ANNALS OF THE UNIVERSITY OF CRAIOVA
Series: Automation, Computers, Electronics and Mechatronics, Vol. 14 (41), No. 1, 2017

Initial Results on the Effectiveness of a Skill-Based Approach to Human Resource Allocation

Ionuț Murareț u[*], Sorin Ilie[*], Mihaela Ilie[*]

[*]*Department of Computers and Information Technology, University of Craiova*
107 Decebal Bvd, Craiova, Romania
(email: imuraretu@gmail.com, silie@software.ucv.ro, ela.pirvu@gmail.com)

Abstract: This paper introduces a skill based approach to human resource allocation. For this purpose a mathematical model is introduced modelling tasks and employees as vectors of skill. Then we present 5 strategies of allocating tasks to employees modeled as fitness functions. These functions are then compared in a simulated environment. The conclusion of the experiment is that allocating tasks to suboptimal employees can speed up the project delivery time at least fivefold. The novelty of this approach is mainly that we opted for a heuristic model of recommendation focused resource allocation that can be applied in continuous task flows.

Keywords: Mathematical model, Human resource management, Enterprise resource planning, Optimal matching

1. INTRODUCTION

The human resource management problem within an organization using Agile development methodology is a challenge that is normally left to the most experienced workers in the team. This is usually because the task requires the assignee to know both the team and technical details of the project. In this paper we will be referring to Agile software development and the teams of programmers using this methodology.

When large teams and task backlogs Minor et al.(2007) are involved the task of human resource management critical and time consuming. Therefore, a time-saving solution is called for. The systems we found during our literature review are based on optimization algorithms. These approaches tend to be hard to put into practice because in real life the input data keeps changing: new tasks appear, employees leave, team leaders impose their decisions based on experience Yngve et al. (2016). Any of these situations require the mentioned approaches to re-assign some of the tasks in order to reach the optimum and never take into account the current load of each team member. As presented in paper Minor et al (2007), task flow in Agile development has a feedback loop Fig.1. This is why we believe that creating a mathematical model based on fitness evaluations and ordered lists of recommendations for task assignment would be more directly implementable in practice.

The rest of this paper is structured as follows: section 2 presents related work from our survey of the literature on the subject; section 3 presents the mathematical model of the proposed approach and the performance measures; section 4 briefly presents how this approach could be implemented in practice; section 5 presents an experimental simulation with the proposed approach and discusses the results; section 6 discusses future work and draws conclusions.

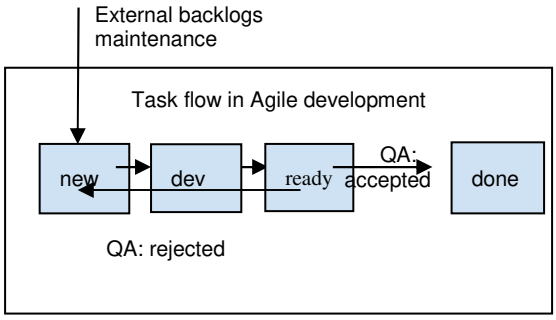

Fig. 1. A simplified business process model of task flow management in agile software development.

2. RELATED WORK

Bojan et al (2012) creates an agent-based simulation (ABS) Helbing et al (2012) for "Simulation Process Simulation Modeling (SPSM)" applied in the case of a real software development project. They show that the ABS can estimate project duration well. Also they propose an effort function which estimates the behavior of a developer for a given task. The authors do not attempt to present how this simulation can be used to improve task management in the real world.

In paper Colucci et al (2004) the authors propose a skill based solution for task assignment in an organizational context. They use weighted bipartite graph in which arcs' weight represents computed suitability of users on each task using a skill matching algorithm. The skill matching system has two components. The first component uses

classical information retrieval techniques with semantics to extract individual profiles from text files. The second component of the system sends extracted profiles to a matchmaker service. Third component is based on Khun algorithm Burkard et al (2012) will do the assignment based on the score obtained by the second component. The authors presume that the number of tasks equals the number of workers which is restrictive to say the least. Their algorithm requires a complete reallocation in case a single extra task would appear.

In paper Roque et al. (2016) the authors propose a multi-objective approach for minimizing cost and time of software development based on the Scrum(Stellman et al 2014) method. They calculate both similarity of tasks and skill matching to obtain time and cost values for tasks. Then the computed time and cost are being minimized respecting that the amount of overtime worked by an employee which cannot be greater than allowed and an employee cannot be assigned to a task if he does not possess a required skill.

Otero et al (2009) propose a resource allocation methodology called Best-Fitted Resource(BFR) that describe how learning of a required skill is affected by a previous knowledge on a related skill. The BFR methodology has four steps with each step resulting in a table. Their hypothesis is supported by a survey they conducted on real subjects. Their methodology setup requires skills to be marked as related to other skills if they can influence learning speed for those other skills.

In paper Bădică et. al. (2011) the authors present a way to match service providers and consumers using weighted utility functions.Service providers and consumers are represented as agents (Ilie et al 2010). This approach is similar to our own however it is applied between organisations and at each point the users need to fill in a form to fill in the service attributes. In order to keep consistent service attributes throughout this system, a term ontology is used. The authors present no evaluationand only minimal mathematical formalization. Therefore, it cannot be easily adapted to a task management scenario.

It is evident that in the very specific scenario in which a project has just started with a completely fresh team the aforementioned approaches are valid and quite efficient at keeping cost down. However, besides the individual comments we had on the papers we also have a global criticism of the papers we reviewed. The approaches they propose are static: they expect to know all the tasks from the beginning, no reassessment and the team leaders cannot impose their will on the system. None of the approaches take into account the fact that the employees might already have workload assigned, presuming that all employees are completely free at the time of assignment. This does not reflect the reality of team and project management in practical situations, especially in the domain of software development as shown in the real world analysis presented in McBride et al(2008). At this time, we do not propose a replacement approach for the usual optimization algorithms but rather we evaluate the effectiveness of a recommender system. This system could be used after the initial optimal assignment using an iterative algorithm.

3. MATHEMTICAL MODEL

In this section we will present how we model and propose to calculate instantaneous fitness functions based only on current task assignment for each employee and skill suitability. Towards the end of the section we will present performance measures calculated on the tasks assigned so far to the employees.

We define k as the total number of skills required in a project. This number is not task dependent but a global variable. A project is assigned to a team $\Lambda$of n employees P . The project also has an initial ordered list Θ of m tasks T defined from the start.

An employee P is modelled as a vector of k skill levels S_i.

$$P = (l_1, l_2, l_3 \dots l_k) \quad (1)$$

A task T is modelled as a vector of k skill levels Si expected from an employee in order to finalize this task efficiently and an estimation of time for task completion given a suitably adept employee.

$$T = \{(S_1, S_2, S_3 \dots S_k), e_t\} \quad (2)$$

We can now visualize a task and employee with 10 skills as a radar chart (Kaczynski et al 2008). For example, in Fig. 2 we show visually how the required skill levels and the employee skill levels overlap.

A few simplifying assumptions are:

- all employees start working simultaneously on their assigned tasks in sequence until they finish them all
- it may be necessary to assign a task to an employee of inadequate skill level but in this case the estimated time of task completion will go up, as explained in the rest of this section.
- tasks do not depend on other task results. For example, this could be a SCRUM[1]Sprint, a project phase, or maintenance tasks.
- the speed of learning a new level in a skill is directly proportional to the employee's current skill level. A higher level implies faster acquiring of new knowledge in that skill.
- all employees have the exact same cost per hour therefore only hours will be taken into account. The number of hours will be sufficient in assessing performance.

[1]The SCRUM methodology of task managementhttps://www.scrum.org/

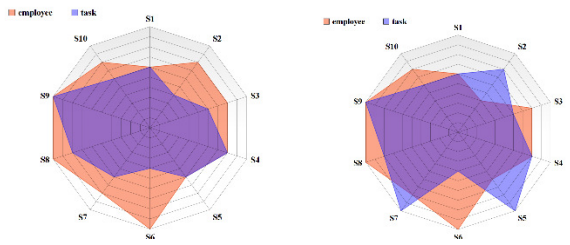

Fig. 2. Representing tasks(blue) and employees(orange) as areas in radar charts. On the left: an employee that can immediately do a task. On the right: an employee that needs to learn more about 3 of his skills before being able to handle the task The overlapping area becomes purple.

We define positive distance to minimum skill $\delta(P,T)_i$ that is strictly >0 if and only if the employee P i-th skill level is less that that required by task T. Quantitatively, the value of $\delta(P,T)_i$is the actual modulo difference of skill. In Fig2 , $\delta(P,T)$would be the number of points of a skill that are required by a task but unsatisfied by an employee (blue areas). Mathematically this can be calculated for skill number i as follows:

$$\delta(P,T)_i = \frac{(1+sgn(S_i-l_i))sgn^2(S_i-l_i)}{2}|S_i-l_i| \qquad (3)$$

where sgn is the mathematical function signum that returns 1 if the parameter is positive, -1 if the parameter is negative and 0 if the parameter is 0. The first fraction equals 1 if $S_i > l_i$and 0 otherwise

Similarly to the work of Otero et al (2009) we estimate the amount of time for a employee P to finalize task T depending on the $\delta(P,T)_i$ and the level of the employe. For each distance unit we add $\varphi/l_i$to the initial estimated time e.

$$e(P,T) = e_t + \varphi\sum_1^k \frac{\delta(P,T)_i}{l_i} \qquad (4)$$

where φ is the difficulty constant of each skill in particular. This constant determines how hard it is to lean a new skill level.

We define the instantaneous fitness functions $f: \Lambda \times \Theta \rightarrow \Re$, where $\Lambda$is the set of a employees$P$ and $\Theta$is the set of tasks T, using the following rationales:

1. assign to the best prepared employee. The rationale being that this person will do the task in the least amount of time. In Fig 2 the orange surface of the employee should cover as much as possible of the radar and implicitly the task.
2. assign to the most suitable. The rationale here is that the task should always be assigned to the employee that is closest match to the skills required by the task. Compared to strategy 1, this should free up some time for the best employees while avoiding to use ill prepared employees for difficult tasks. In Fig2 the aim would be to find the employee and the task that over impose exactly.
3. assign to the employee that can do the task the fastest. This strategy does not take into account how busy that employee might be. assign to the most suitable employee that is least occupied.
4. assign to the most suitable employee who is is also least occupied.
5. assign to the employee that can do the task the fastest but is also least occupied.

In case number 1 to find the best employee for a task, we define the fitness function f_1 of an employee P for a given task T as follows:

$$f_1(P,T) = \frac{\sum_1^k (l_i+1)S_i}{\sum_{j=1}^n \sum_{i=1}^k (l_{ji}+1)S_i} \qquad (5)$$

where l_{ji} is the level of skill i of user j, these values are translated with +1 in order to avoid cancellation of necessary skills from the task in case l=0.

The denominator is used to refer each employee to the whole team as a whole, i.e. the fraction is normalized.

In case 2 to find the most suitable employee for a job, we define the fitness f_2 as follows:

$$f_2(P,T) = r(P,T) \qquad (6)$$

where r is the correlation of T and P defined as the dot product of their skill vectors as follows:

$$r(P,T) = \frac{\sum_{i=1}^k l_i S_i}{\sqrt{\sum_{i=1}^k l_i^2}\sqrt{\sum_{i=1}^k S_i^2}} \qquad (7)$$

In case 3 in order to identify the fastest employee for a given task T, we define the fitness function $f_3$as follows:

$$f_3(P,T) = \frac{1}{e(P,T)} \qquad (8)$$

The fitness function $f_3$generates the least amount of man-hours by always assigning the fastest employee to each task. Implicitly $f_3$generates the least amount of cost.

In case 4 to identify the most suitable employee that is least occupied at the moment. In essence the fitness of the employee is directly proportional to r, and inversely proportional to estimated time of the task $e(P,T)$ when it is executed by P , and the sum of we define the fitness function $f_4$as follows:

$$f_4(P,T) = \frac{r(P,T)}{e_p}\frac{\bar{e}}{e(P,T)} \qquad (9)$$

Where:

r is the correlation between the skill vector of the task T and the skill vector of the employee P as defined above

$e_p$is the sum of all time estimations for employee P on tasks allotted to him/her

$\bar{e}$ the mean of the $e_p$for each employee within a margin of ε to a perfect correlation r=1 to task T

In case 5 to identify the fastest employee that is least occupied at the moment, we define the fitness function f similarly to case 4, as follows:

$$f_5(P,T) = \frac{1}{e_p}\frac{\bar{e}}{e(P,T)} \quad (10)$$

We will now define performance measures and mathematical models that they are derived from. After applying these allocation strategies, the result is would be a set of m employee-task pairs with the fitness function up to that point in the project lifetime similar to the work of Colluci et al (2004).

$$\alpha_x = \{(P,T)|\forall\, T \in \Theta, \exists P \in \Lambda, f(P,T) = max_{i=1}^{k}(f_x(P_i,T))\}$$

where x is the index of the fitness function, used to do the task allocation, as it results from the current paper

In order to compare the four fitness functions, we calculate the following performance measures similarly to Minor et al (2007):

- estimated project delivery time d. Defined as the estimated time it takes the busiest employee to finish his/her work.

$$d(\alpha) = max\{e(P,T)|\ (P,T) \in \alpha\} \quad (11)$$

- project total man-hours $t(\alpha)$. This number also reflects the project cost, as this is directly proportional to the financial resources required by the development team

$$t(\alpha) = \sum_{(P,T)\in\alpha} e(P,T) \quad (12)$$

4. IMPLEMENTATION

This approach is meant to be used as a recommender module added to an existing task management system like Redmine[2]. The module would receive as input a given task T and the complete list of employees. It would then calculate the fitness of each employee w.r.t. to the given task. This is done by taking into account the considered employee skills, the current unassigned task skill requirements and the currently assigned tasks to the employee. See figure 3 for the information flow (Ilie et. al. 2012) of the implementation.

The proposed mathematical model does not require multiple iterations. The output would be a descending ordered list of employees in terms of fitness for each unassigned task. That is to say, the complexity of the recommender algorithm is O(n). Therefore, tasks can be a variable queue just like in real projects.

For the purpose of finding the fitness function with the best performance we implemented an algorithm that receives as input the list of unassigned tasks $\Theta$and the list of employees Λ.

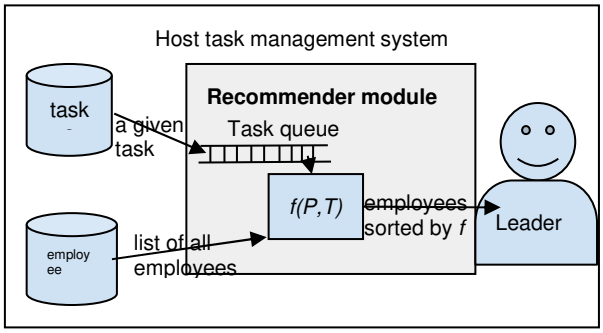

Fig. 3. In this figure, the tasks, employees and team leader are part of the task management system, the Recommender module is the additional module needed to tell the team leader assign tasks

For each unassigned task $T \in \Theta$ we then calculate the fitness function $f(P,T)$ for every employee $\in \Lambda$. We can then evaluate the estimated time in which the employee with the maximum fitness function f can finalize the current task and add it to the allocation α. At this point performance measures can be calculated over α. In short the algorithm used is as follows:

```
1. program Simulation (Θ,Λ)
2.  begin
      for each T ∈ Θ :
3. for each P ∈ Θ :
4.       * find maximum f(P,T) and
              add it to α
5.       * calculate e(P,T)
6.       * sort α
7.  * send them to the view module
8.   end.
```

This algorithm calls a subroutine for recommendations view at line 7 to show how this code could be used in a real world implementation. However, this line has no functionality for the purpose of this paper so it is not used.

For the following experiment we evaluate the scenario in which the team leader always accepts all the recommended task assignments. This was done in Python[3], using the amazon Cloud 9 web based IDE equipped with an Ubuntu virtual machine[4]. For replication and evaluation, the code can be found on a public gitHub repository[5].

We generated a random list of m=2000 tasks Θ, n=100 employees Λ, with skill arrays of size k=100, skill levels $S_i, l_i \in \overline{0..5}$ and initial estimated time for each task $e_t \in \overline{1..8}$. These values represent a relatively complex phase of

[2]The Redmine task management system website https://www.redmine.org/

[3] Python programming language, release 3 https://www.python.org/download/releases/3.0/

[4] Cloud 9 web-based IDE from Amazon equipped with an Ubuntu virtual machine https://aws.amazon.com/cloud9/

[5] The public code GitHub repository used for the implementation of the experimental simulation https://github.com/sorinilie/taskmanagement

a project where each task requires 100 separate skills to complete, and the complete team is composed of 100 employees of various skill levels. As these skills are imaginary we simplified calculations by choosing the learning difficulty of all skills $\varphi = 1$. This does not influence the fitness function calculations since this constant would be applied to all employees. We will not mention the specifications of the machine we used because no execution time evaluation will be presented.

After executing the simulation algorithm detailed in the previous section, we calculated the performance measures presented in the mathematical model section of this paper: Project delivery time d and total man-hours t. Additionally we found it interesting to present the percentage amount of human resources allotted to the project for each fitness function presented. The numeric results are presented in Table 1 and visualized in Fig 4 and Fig 5.

As expected, allocation α_3, "assign the task to the employee that can finish it fastest", has the smallest total man-hour cost $t(\alpha_3) = 80441$ therefore we can use it as a baseline for the cost discussion. Any other allocation is more expensive, however, there are other allocations that present a better project delivery time. The good performance of $\alpha_3$is due to the fact that $f_3$minimises the amount of hours spent learning new skills.

The results show that α_4 "assigning tasks to the suitable employee that is less occupied" and α_5 "assigning tasks to the employee that can finish the task fastest that is also less occupied" are the most efficient in obtaining fast project delivery times. This is due to the fact that their fitness function spread out the workload more evenly between the employees.

We calculated that allocation α_4 will lead to finishing the project in $d(\alpha_4) = 1284$ calendar hours. Therefore, it will bring the project to an end 93.7% faster than $\alpha_3$but it costs 16.4% more than α_3. Similarly, allocation α_5 finalized the project in $d(\alpha_5) = 1278$ callender hours. In consequence, it will bring the project to an end 93.8% faster than $\alpha_3$but it costs 16.6% more than α_3. We postulate that this reduced project duration is due to the bias for less occupied employees of $f_3$and f_4. In support of this affirmation we present f_2 and $f_3$that do not use this bias and result in considerably longer project duration.

Table 1. Experimental result for the 4 fitness functions proposed for comparison.

measure	$f_1$eq(5)	$f_2$eq(6)	$f_3$eq(8)	$f_4$eq(9)	$f_5$eq(10)
project delivery time *d*eq(11)	68712	5911	20536	1284	1278
total man-hour *teq(12)*	86875	87701	80441	96228	96492
resource allocation	20%	97%	47%	100%	100%

The "best skilled employee per task" allocation α_1 is bad for project duration $d(\alpha_1) = 68712$ and cost $(\alpha_1) = 86875$. It will bring the project to an end 334% slower than $\alpha_3$and it costs 7% more than α_3. This strategy only uses the 20% of the human resources, i.e. the best skilled 20% of employees will be busy while the others are completely free. This particular allocation is completely impractical in a real software development organization.

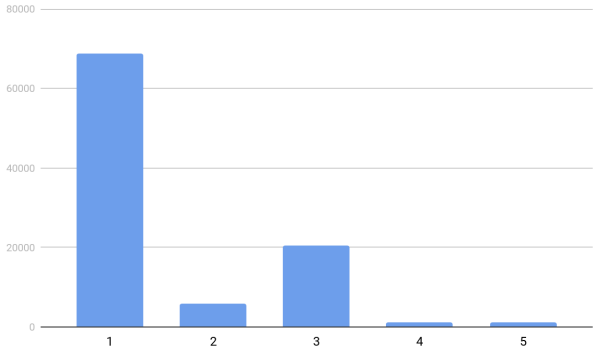

Fig. 4. The calendar duration of the project with the 5 strategies: 1) best worker ; 2) most suitable; 3) fastest; 4) most suitable that is least occupied; 5) fastest that is least occupied.

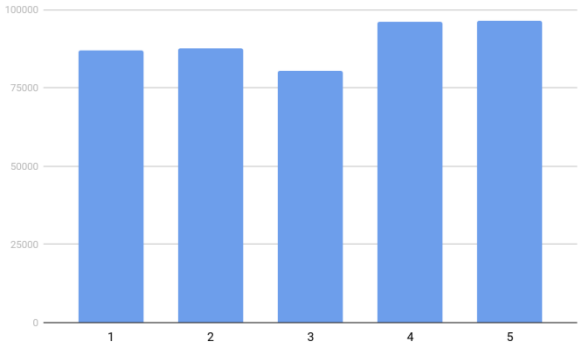

Fig. 5. The comparison of the total man-hour resulting from allocation α for the 5 fitness functions.

5. CONCLUSIONS AND FUTURE WORK

This paper introduced a skill-based approach to human resource allocation. The conclusion of the experiment is that allocating tasks to suboptimal employees can speed up the project delivery time at least fivefold. The novelty of this approach is mainly that we opted for a heuristic model of recommendation focused resource allocation that can be applied in continuous task flows. This topic requires a lot of future work: natural language processing and sentiment analysis on task comments, assessing task progress health, adjustment of skill vectors automatically when employees learn new skills, implementing the system on Redmine or Taiga[6].

[6] A task management system that allows skills to be specified for users and tasks https://taiga.io/

ACKNOWLEDGMENT

Sorin Ilie was supported by QFORIT Programme, University of Craiova, 2017.

REFERENCES

BojanSpasic,Bhakti S. S. Onggo: "Agent-based Simulation Of The Software Development Process: A Case Study At Avl", Proceedings of the 2012 Winter Simulation Conference, ISBN 978-1-4673-4782-2

Helbing, D., 2012. Agent-based modeling. In Social self-organization (pp. 25-70). Springer Berlin Heidelberg.

Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M. and Piscitelli, G., 2004. Semantic-based Approach to Task Assignment of Individual Profiles. J. UCS, 10(6), pp.723-730.

Burkard, R., Dell'Amico, M. and Martello, S., 2012. Assignment problems: revised reprint. Society for Industrial and Applied Mathematics, pp 42-47.

Roque, L., Araújo, A.A., Dantas, A., Saraiva, R. and Souza, J., 2016, October. Human Resource Allocation in Agile Software Projects Based on Task Similarities. In International Symposium on Search Based Software Engineering (pp. 291-297). Springer International Publishing.

Stellman, A. and Greene, J., 2014. Learning agile: Understanding scrum, XP, lean, and kanban. " O'Reilly Media, Inc."

YngveLindsjørn, Dag I.K. Sjøberg ,TorgeirDingsøyr , Gunnar R. Bergersen , Tore Dybåa : "Teamwork quality and project success in software development: A survey of agile development teams" The Journal of Systems and Software 122 (2016) 274–286

Kaczynski, D., Wood, L. and Harding, A., 2008. Using radar charts with qualitative evaluation: Techniques to assess change in blended learning. Active Learning in Higher Education, 9(1), pp.23-41.

McBride, T., 2008. The mechanisms of project management of software development. Journal of Systems and Software, 81(12), pp.2386-2395.

Otero, L.D., Centeno, G., Ruiz-Torres, A.J. and Otero, C.E., 2009. A systematic approach for resource allocation in software projects. Computers & Industrial Engineering, 56(4), pp.1333-1339.

Minor, M., Tartakovski, A. and Bergmann, R., 2007, August. Representation and structure-based similarity assessment for agile workflows. In International Conference on Case-Based Reasoning (pp. 224-238). Springer, Berlin, Heidelberg.

Ilie, S., Bădică, C., Bădică, A., Sandu, L., Sbora, R., Ganzha, M. and Paprzycki, M., 2012 :" Information flow in a distributed agent-based online auction system". In Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics (p. 42). ACM.

Bădică, C., Ilie, S., Kamermans, M., Pavlin, G. and Scafeş, M., 2011:"Using negotiation for dynamic composition of services in multi-organizational environmental management". In International Symposium on Environmental Software Systems (pp. 177-188). Springer, Berlin, Heidelberg.

Ilie, S. and Bădică, C., 2010: "Distributed multi-agent system for solving traveling salesman problem using ant colony optimization". In Intelligent Distributed Computing IV (pp. 119-129). Springer, Berlin, Heidelberg.