Staffing a Software Project: a Constraint Satisfaction Approach

Ahilton Barreto
COPPE-UFRJ
Caixa Postal: 68511
Rio de Janeiro, Brazil
55-21 2562-8675
ahilton@cos.ufrj.br

Márcio Barros
DIA-UNIRIO
Av. Pasteur 458, Urca
Rio de Janeiro, Brazil
55-21 2244-5613
marcio.barros@uniriotec.br

Cláudia Werner
COPPE-UFRJ
Caixa Postal: 68511
Rio de Janeiro, Brazil
55-21 2562-8675
werner@cos.ufrj.br

ABSTRACT

This paper presents an optimization based approach to support staffing a software project. The approach takes into account the characteristics of the project activities, the characteristics of the available human resources, and constraints established by the software development organization in charge of the project. According to these needs, the project manager selects a utility function to be maximized or minimized by the optimizer. We propose several utility functions, each addressing specific values which can be sought by the development organization.

Keywords
Staffing, Constraint Satisfaction Problems.

1. INTRODUCTION

Software development organizations survive in a competitive market by profiting from the conversion of developers’ effort to useful and successful software products. To build such products, the organization usually follows a process that divides the development effort into several activities. Each of these activities requires specific characteristics (e.g., such as skills, capabilities, and experiences). Most of these characteristics are found in human resources assigned to accomplish the project activities. So human resource allocation, or staffing, is an important issue to be analyzed when software development is undertaken as a value-driven business.

This raises a fundamental question: given a group of available developers and a set of project activities, which developers-to-activity allocation yields more value to the organization? Since human resources usually represent the major cost account for a software project and the alignment of their characteristics with activities requirements is a driving factor towards productivity and quality, optimizing the allocation of available developers in accordance to the constraints imposed to (e.g., such as schedule deadline, project budget, and head-count) or by (maximum allocation, minimum effort to participate, among others) the development organization may determine whether a project will be profitable or not.

However, staffing a software project is not a simple activity. There are many alternatives to ponder, several developer-to-activity combinations to evaluate, and the manager may have to choose a team from a larger set of available developers, according to the project and organizational needs. For instance, consider a situation where there are ten professionals and ten activities. The total number of available combinations is about 10 billions. To perform the allocation activity considering all the attributes involved and all possible solutions without automated support is a difficult or even impossible task, when considering all combinations. Due to this complexity, the manager usually is not able to consider all the different possible combinations and has to choose a team based on his experience and impressions. But once he doesn’t consider all the possible teams, he may choose a team that is not the best one to the specific situation.

In our approach, we address project staffing as a constraint satisfaction problem [1], defining utility functions that should be maximized or minimized by the selected development team, in order to provide a greater value for the organization. We devised several utility functions, among which one should be selected according to organizational needs or constraints.

This paper is divided into four sections. The first one comprises this introduction. The second section provides some assumptions upon which we draw a model for software project staffing. In Section 3, we present the utility functions currently available to optimize the model presented in Section 2. In Section 4 we show an example situation where our approach could be used. In Section 5 we describe the experimental study which is being planned to evaluate the viability of the approach. Finally, in Section 6 we present some conclusions and work to be further addressed in our research.

2. STAFFING A SOFTWARE PROJECT

Staffing must be done in a way that maximizes the creation of some value to a project. In this sense, the semantics of value must be addressed regarding project and organizational characteristics. Some projects are schedule driven: to create value for such project could mean to act in a way that reduces its schedule or risks associated to it. Other projects may be driven by budget, resource allocation, and so on. So, the maximization target for the staff allocation optimizer cannot be fixed by a single utility function, but several such functions should be available for the manager to decide which best fit the project under analysis.

We have investigated the project staffing problem as a constraint satisfaction problem. A constraint satisfaction problem can be described as a tuple \( S = (V, D, R) \), where \( V = \{x_1, x_2, \ldots, x_n\} \) is a finite set of variables; \( D = \{d_1, d_2, \ldots, d_m\} \) is a finite set of domains; and \( R \) is a set of constraints, limiting the values that the variables can assume simultaneously. For the project staffing problem, \( V \) is the set of activities of the project, \( D \) is the set of professionals that can perform each activity and \( R \) is the set of constraints (characteristics needed and possessed).
In our approach, we assume that the first step in staffing is related to defining the characteristics that a team member might have to be able to perform at each project activity. There is also a need to determine what characteristics the available developers have. In our approach, we consider that a characteristic may be a skill, a capability, an experience, some knowledge, a role in the organization or in the project, and others. Each characteristic is associated with a rating scale, with the intensity levels the characteristic may assume. This first step is complex and is not the focus of this work. Some work has been done trying to determine capability-person and capability-role relationships [2], but in our work we consider that this initial definition has to be done by the manager, based on his experience. Such preparations aim to reduce the risk of allocating a developer to an activity not compatible with the developer’s skills. So, staffing is performed according to the following rules:

1. (a) A person can only be allocated to an activity if he or she possesses at least all the characteristics demanded by the activity, in an intensity level greater or equal to the demanded;

2. (b) A person can only be allocated to an activity if he or she is available to perform the activity in the period it needs to be performed (reasons of unavailability could be allocation to other activities, vacation, etc).

Expressing it in a formal way:

- Let P be a set of projects. Each element \( P_i \in P \) is composed by a name and a set of activities (Aij);

\[
P = \{ P_i \}
\]

\[
P_i = [\text{name}, \{Aij\}]
\]

- Let HR be a set of professionals from the organization, available to be allocated to project activities. Each HRi \( \in HR \) is described by a name, cost (per hour), maximum number of workable hours per day, a set of periods in which the developer will not be available to take part in the project (PUij), and a set of characteristics (CHRij);

\[
HR = \{ HRi \}
\]

\[
HRi = [\text{name}, \text{Shour}, \#hour, \{PUij\}, \{CHRij\}]
\]

- Let Ai be a set of activities from a given project Pi. Each Aij \( \in Ai \) is described by a name, initial and final dates, minimum daily effort that should be spent by a developer on the activity (expressed in hours), and a set of desired developer characteristics (CAij);

\[
Aij = [\text{name}, \text{initial_date}, \text{final_date}, \#hours, \{CAijk\}]
\]

- Let UPi be a set of periods in which a given developer HRi will not be available to take part in the project. Each UPij \( \in UPi \) is described by initial and final dates, and the number of hours per day that the professional will not be available;

\[
UPij = [\text{initial_date}, \text{final_date}, \#hours]
\]

- Let C be a set of characteristics possessed by a professional HRj or desired by an activity Ak. Each Ci \( \in C \) is described by a name, a scalar value that denotes a maximum intensity for such characteristic among software developers or required by activities, and its intensity for a specific developer or required by an activity;

\[
Ci = [\text{name}, \#maximal, \#desired/possessed]
\]

So, the staffing problem may be described as follows:

\[
\forall Aij \in Ai \subset Pi \in P (\exists HRk \in HR (\alpha \wedge \beta))
\]

The formulation above shows that for each activity belonging to a given project’s set of activities, there is some professional for whom both \( \alpha \) and \( \beta \) conditions are true. The \( \alpha \)-condition is described as follows:

\[
\forall ((Cm \in C \subset C)(\exists (Cl \in C \subset HRk) \wedge Cm = Cl \wedge Valu(Cl) \geq Valu(Cm))))
\]

The above formula establishes that for each characteristic that belongs to the set of desired characteristics of an activity, there must be a characteristic belonging to the set of possessed characteristics of a professional, and this characteristic has the same intensity level desired by the activity, and the intensity level of the characteristic possessed by the professional is greater or equal to the intensity level desired by the activity. The \( \beta \)-condition is described as follows:

\[
\neg \exists (PUi \in PU \subset HRk)
\]

\[
\begin{align*}
(\text{InitDate}(Ai) & \geq \text{InitDate}(UPi)) \wedge \\
(\text{InitDate}(Ai) & \leq \text{EndDate}(UPi)) \wedge \\
(\text{EndDate}(Ai) & \geq \text{EndDate}(UPi)) \wedge \\
(\text{InitDate}(Ai) & \leq \text{EndDate}(UPi)) \wedge \\
\text{MaxHoursDay}(HRk) & < \text{NumHours}(UPi) + \text{NumHours}(Ai)
\end{align*}
\]

The above formula establishes that there is no period over which the developer will be unavailable for the project that intersects the period in which the activity is scheduled to be accomplished. It also states that the developer’s daily effort is greater or equal to the effort required by the activity.

3. STAFFING UTILITY FUNCTIONS

Staffing a software project, to account for the characteristics required by activities and those possessed by developers is not always enough. Several other values may distinguish development teams, according to project or organizational characteristics. For instance, a project may require the most qualified and skilled team available, when considering aspects like productivity, client satisfaction, product reliability, time-to-market and so on.

On the other hand, the better usage of the skills available in an organization may be more important, and these skills should not be wasted. So, a manager may choose the team that minimizes skills’ overusage (in other words, where the distance between the needed and employed skills is minimal). In a different scenario, a manager may need to choose the cheapest team that can execute the project, considering developers’ salaries and overhead costs. Sometimes large teams are a problem, because there are so many communication channels that the productivity can be affected. So, the manager may want to choose the smallest possible team to the project.
Moreover, though accounting for the characteristics required by the project activities is important, sometimes it is not possible to get a team where members possess all the characteristics required by the project activities. So, a manager may choose to minimize the distance between what is needed and what the team possesses, building a team as close as possible to the desired characteristics.

Considering these issues and that their importance may differ for distinct projects, our approach suggests that the manager staffs the project considering one of these utility functions:

- most qualified team: team in which the distance between the demanded characteristics and the possessed ones is maximal. In other words, the optimizer will suggest a team in which developers are more capable than project activities require them to be:

  - Let T be the set of possible teams that satisfy all the constraints. Let Overusage (T_i) be a function that returns the distance between the qualification of a given team T_i and the qualification demanded by the project. The selected team T_i ∈ T is chosen according to the following equation:

  \[ \neg \exists (T_i \in T) (Overusage (T_i) < Overusage (T_i)) \]

- cheapest team: the optimizer selects the team that represents the lowest cost to the project, considering a fixed cost for each team member throughout the project:

  - Let T be the set of possible teams that satisfy all the constraints. Let Cost (T_i) be a function that returns the cost (monetary) of a given team T_i to the project. The selected team T_i ∈ T is chosen according to the following equation:

  \[ \neg \exists (T_i \in T) (Cost (T_i) > Cost (T_i)) \]

- least qualified team: team in which the distance between the demanded characteristics and the possessed ones is minimal. In other words, the optimizer will search for a team in which skill overusage is minimized:

  - Let T be the set of possible teams that satisfy all the constraints. Let Overusage (T_i) be a function that returns the distance between the qualification of a given team T_i and the qualification demanded by the project. The selected team T_i ∈ T is chosen according to the following equation:

  \[ \neg \exists (T_i \in T) (Overusage (T_i) > Overusage (T_i)) \]

- smallest team: the optimizer suggests a team composed by the smallest possible number of members:

  - Let T be the set of possible teams that satisfy all the constraints. Let Size (T_i) be a function that returns the size (number of different members) of a team T_i. The selected team T_i ∈ T is chosen according to the following equation:

  \[ \neg \exists (T_i \in T) (Size (T_i) > Size (T_i)) \]

- best partial solution team: the optimizer indicates a team that is the best partial solution to the problem. This solution is usually applied when the available developers’ characteristics do not satisfy all requirements of the project activities. So, the optimizer searches for a solution (that is, a team) that minimizes the number of broken constraints:

  - Let T be the set of possible teams. Let Broken (T_i) be a function that returns the number of broken constraints of a team T_i. The selected team T_i ∈ T is chosen according to the following equation:

  \[ \neg \exists (T_i \in T) (Broken (T_i) < Broken (T_i)) \]

We suggest these five functions to show how our approach could be used, but it would be possible to define and use other functions based on project or organizational needs.

### 4. Example Use of the Technique

To illustrate the use of our approach, in this section we present an example situation where the approach could be used.

The fictitious project is composed by five activities and seven professionals, as follows:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Period</th>
<th>Required Characteristics (Intensity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>22/03/2005 to</td>
<td>- Relationship with people (2)</td>
</tr>
<tr>
<td>Elicitation</td>
<td>29/03/2005</td>
<td>- Negotiation (2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Requirements elicitation techniques (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Experience in telecommunications (3)</td>
</tr>
<tr>
<td>Analysis</td>
<td>30/04/2005 to</td>
<td>- Team work (2)</td>
</tr>
<tr>
<td></td>
<td>07/04/2005</td>
<td>- Object Oriented Analysis (3)</td>
</tr>
<tr>
<td>Design</td>
<td>06/04/2005 to</td>
<td>- Team Work (2)</td>
</tr>
<tr>
<td></td>
<td>14/04/2005</td>
<td>- Databases (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Object Oriented Design (3)</td>
</tr>
<tr>
<td>Construction</td>
<td>15/04/2005 to</td>
<td>- Databases (3)</td>
</tr>
<tr>
<td></td>
<td>30/04/2005</td>
<td>- Java (3)</td>
</tr>
<tr>
<td>Tests</td>
<td>01/05/2005 to</td>
<td>- Tests Techniques (3)</td>
</tr>
<tr>
<td></td>
<td>06/05/2005</td>
<td>- Java (3)</td>
</tr>
</tbody>
</table>

Table 1 – Activities

<table>
<thead>
<tr>
<th>Professional 1</th>
<th>Cost/ Hour</th>
<th>Possessed Characteristics (Intensity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professional 1</td>
<td>15,00</td>
<td>- Databases (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Java (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Negotiation (2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Team work (2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Professional 2</th>
<th>Cost/ Hour</th>
<th>Possessed Characteristics (Intensity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professional 2</td>
<td>30,00</td>
<td>- Object Oriented Design (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Experience in telecommunications (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Object Oriented Analysis (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Requirements elicitation techniques (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Databases (3)</td>
</tr>
</tbody>
</table>
in any discipline involves building models that can be tested, through empirical study, to check whether the current understanding of the field is correct. Progress comes when what is actually true can be separated from what is only believed to be true [3].

So, an empirical study is being planned to evaluate the viability of using the proposed technique and its supporting tool.

The goal of the study will be:

**Analyse the proposed approach and its supporting tool**

**for the purpose of characterize the viability of use and continuity of the development**

**with respect to the gains obtained by its use**

**from the point of view of the researcher**

**in the context of Software Engineering MSc and DSc students.**

The subjects of the study will be asked to perform the activity of selecting a team to a fictitious project defined in laboratory (In-Vitro study). The same project will be presented to all the subjects and they will have to determine a team that satisfies all the constraints involved in the problem and that is the cheapest possible team. The teams chosen by the subjects will be compared with the teams chosen by the supporting tool.

The null hypothesis of the study states that the use of the proposed approach and its supporting tool brings no benefits to staffing a software project, i.e., there is no significant difference in the time needed to perform the activity and in the quality (with respect to the cost) of the chosen teams if the activity is performed ad-hoc or using the proposed approach.

\[ H_0: \mu_{\text{cost without technique}} = \mu_{\text{cost with technique}} \text{ and } \mu_{\text{time without technique}} = \mu_{\text{time with technique}} \]

The alternative hypothesis, on the other hand, states that the subjects will get worse results, once they will not use the proposed approach, i.e., their chosen team will be more expensive than the ones chosen by the supporting tool, and they will take more time to choose the team than the tool will.

\[ H_1: \mu_{\text{cost without technique}} > \mu_{\text{cost with technique}} \text{ and } \mu_{\text{time without technique}} > \mu_{\text{time with technique}} \]

**6. CONCLUSION**

Software development involves time, talent, and money. In a competitive market, one of its major goals is to maximize value creation for a given investment [4]. So, a good use for every available resource is very important.

Software designers, engineers, and managers must understand and reason effectively about the connections between technical decisions and enterprise-level value maximization [4]. In our approach we address these connections by accounting for technical needs (characteristics possessed by people and demanded by project activities) in the light of business constraints (cost, schedule, time size, among others). So, by offering distinct value (utility) functions to drive the technical-oriented staffing optimization process, we allow the manager to balance between project technical needs and organizational constraints.
To support the operational usage of our approach, we developed a tool in which the manager informs the characteristics required by project activities, the available developers and their characteristics, and selects one of the proposed utility functions. The tool runs the optimization problem and generates a project staffing suggestion according to the given parameters.

The next step of our work is to execute the empirical study briefly described in section 4. As further works we could mention the investigation of the determination of the required characteristics by the activities and of the possessed characteristics of each professional.

7. REFERENCES