrease, to some extent, the training time to become proficient in the C# programming language because they are both object-oriented languages and have a somewhat similar syntax. An SR Table helps to determine a more complete assessment of the capabilities of resources. The objective of the SR Table is to show the learning curve relationships (r_{jk}) between known and unknown skills. Decision makers subjectively define learning curve relationships with discrete values ranging from 0 to 1. Default values for r_{jk} values are as follows:

- No relationship = 0
- Weak = 0.2
- Intermediate = 0.5
- Strong = 1.0

Table 2 shows the SR Table for the sample problem. It can be seen that high knowledge on the Java programming language significantly reduces the time required to learn C++, since the syntax of both programming languages is very similar. This is represented in the SR Table with an r_{jk} index showing a strong relationship (i.e., value of 1) between the two programming languages.

4.3. Resource's Skill Set (RSS) Table

The third step of the process is to prepare a tabular representation of the knowledge of available resources with discrete values

A software engineer with the following **requirements** is needed:

- **Expert knowledge with the C++ programming language.**
- **Good knowledge of the Windows NT and 2000 operating systems.**
- **Experience with programming the XYZ Hardware equipment is preferable**.

There are six software engineers available with particular skill levels. Please select the best-fitted engineer for the task from the table below:

Known Skills / Resources	C++ Programming Language	Windows NT	Windows 2000	XYZ Hardware	Visual Basic Programming Language	Java Programming Language
Engineer_1	Low	Intermediate	Intermediate	High	Intermediate	High
Engineer_2	Intermediate	Low	Low	0	Low	Low
Engineer_3	Intermediate	Low	Intermediate	Intermediate	Low	Low
Engineer_4	Low	0	Low	0	Low	Low
Engineer_5	High	Low	Low	0	0	0
Engineer_6	Low	High	Intermediate	0	High	0

Fig. 1. Sample resource allocation scenario.

Table 1
TRS Table

Required skills	e_{jt}	c_{jt}	s_{jt}
C++	1	1	1
Windows NT	0.7	0.5	0.35
Windows 2000	0.7	0.5	0.35
XYZ Hardware	0.3	0.2	0.06

Table 2
Sample SR Table showing LC indexes

	C++	Windows NT	Windows 2000	XYZ Hardware	VB	Java
C++	1	0	0	0	0.5	1
Windows NT	0	1	1	0	0	0
Windows 2000	0	1	1	0	0	0
XYZ Hardware	0	0	0	1	0	0
VB	0.2	0	0	0	1	0.2
Java	1	0	0	0	0.2	1

ranging from 0 to 1. These values are also subjectively defined by decision makers. Default values are as follows:

- No knowledge = 0
- Low = 0.2
- Intermediate = 0.5
- High = 1

Table 3 shows the RSS table for the sample problem. This table shows the levels of knowledge of six available engineers in different skills.

4.4. BFR Table

The fourth step of the process is to set up a BFR table to determine the suitability of available resources with the skills required for a task. The most suitable resource will most likely take the least amount of training time. For a required skill k, there are two factors considered for each resource y. The first factor is the level of knowledge of resource y in the required skill, denoted by l_{yk}. The second factor is the level of knowledge of resource y in all other possible skills and their relationship to the desired skill, obtained by multiplying $l_{yh} * r_{hk}$, where $h \in H$. The capability of a resource in a required skill, which is an indicator of the expected training time, is defined as:

$$B_{yk} = \mathrm{Max}_{h \in H}[l_{yh} * r_{hk}] \quad (1)$$

Table 4 shows the capabilities of the available resources in each of the required skills from the sample scenario, taking into consideration the relationships between skills. For example, the calculation of the knowledge of Engineer_1 in C++ is as follows. Since the level of knowledge of Engineer_1 in C++ is 0.2 (i.e., low), and $r_{jj} = 1$, the capability of this resource in C++ based on his/her knowledge of C++ is 0.2. Now, since knowledge of Windows NT, Windows 2000, or XYZ Hardware has no relationship with C++, then $r_{hj} = 0$, which means that knowledge in any of these three skills do not contribute

Table 3
RSS Table showing *lyj* values

Resources	Known skills					
	C++	Windows NT	Windows 2000	XYZ Hardware	Visual Basic	Java
Engineer_1	0.2	0.5	0.5	1	0.5	1
Engineer_2	0.5	0.2	0.2	0	0.2	0.2
Engineer_3	0.5	0.2	0.5	0.5	0.2	0.2
Engineer_4	0.2	0	0.2	0	0.2	0.2
Engineer_5	1	0.2	0.2	0	0	0
Engineer_6	0.2	1	0.5	0	1	0

Table 4
Capabilities of resources in the required skills

	C++	Windows NT	Windows 2000	XYZ Hardware	Visual Basic	Java	b_{yj}
Engineer_1							
C++	0.2	0	0	0	0.1	1	1
Windows NT	0	0.2	0.5	0	0	0	0.5
Windows 2000	0	0.2	0.5	0	0	0	0.5
XYZ Hardware	0	0	0	1	0	0	1
Visual Basic	0.04	0	0	0	0.5	0.2	0.5
Java	0.2	0	0	0	0.1	1	1
Engineer_2							
C++	0.5	0	0	0	0.04	0.2	0.5
Windows NT	0	0.2	0.2	0	0	0	0.2
Windows 2000	0	0.2	0.2	0	0	0	0.2
XYZ Hardware	0	0	0	0	0	0	0
Visual Basic	0.1	0	0	0	0.2	0.04	0.2
Java	0.5	0	0	0	0.04	0.2	0.5
Engineer_3							
C++	0.5	0	0	0	0.04	0.2	0.5
Windows NT	0	0.2	0.5	0	0	0	0.5
Windows 2000	0	0.2	0.5	0	0	0	0.5
XYZ Hardware	0	0	0	0.5	0	0	0.5
Visual Basic	0.1	0	0	0	0.2	0.04	0.2
Java	0.5	0	0	0	0.04	0.2	0.5
Engineer_4							
C++	0.2	0	0	0	0.04	0.2	0.2
Windows NT	0	0	0.2	0	0	0	0.2
Windows 2000	0	0	0.2	0	0	0	0.2
XYZ Hardware	0	0	0	0	0	0	0
Visual Basic	0.04	0	0	0	0.2	0.04	0.2
Java	0.2	0	0	0	0.04	0.2	0.2
Engineer_5							
C++	1	0	0	0	0	0	1
Windows NT	0	0.2	0.2	0	0	0	0.2
Windows 2000	0	0.2	0.2	0	0	0	0.2
XYZ Hardware	0	0	0	0	0	0	0
Visual Basic	0.5	0	0	0	0	0	0.5
Java	1	0	0	0	0	0	1
Engineer_6							
C++	0.2	0	0	0	0.2	0	0.2
Windows NT	0	1	0.5	0	0	0	1
Windows 2000	0	1	0.5	0	0	0	1
XYZ Hardware	0	0	0	0	0	0	0
Visual Basic	0.04	0	0	0	1	0	1
Java	0.2	0	0	0	0.2	0	0.2

to the resource's capability in C++. The relationship between Visual Basic and C++ is low, therefore $r_{hj} = 0.2$. Since the level of knowledge of the resource in Visual Basic is Intermediate, then $l_{yh} = 0.5$. Therefore, the capability of Engineer_1 in C++ due to his/her knowledge in Visual Basic is $0.2 * 0.5 = 0.1$. The relationship between Java and C++ is high, therefore $r_{hj} = 1$. Since the level of knowledge of the resource in Java is High, then $l_{yh} = 1$. Therefore, the capability of Engineer_1 in C++ due to his/her knowledge in Java is 1.0. Even though the resource's experience with C++ is low, his/her expert knowledge in Java will greatly decrease the training time in C++.

The suitability of resource y with task t with skill set $H(t)$ is determined by the sum of the products of the significance of skill j for task t times the capability of resource y in skill j as

$$f_{yt} = \sum_{j \in H(t)} s_{jt} * b_{yj}, \quad \forall\, y \tag{2}$$

Table 5 shows the results from the sample scenario. Based on the BFR method, Engineer_1 is the best fitted followed by Engineer_5.

5. Resource allocation to multiple tasks

The sample scenario shown in Fig. 1 dealt with a resource allocation problem of a single task. In personnel assignment situations with multiple tasks, the BFR methodology could be extended to provide a solution that will maximize the suitability of resources with tasks. The first step is to determine the suitability of resource y with each task t using the following equation:

$$f_{yt} = \sum_{j \in H(t)} s_{jt} * b_{yj}, \quad \forall\, y \;\; \forall\, t \tag{3}$$

The next step is to formulate the problem as a linear programming model based on values of f_{yt}. It is assumed that the number of available resources is greater than or equal to the number of tasks. The constraints are that a resource can be assigned to at most one task and each task must be assigned to a resource.

The decision variable is defined as

$$X_{yt} = \left\{ \begin{array}{ll} 1 & \text{if resource } y \text{ is assigned to task } t \\ 0 & \text{otherwise} \end{array} \right\} \tag{4}$$

The corresponding objective function is

$$\text{Maximize } Z = \sum_{t=1}^{T} \sum_{y=1}^{Y} f_{yt} * X_{yt} \tag{5}$$

Since f_{yt} describes how suitable is resource y to work task t, the objective function states that the best assignment policy is the one that maximizes the sum of the f_{yt} coefficients. The first constraint states that a resource may be assigned to at most one task.

Table 5
Results

	S_{jt}	b_{yj}	$S_{jt}^{*} b_{yj}$
Engineer_1			
C++	0.981	1	0.981
Windows NT	0.3	0.5	0.15
Windows 2000	0.15	0.5	0.075
XYZ Hardware	0.05	1	0.05
		f_{yt} = 1.256	
Engineer_2			
C++	0.981	0.5	0.4905
Windows NT	0.3	0.2	0.06
Windows 2000	0.15	0.2	0.03
XYZ Hardware	0.05	0	0
		f_{yt} = 0.5805	
Engineer_3			
C++	0.981	0.5	0.4905
Windows NT	0.3	0.5	0.15
Windows 2000	0.15	0.5	0.075
XYZ Hardware	0.05	0.5	0.025
		f_{yt} = 0.7405	
Engineer_4			
C++	0.981	0.2	0.1962
Windows NT	0.3	0.2	0.06
Windows 2000	0.15	0.2	0.03
XYZ Hardware	0.05	0	0
		f_{yt} = 0.2862	
Engineer_5			
C++	0.981	1	0.981
Windows NT	0.3	0.2	0.06
Windows 2000	0.15	0.2	0.03
XYZ Hardware	0.05	0	0
		f_{yt} = 1.071	
Engineer_6			
C++	0.981	0.2	0.1962
Windows NT	0.3	1	0.3
Windows 2000	0.15	1	0.15
XYZ Hardware	0.05	0	0
		f_{yt} = 0.6462	

$$\sum_{t=1}^{T} X_{yt} \leqslant 1, \quad \forall\, y \tag{6}$$

The second constraint states that each task must be assigned to a resource.

$$\sum_{y=1}^{Y} X_{yt} = 1, \quad \forall\, t \tag{7}$$

6. Survey

The high percentage of software projects that are late can be explained by the methods currently used for personnel assignment, since these methods do not incorporate aspects such as learning capabilities or other skills known by the resources. In fact, Plekhanova (1999) mentions the assumption that decision makers view tasks as one main skill requirement. If tasks are viewed as single main skills, the current resource allocation process could be modeled based on this fact. Since the assumption of describing tasks as single main skills has not been formally tested in the literature, a survey analysis was conducted to test its validity.

The survey was conducted among 30 individuals from both industry and academia. The participants were software engineers, software managers, computer science students, and computer science professors. The survey presented the participants the same sample resource allocation scenario shown in Fig. 1. Recall that this scenario was used in Section 4 to describe the BFR methodology. The survey asked participants to rank the available resources based on the resources' capabilities in the required skills.

In the sample scenario (see Fig. 1), it is safe to assume that expertise in the C++ programming language is the main required skill. Engineer_5 has the highest skill rating in C++. If results from the survey analysis lead to the conclusion that Engineer_5 is the preferred alternative among the survey respondents, then there would be strong evidence to suggest that the selection was based on the capability of candidates in a single main skill. Recall that Engineer_1 was selected as the preferred alternative when considering the complete set of skills of the candidates using the BFR methodology, Engineer_1 was selected as the preferred.

The results of the survey are shown in Table 6. The results indicate that the probability of selecting Engineer_5 as the first choice was the highest with 73%, and Engineer_3 was the second highest with a probability of 27%. A Sign Test was conducted to determine if there was a significant difference in preference between selecting Engineer_5 or Engineer_3 as the first choice. The Sign Test, a non-parametric test commonly used to analyze questionnaire responses, was used to determine if there was a significant difference in preference between alternatives. Letting pE5 indicate the proportion of the population selecting Engineer_5 as the first choice, the following hypotheses were tested:

$$H_0 : pE5 = 0.50$$

$$H_A : pE5 \neq 0.50$$

If the null hypothesis H_0 is supported, then there would be no significant difference in preference among the survey respondents between selecting Engineer_5 or Engineer_3 as the first choice. A supported H_0 would mean that respondents did not consider only C++ (i.e., the main skill) to assess the capabilities of the available resources. A supported alternative hypothesis H_A would mean that the resource assignment was based on knowledge of resources in the single main skill required for the task.

In the Sign Test, a plus sign was used if Engineer_5 was selected as the first choice and a negative sign if it was not. Since a sample size of thirty is considered a large sample (Washington, Karlaftis, & Mennering, 2003), the number of plus signs was approximated by a normal probability distribution with mean and standard deviation given by

$$E_{+} = 0.5n = 0.5 * 30 = 15 \tag{8}$$

$$\sigma_{+} = \sqrt{0.25n} = \sqrt{0.25 * 30} = 2.739 \tag{9}$$

The large sample test statistic is given by

$$Z* = \frac{x - E_{+}}{\sigma_{+}} = \frac{22 - 15}{2.739} = 2.556 \tag{10}$$

where x equals the number of times that Engineer_5 was selected as the first choice. The hypothesis was tested at the 0.05 level of significance; therefore, the Z^{*} test statistic was compared to 1.96. Since $|Z^{*}| \geqslant 1.96$, it is concluded that the null hypothesis is rejected. The analysis of the data collected provided significant evidence that Engineer_5 was the preferred selection for first choice. Therefore, it is concluded that the assumption that decisions are based on the capability of candidates in one main skill is valid. These results

Table 6
Probability of selecting engineers to each choice category

	1st	2nd	3rd	4th	5th	6th
Engineer_1	0	0.2	0.2	0.4	0.2	0
Engineer_2	0	0	0.53	0.13	0.33	0
Engineer_3	0.27	0.67	0.07	0	0	0
Engineer_4	0	0	0	0	0.07	0.93
Engineer_5	0.73	0.13	0.13	0	0	0
Engineer_6	0	0	0.07	0.47	0.4	0.07

are very important because they provide evidence that complete skill sets of resources are not being considered to determine the suitability of candidates with tasks.

7. Contributions and future research

The contributions of this study are twofold. First, a survey analysis provided evidence to support the hypothesis that only one main skill is considered to subjectively assess the suitability of a candidate with a software task. The survey participants were software engineers, software managers, computer science students, and computer science professors. The survey presented the participants with a sample personnel assignment scenario consisting of a task with three skill requirements and a set of six available engineers. Second, a novel methodology was developed to determine the fit (i.e., suitability) between the complete set of skills available from a candidate and the skills required for a task. The model can be used effectively to assign resources to tasks even when the most desirable skills are not available from the existing workforce. The methodology considers the capabilities of candidates in the skills required, the levels of skills required, and the priorities of required skills for the task. A sample case showed the model capabilities. The BFR methodology can be applied in other disciplines with minor modifications.

The work developed in this study opens up areas for future research. First, in this paper resources were evaluated in terms of their technical skills. The BFR methodology could be expanded to incorporate other capability assessment criteria such as project management, social and personal skills, and workplace preferences. Second, tasks' priorities could be included to represent specific organizational goals. Incorporating these priorities into the resource allocation process will provide more effective staffing decisions to high-priority projects, which will result in better return of investment for companies.

Another possible expansion to this study would be to incorporate factors for a more detailed comparison of skills. In the case of programming languages, for example, factors such as syntax and object-oriented capability can be used to compare them. A similar approach can be used for operating systems and project management skills.

Another area for future research is to incorporate imprecise capability assessments using concepts from fuzzy set theory. The relationships between known and unknown skills could be described by fuzzy variables, and fuzzy set rules could be used to describe the suitability of candidates with tasks.

The capability of the BFR methodology was demonstrated using a simple resource allocation sample case consisting of a single task and six available software engineers. Future work is directed towards extending the BFR methodology to medium and large sized projects that call for more requirements and involve larger sets of available personnel.

References

Abdel-Hamid, T. K. (1989). The dynamics of software project staffing: A system dynamics based simulation approach. *IEEE Transactions on Software Engineering, 15*(2), 109–119.

Acuña, S. T., & Juristo, N. (2004). Assigning people to roles in software projects. *Software Practice and Experience, 34*, 675–696.

Acuña, S. T., Juristo, N., & Moreno, A. (2006). Emphasizing human capabilities in software development. *IEEE Software*, 94–101.

Antoniol, G., Cimitile, A., Di Lucca, G. A., & Di Penta, M. (2004). Assessing staffing needs for a software maintenance project through queuing simulation. *IEEE Transactions on Software Engineering, 30*(1), 43–58.

Brooks, F. P. (1975). *The mythical man-month*. Reading, MA: Addison-Wesley.

Duggan, J., Byrne, J., & Lyons, G. (2004). A task optimizer for software construction. *IEEE Software*, 76–82.

Ho-Leung, R. (2001). Using analytic hierarchy process (AHP) method to prioritise human resources in substitution problem. *International Journal of the Computer: The Internet and Management, 9*(1).

Kan, S. H. (1995). *Metrics and models in software quality engineering*. Addison-Wesley.

Kolisch, R. (1999). Resource allocation capabilities of commercial project management software packages. *Interfaces, 29*(4), 19–31.

Linberg, Kurt R. (1999). Software developer perceptions about software project failure: A case study. *The Journal of Systems and Software, 49*, 177–192.

Plekhanova, V. (1999). Capability and compatibility measurement in software process improvement. In *Proceedings of the 2nd European software measurement conference – FEESMA'99, Amsterdam, The Netherlands* (pp. 179–188).

Ross, P. (1996). *Taguchi techniques for quality engineering*. McGraw-Hill.

Roy, R. K. (2001). *Design of experiments using the Taguchi approach*. John Wiley & Sons, Inc..

Shaikh, M. A. (1998). A 'Peak Shaving' approach to project staff reallocation. *Computers and Industrial Engineering, 35*(1–2), 129–132.

Tsai, H., Moskowitz, H., & Lee, H. (2003). Human resource selection for software development projects using Taguchi's parameter design. *European Journal of Operational Research, 153*(1), 167–180.

U.S. General Accounting Office (GAO) (2004). Defense acquisitions: Stronger management practices are needed to improve DOD's software intensive weapon acquisitions. Document number GAO-04-393.

Washington, S. P., Karlaftis, M. G., & Mennering, F. L. (2003). *Statistical and econometric methods for transportation data analysis*. Chapman & Hall/CRC.