

The Battle for the Right Developer or: Who Should be Assigned to this Task?¹

Guenther Ruhe
Expert Decisions Inc.

ruhe@expertdecisions.com

Abstract:

Software project management includes management of human and other resources. Human resources are often the most expensive ones. Their proper usage is of key importance for project success. This article suggests considering assignment of tasks or bugs to developers as a systematic decision-making process. Using advanced IT services such as the ones offered by the decision support tool ReleasePlanner™, can substantially reduce the effort needed for hiring and staffing decisions. Results from a case study project are reported.

The Importance of Staffing

It is well-known that there are major differences in the competences and productivity of software developers [DeMarco and Lister '99]. [Sackman et al. '68] studied professional programmers with an average of 7 years' experience and found that the ratio of initial coding time between the best and worst programmers was about 20 to 1. The general finding that "There are order-of-magnitude differences among programmers" has been confirmed by other studies such as [Valett and McGarry '89] and [Boehm et al. '00]. Acknowledging that all developers have certain strengths in their competence profile, the question becomes: How to identify those people that are best suited for the different development roles?

Despite its importance, there is little support for staffing and hiring processes. Staffing is confronted with the need to adjust to predicted and unpredicted periods of absence. Re-staffing is becoming more and more complex that way. This is both time-consuming and very complex. The same is true for finding the best hires matching predefined requirements. Substantial time savings and improvements in the assignment of developers to tasks can be expected.

¹ The content of this article is largely taken from Chapter 8 of [Ruhe '10]

Staffing – An Illustrative Example

The key idea is to formalize and optimize the staffing and hiring process. For that, a formalization of the actual problem needs to be made. This has been done in [Kapur et al. '08]. Instead of going into the formal details, we present an illustrative example.

We consider a software project for implementation of six candidate features. The resulting product is offered incrementally in two releases. The first one is due after four weeks and the second one after eight weeks from now. Implementation of a feature requires performing three tasks: design, coding and test. Each feature can only be offered if all the three related tasks are performed. There are dependencies between the starting dates of tasks for each feature that need to be fulfilled: Coding should not start before design starts. Similarly, testing should not start before coding starts. There are also dependencies between the end dates: Coding should not finish before design is finished, and testing should not finish before coding is done.

Each feature, once made available in the market, will realize a certain value. It is assumed that this value depends on when it is released. The features $f(n)$ and their estimated relative (measured on a nine-point scale) values called $value(n,k)$ for the case of being offered in release $k = 1$ respectively $k = 2$ are given in Table 1. In addition, Table 1 contains the estimated effort (in person weeks) for each of the three tasks. According to that, feature *Reporting* would create a value of 6 if delivered in the first release, and only a value of 1 if offered in release 2. The same feature has an estimated effort of four person weeks for design, three person weeks for coding, and two person weeks for testing.

Table 1 Feature related data for illustrative sample project

Feature $f(n)$	Functionality	value $v(n,k)$ of $f(n)$ per release k		Effort (in person weeks)		
		$k = 1$	$k = 2$	Design	Coding	Test
$f(1)$	Reporting	6	1	4	3	2
$f(2)$	Billing	5	3	2	3	1
$f(3)$	User administration	9	2	2	3	3
$f(4)$	Notification	8	6	3	4	2
$f(5)$	Data import	9	7	5	6	4
$f(6)$	Data export	9	5	4	5	3

The project team consists of three developers (Anna, Dave, and Ken). From their former work experience, the developers have different productivity levels for each of the tasks of design, coding and testing. The productivity evaluations are estimates and based on the performance in past projects. Average productivity is normalized to 1. A productivity value of 2 for a developer means that this developer can perform a given task twice as

fast as a developer with average productivity. In Table 1, Anna is judged as performing about average for design, 50% above average in coding, and 100% above average developers in testing. While these assumptions are applied for all features of the same project, they might vary across varying projects.

The staffing plan also needs to look at certain periods where developers are not available (due to training, vacation, or other planned periods of absence). During the deployment of the project (8 weeks), Anna is not available to work during weeks 6 and 7.

The developer related data are summarized in Table 1. The general assumption is that one task can only be assigned to one developer. Another assumption is that one developer only can work on one task at a time.

Table 1 Productivity and time windows of unavailability

Developer	Productivity			Unavailable	
	Design	Coding	Test	From week	To week
1 (Anna)	1	1.5	2	6	7
2 (Dave)	2	0.75	1	-	-
3 (Ken)	0.5	2	1	-	-

Due to the limitations in the availability of developers it might not be possible to deliver all features within the predefined period of time. Those that cannot be implemented are postponed. One objective of the planning process is creating a staffing plan that gives the maximum total value for the two product releases.

The value of a product release is additive in terms of the value of the features of that release. An example of a release and staffing plan is shown in Table.2. Each line of the Gantt chart represents a task q of a feature n. The bar represents the developer responsible for that task as well as the time allocated to it. According to this initial plan, feature 1 is released in release 1 (after four weeks) and feature 2 is delivered in release 2 (after eight weeks). All other features f(3), f(4), f(5), and f(6) are postponed.

Table.2 Gantt chart of initial plan x1

Feature/task		Week							
		1	2	3	4	5	6	7	8
Reporting	Design	Anna							
	Code	Dave							
	Test			Ken					
Billing	Design					Ken			
	Code					Dave			
	Test							Anna	

The value of this plan called x1 is composed by adding the value of the two proposed features, i.e., $value(x1) = v(1,1) + v(2,2) = 6 + 3 = 9$. However, this plan is poorly designed because many of the tasks are done by developers who are not the best qualified to perform the job. For example, Dave is responsible for coding, the task for which he is least qualified. Also, some developers are idle at certain point in times such as Ken at weeks 1 and 2

A better solution is shown in Table 3. In this plan, the assignment of developers is significantly improved. There is less idle time for the developers. Tasks are assigned to more productive developers. According to this plan, feature 3 is provided in release 1, features 1, 2, 4 are in release 2 and the two remaining features 5 and 6 are postponed. The overall value of this plan x2 increases to $value(x2) = 19$ compared to the value of 9 of the previous plan x1. In fact, this plan can be shown to be optimal. We observe that the design task for the notification feature is interrupted after the end of week two. This allows Dave to start with the design for billing, which allows starting with the coding for the same feature. For a final plan offered, all the originally defined dependencies need to be satisfied.

Table 3 Table of an improved staffing plan x2

Feature/task		Week							
		1	2	3	4	5	6	7	8
User administration	Design	Dave							
	Code	Anna							
	Test	Ken							
Notification	Design		Dave		Dave				
	Code				Ken				
	Test					Anna			
Reporting	Design					Dave			
	Code							Ken	
	Test								Anna
Billing	Design			Dave					
	Code			Anna					
	Test						Ken		

While the above plan represents the best alternative to select features creating maximum total value, another scenario could be to try implementing all the six pre-selected features with the objective to achieve this with minimum duration. In that case, the value of the features is no longer taken into account. The assumption is that this was done already in the selection of the features. For the set of features listed in Table 1, this results in an overall duration of 13 weeks.

Formulation of Three Staffing Problems

Staffing problem 1: Staffing for maximum total release value

The objective is to find the best subset of features with the best staffing option which results in maximum total release value. Objects of investigation are features with their related tasks and developers with their individual productivity profiles. Release due dates are fixed. All staffing plans are in conformance to the stated availability constraints of developers.

Staffing problem 2: Staffing for minimum make-span

The objective is to implement all the proposed features in minimum overall duration (also called *make-span*). Objects of investigation are features with their related tasks and developers with their productivity profile. Given release due dates can be exceeded. All staffing plans are in conformance to the stated availability constraints of developers.

Staffing problem 3: Staffing for maximum competence match

The objective of the third staffing problem is to find the best assignment of developers to tasks. Each task has a specified profile of requested competences. The tasks are executed by one developer or a set of developers. Even though this is a simplification of reality, it helps to get started with the generation of good assignment plans. Each developer has a specific competence profile, reflecting the amount and characteristics of former working experience. The staffing problem is to assign developers to tasks such that there is best possible total match between requested and offered (by the assigned developers) competences.

Solution Approach

Assigning most competent developers to appropriate tasks is a complex process. If this process is executed in an ad-hoc manner, and solely based on intuition, then the likely results will be poor resource utilization and schedule overrun.

Decision support based on methods and techniques from optimization, machine learning and knowledge management has proven to increase efficiency and effectiveness of the decision-making process. We have applied the staffing component of the decision support tool ReleasePlanner™ offered by Expert Decisions Inc. The tool applies special purpose algorithms from mathematical optimization to generate staffing solutions of proven degree of optimality. The key modeling assumptions for applying rigorous and systematic planning techniques are as follows:

- Pool of developers with specific profiles of productivities
- Consideration of different types of tasks (e.g., design, coding, testing)
- Creation of features as a result of performing a sequence of feature-related tasks.
- Different types of dependencies between tasks
- Periods of absence of developers
- Possibility of pre-assignment of developers to tasks

- Optimized make-span computation

Case Study Results

For staffing problem 2, results of a case study are reported in [Kapur et al. '08]. Two planning approaches were compared:

- Manual planning based on experience and knowledge of both the features and the developers. The respective solution is called a manual plan.
- Optimized planning. The resulting plan is called an optimized plan.

Three main findings were gained.

Finding 1: Quality of plans

A formal comparison of the two plans indicates that the optimized plan is about 43.5% better in terms of the stated quality criteria (satisfaction of stakeholders). Another indicator of the improved quality is the number of features provided according to both plans. While the optimized plan offers 52 features with the two releases, it offers only 42 features for the manual plan.

Finding 2: Quality of staffing

We call an assignment of a developer to a task as *good* if the competence level of the developer for this task is above average. A formal comparison indicates that the optimized plan has more (in total) assignments of developers to tasks. The total number of assignments has increased from 161 to 203 with an increase of the good assignments from 136 to 155. Simultaneously, the amount of idle times has been reduced.

Finding 3: Effort required

Besides the improved quality, the comparison also reveals a significant advantage for the optimized staffing plan in terms of effort needed. The time savings for generation of plans becomes the more significant, the more often additional re-planning or pro-active what-if analysis is conducted (which in reality is happening quite often, in order to adjust to change or to run possible planning scenarios).

Conclusions

Using advanced IT services can substantially reduce the effort needed for hiring and staffing decisions. Experienced project managers have reported spending one third of their time on manually handling the planning process [Kapur et al. '08]. This substantial amount of effort compares with seconds of running the algorithms, plus the initial effort to set-up the formalized planning effort, which is in the order of 2 to 3 hours, depending of the size and complexity of the project.

Optimized staffing policies were also applied to the case of assigning developers to tasks for bug fixing. As a result of the optimized assignments, time savings in the order of 20%

were achieved in the retrospective analysis of 9 milestone Eclipse projects [Rahman et al. 09].

Four scenarios for staffing and re-staffing in case of unpredicted absence and/or changed project data (such as estimated versus actual effort) are studied in [Livani and Ruhe '10]. The scenarios include decision support for hiring new people to complement an existing team structure. They utilize the capabilities of the web-based decision support system ReleasePlanner™.

References

[Boehm *et al.* '00]

Boehm, B., Abts, C. and Chulani, S., *Software Development Cost Estimation Approaches - a Survey*, *Annals of Software Engineering*, 10 (2000), pp. 177-205.

[DeMarco and Lister '99]

DeMarco, T. and Lister, T., *Peopleware: Productive Projects and Teams*, 2nd ed., Dorset House, 1999.

[Kapur *et al.* '08]

Kapur, P., Ngo-The, A., Ruhe, G. and Smith, A., *Optimized Staffing for Product Releases and Its Application at Chartwell Technology*, *Journal of Software Maintenance and Evolution*, 20 (2008), pp. 365-386.

[Livani and Ruhe '10]

Livani, E. and Ruhe, G., *Decision Support for Staffing and Re-staffing of the Next Software Product Release*, accepted for SEKE '2010.

[Rahman *et al.* '09]

Rahman, M. M., Ruhe, G. and Zimmermann, T., *Optimized Assignment of Developers for Fixing Bugs: An Initial Evaluation for Eclipse Projects*, *Proceedings Empirical Software Engineering and Measurement*, 2009, pp. 439 - 442.

[Ruhe '10]

Ruhe, G., *Product Release Planning – Methods, Tools, and Applications*. CRS Press, appears in June 2010.

[Sackman *et al.* '68]

Sackman, H., Erikson, W. J. and Grant, E. E., *Exploratory Experimental Studies Comparing Online and Offline Programming Performance*, *Communications of the ACM*, 11 (1968), pp. 3-11.

[Valett and McGarry '89]

Valett, J. D. and McGarry, F. E., *A Summary of Software Measurement Experiences in the Software Engineering Laboratory*, *The Journal of Systems and Software*, 9 (1989), pp. 137-148.