A Hybrid Approach to Solve the Agile Team Allocation Problem

Ricardo Britto
DIE - UFPI - Brazil
rbritto@ufpi.edu.br

Pedro Santos Neto
DIE - UFPI - Brazil
pasn@ufpi.edu.br

Ricardo Rabelo
SEL - USP - Brazil
ricardor_usp@ieee.org

Werney Ayala
DIE - UFPI - Brazil
werney.zero@gmail.com

Thiago Soares
DIE - UFPI - Brazil
thiagoacs2@gmail.com

***Abstract*—The success of the team allocation in a agile software development project is essential. The agile team allocation is a NP-hard problem, since it comprises the allocation of self-organizing and cross-functional teams. Many researchers have driven efforts to apply Computational Intelligence techniques to solve this problem. This work presents a hybrid approach based on NSGA-II multi-objective metaheuristic and Mamdani Fuzzy Inference Systems to solve the agile team allocation problem, together with an initial evaluation of its use in a real environment.**

I. Introduction

A well known definition of Software Engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software [1]. Summarizing, this could be explained as the application of engineering to software, since this comprises the use of mathematics, computer science and practices whose origins are in Engineering.

There are several software development methodologies based on Iterative and Incremental Lyfe Cycle Model, the most used currently [2]. Most of them are called Agile Methodologies. In these methodologies there is an intensive collaboration between self-organizing and cross-functional teams. The Agile Methodologies promote adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change [3].

It is important to emphasize the relevance of the team in the Agile Methodologies. A successful agile software development team is made up of competent developers. However, this introduces a new concept that needs to be understood at all: competency. Competency is the ability of a developer to perform a job properly, but it is also defined as a combination of Knowledge, Skills and Attitudes (KSA) used to improve performance [4].

We can notice that the composition of a team is complex, since we have to consider all the characteristics mentioned before. This task must be done during the project planning, usually in the beginning of the project. This task is very relevant: if done in a wrong way, could drive the project to the cancellation state [5].

Since the team allocation is a NP-hard problem [6], using Software Engineering standard techniques to solve that problem could be very complex. Some studies present alternatives for solving that problem, mainly through the use of metaheuristics ([7], [8], [9]). In this context, a new research area has emerged by the application of search techniques to complex software engineering problems. This new research area is called Search-Based Software Engineering (SBSE) [10], which concerns on finding solutions to complex problems formulated as a search problem in the software development process.

This paper presents an approach based on NSGA-II multi-objective metaheuristic [11] and Mamdani Fuzzy Inference System [12] to solve the agile team allocation problem. This approach allows an easy specification of rules to choose the best developers to a given project.

The idea is innovative and has the following major contributions:

- An approach to evaluate developers based on a questionnaire;
- An approach to estimate developers productivity based on a fuzzy inference system and KSA evaluation;
- An approach to qualify the generated solutions (allocated teams) based on a fuzzy inference system;
- A hybrid approach to solve the addressed problem which incorporates in all of its stages the expertise of a project manager;
- An empirical evaluation of the proposed approach, comparing its application against the intuition of some project managers.

This paper is organized as follow: Section II presents some related works; Section III provides a brief description of fuzzy inference systems; Section IV contains a brief description about multi-objective metaheuristics; Section V depicts the proposed technique to evaluate developers; Section VI presents our proposed approach to solve the addressed problem; Section VII discusses the initial evaluation performed to validate the proposed approach, and finally, Section VIII presents the conclusion and future works.

II. Related Works

In [13] combining search-based optimization techniques with a queuing simulation model for allocating human resources to a software project and assigning tasks to teams. The obtained solution of this approach aims to minimize the completion time and reduce schedule fragmentation. Project managers can run multiple simulations and consider trade-offs between increasing the staffing level and anticipating

the project completion date and reducing the fragmentation and accepting project delays. The presented results are obtained from two large scale commercial software maintenance projects.

In [7], the team allocation problem was handled as a two phase choice model: a company chooses the desired employees to compose a team, and the employees choose the desired positions in the team. The authors used genetic algorithm to optimize team allocation in such model.

In [14] the authors present a human resource evaluation approach to team allocation. The approach uses a hierarchical analysis process combined with fuzzy mathematics to evaluate the employees in terms of skill and external factors (social and health factors).

Shan, Jiang and Huang [8] proposed an approach that considers the different human resources demands required by different tasks and the characteristics of different human resources roles. The approach realized the goal of assigning different team members to different tasks at the least cost under the conditions of satisfying the resource demand of tasks and the constraint of effort of different stages, which is based on accurate estimation by the information available at the planning stage. A genetic algorithm is then designed to solve this complex problem.

The work of [9] addresses the team allocation problem in a multi-objective way. That approach takes into account aspects such as employee preferences, the overall ability of the team, and the total salary cost. To solve the problem it is used the multi-objective metaheuristics NSGA-II and MOCell.

Ning [15] studies the allocation of enterprise human resources based on artificial neural networks. Firstly, the concept and characteristics of the enterprise human resources allocation are introduced. And then, a number of factors which affect the allocation of human resources in enterprises and which also influence personnel transference are analyzed. On this basis, a neural network is designed in order to predict the probability of the human resources transference.

In [16], the authors propose an optimization model for the allocation of multi-skilled human resources to research and development projects, considering individual workers as entities having different knowledge, experience and ability. The proposed model focuses on three aspects of human resources: the different skill levels, the learning process and the social relationships existing in working teams. The resolution approach for the multi-objective problem consists of firstly, obtain a set of non-dominated solutions exploring the optimal Pareto frontier and secondly, select the best solution compromise with regards to the considered objectives. The uncertainty associated to each solution is modelled by fuzzy numbers.

Considering the related works mentioned before, our work is the one that proposes a hybrid approach which uses fuzzy inference systems to simplify the team allocation process. This feature reduces the number of variables to be optimized by the multi-objective metaheuristic and reduces the search space. We also include the expertise of project manager into the fuzzy inference systems of our approach in a simple manner. This expertise is used to estimate the developers productivity and also to choose the best solution generated by the NSGA-II metaheuristic.

Our work also addresses the team allocation problem from the agile development point of view, evaluating all the available resources as developers. This feature influences our proposed human resource evaluation technique and the proposed fuzzy inference systems. Once we considerate all the human resources as developers, the whole procedure to alocate a team is simplified.

III. Fuzzy Inference Systems

Fuzzy inference systems are capable of dealing with highly complex processes, which are represented by inaccurate, uncertain and qualitative information. Normally, fuzzy inference systems are based on linguistic rules of the type "if **condition** then **action**", in which the fuzzy set theory [17] and fuzzy logic [18] provide the necessary mathematical basis to deal with inaccurate information and with the linguistic rules.

In this work, Mamdani fuzzy inference systems [12] are used to generate productivity of an human resource using knowledge, skill and attitude and also to estimate the team quality using team productivity and team cost. The production rules in an inference model of Mamdani have linguistic variables both in their antecedents as in their consequents. By using fuzzy inference systems of Mamdani, it is possible to represent the heuristic knowledge of software engineer in a way totally linguistic.

IV. Multi-objective Metaheuristics

Multi-objective optimization is the process of simultaneously optimizing two or more conflicting objectives that could be subject to certain constraints. Multi-objective problems are mathematically depicted as [19]: $\underset{x}{min}[\mu_1(x), \mu_2(x), ..., \mu_n(x)]$ subject to $g(x) \leq 0$ $h(x) = 0$ $x_i \leq x \leq x_u$ where μ_i is the $i-th$ objective function, *g* is the inequality constraint, *h* is the equality constraint, *x* is the vector of optimization or decision variables. In multi-objective optimization problems, instead of being a unique solution, the solution is a possibly infinite set of Pareto points.

A point in objective space μ^* is called Pareto optimal if there is not another objective vector μ such that $\mu_i \leq \mu_i^*$ for all $i \in \{1, 2, ..., n\}$, and $\mu_j \leq \mu_j^*$ for at least one index of j, $j \in \{1, 2, ..., n\}$.

There are many approaches to solve multi-objective optimization problems. Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [11], MOCell [20] and Strength Pareto Evolutionary Algorithm 2 (SPEA-2) [21] have become standard approaches. In subsection IV-A, we briefly explain the NSGA-II metaheuristic.

A. NSGA-II

The NSGA-II metaheuristic begins generating a population P_0 of size N, sorted according the dominance operator. A second population, called Q_0, with the same size of P_0,

is generated applying crossover and mutation operators over P_0. The 2 populations are combinated in a third population called R_0, with size $2N$. The population R_0 is sorted using the dominance operator. Then, a population called P_1 is populated with Pareto fronts. The Pareto fronts are generated using the crowding distance operator over the population R_0. The population P_1 is used like initial population to the next generation of the algorithm execution. That process continues until the stopping criterion is reached.

V. Developer Evaluation Approach

The developer evaluation technique depicted in this section was developed in partnership with a software factory of Piauí, Brazil. The technique, named Competence Matrix, has a set of questions used to generate a score summarizing the performance of a developer.

The matrix is split in three areas: Knowledge, Skill, and Attitude. Each area is related to some aspect of the developer's behavior in a company. Formal education gives us the knowledge that we require to be a competent developer. Informal education on the other hand is what you learn outside of books. Together, all sources of education make up the knowledge of a developer and are able to use on daily tasks. Skill is the next factor that a developer may not have, but can acquire through experience over time. A developer could be trained about how to work in a team. The skill factor applies to everything that we do in life, do it often enough and you will become better at it as time goes on. The most difficult ability to acquire if you do not already have it is attitude. Attitude is the mentality that you bring to a task, the way we think, feel and do. It is necessary to have pride, passion and belief in the work, in order to show attitude. Summarizing, the Knowledge is related to know; the Skill is related to know-how; and the Attitude is related to want to do. This is directly related to Competency [4]. The Knowledge (K) area contains the indication of the degree of knowledge of a developer related to several themes. A specific theme is technology. The company mapped all the technology used for its operation. Each theme has four levels of knowledge: null (0), low (1), regular (2) and good (3). Each level has a description used to characterize the knowledge associated to this level.

The knowledge area is very extensive and covers several items related to i) Programming, ii) Modelling, iii) Testing, iv) Database, v) Web frameworks, vi) Office tools, vii) Networking, viii) Support, ix) Web design, x) Business, xi) Management, xii) Software process, and xiii) Natural languages. Nowadays there are 71 items in the Knowledge area.

The Skill (S) area provides information about how to use the knowledge in order to do something. It is composed by six questions in the following areas: i) Communication, ii) Creativity, iii) Emotional Intelligence, iv) Leadership, v) Organization, and vi) Teamwork.

The Attitude (A) area is related to the behavior registered during the daily tasks of a software factory. It is composed by five questions in the following areas: i) Discipline, ii) Initiative / Proactivity, iii) Interest and Dedication, iv) Punctuality, and v) Relationship.

Similarly to the knowledge area, there are levels related to each item in Skill and Attitude area. A level has a detailed description used to facilitate the choice of an answer.

There is a process to run the competence matrix. All the available developers must fill their own knowledge, skill and attitude forms. For the skill and attitude forms, the project manager and the team members must also fill the forms for each developer. Thus, each developer has at least three scores for each item: the self-evaluation (SE), one evaluation for each team member (TM), and the manager evaluation (ME). There are weights for each kind of evaluation category, in order to generate the developer score.

To calculate the score of Knowledge, Skill and Attitude of each developer, we use the Equations 1, 2 and 3:

$$K = K_{SE} * W_{SE} + K_{TM} * W_{TM} + K_{ME} * W_{ME} \quad (1)$$

$$S = S_{SE} * W_{SE} + S_{TM} * W_{TM} + S_{ME} * W_{ME} \quad (2)$$

$$A = A_{SE} * W_{SE} + A_{TM} * W_{TM} + A_{ME} * W_{ME} \quad (3)$$

To calculate the sub-values used in equations 1, 2 and 3, we have used equation 4.

$$SB = \frac{\sum_{i=1}^{N} (Q_i)^2}{\sum_{i=1}^{N} Q_i * 3} \quad (4)$$

Where:

- K = Knowledge of the developer.
- S = Skill of the developer.
- A = Attitude of the developer.
- *SE* is related to the self-evaluation of the developer.
- *TM* is related to the team evaluation of the developer.
- *ME* is related to the manager evaluation of the developer.
- $SB = \{K_{SE}|K_{TM}|K_{ME}|S_{SE}|S_{TM}|S_{ME}|A_{SE}|A_{TM}|A_{ME}\}$.
- W = Weight of an evaluation in the score calculation.
- N = Number of questions of an area.
- Q = Question of an area.
- i = index of a question.

During the use of the proposed technique, we have noticed that the Knowledge, Skill and Attitude scores are closed to the productivity of an evaluated developer. So, this could be generalized in order to help another project managers to allocate teams. Therefore, we decided to analyze what is the key point in the technique, allowing the specification of simple rules related to the three areas of assessment. These rules are used to build the rule base of the proposed fuzzy inference system for estimating developer productivity.

VI. Proposed Approach

This section presents the proposed approach to allocate agile teams (Figure 1). Our approach has 5 stages:

- **Stage 1** - Developers evaluation;
- **Stage 2** - Estimation of developers productivity;

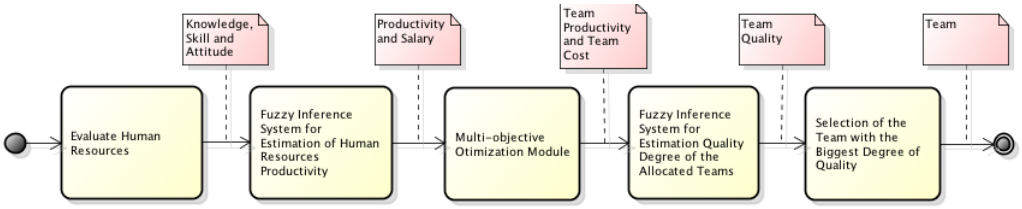


Fig. 1. Proposed Approach

- **Stage 3** - Generation of Teams using a multi-objective metaheuristic;
- **Stage 4** - Evaluation of the generated solutions;
- **Stage 5** - Selection of the best generated solution.

A. Stage 1: Developers Evaluation

We described in section V the methodology used in this work. At this stage, we applied questionnaires to the available developers in order to measure the values of Knowledge, Skill and Attitude of each developer.

B. Stage 2: Estimation of Developers Productivity

We evaluated the 3 values measured in stage 1 and we realized that would be possible to estimate the productivity of each developer from these values. Aiming to extract a single value representing the developers evaluation, we projected a Mamdani fuzzy inference system to infer the developer productivity (Figure 2).

The crisp input variables of the fuzzy inference system are Knowledge, Skill and Attitude. The values of each one of these variables are measured in stage 1. The output variable of the fuzzy inference system is Productivity. The 3 input variables and the output variable can receive real values between 0 and 10.

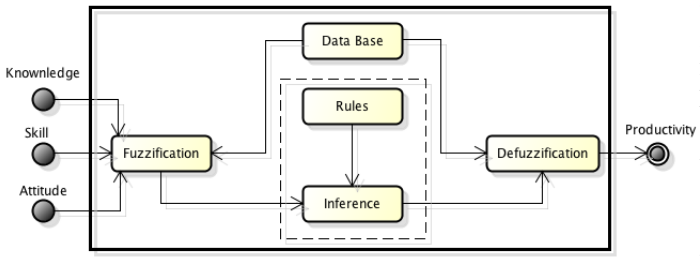


Fig. 2. Fuzzy Inference System for Estimation of Developers Productivity

To map the fuzzy sets, we used a uniform distribution of the sets. This is the normal process to do this task. After done the uniform distribution of the fuzzy sets, it is possible to improve the mapping empirically or using computational intelligence techniques such genetic algorithms [22]. We used triangular membership functions to empirically map fuzzy sets to the input variables. To each input variable 3 fuzzy sets were mapped: Low (L), Medium (M) and High (H). We also used triangular membership functions to empirically map fuzzy sets to the output variable. We mapped 5 fuzzy sets to Productivity: Very Low (VL), Low (L), Medium (M), High (H) and Very High (VH).

To generate Productivity of a developer using the three specified input variables, we must set the rule base in the proposed fuzzy inference system. The implemented rule base, which is the kernel of the productivity estimation strategy for developers, has 27 rules. The rule base is presented in a matrix way, as seen in Table I.

TABLE I
RULE BASE OF THE FUZZY INFERENCE SYSTEM FOR ESTIMATION OF DEVELOPERS PRODUCTIVITY

	Knowledge / Attitude								
Skill	L/L	L/M	L/H	M/L	M/M	M/H	H/L	H/M	H/H
L	VL	L	L	L	L	L	L	M	M
M	L	M	M	M	M	M	H	H	VH
H	L	M	H	H	H	H	VH	VH	VH

In this work we implemented the centroid defuzzification technique [23] for obtaining the numerical value of the productivity.

C. Stage 3: Generation of Teams Using a Multi-objective Metaheuristic

By using inferred productivity (Stage 2) and the salary of each available developer, the next stage of our approach generates a set of possible teams. The final team is chosen from this set of solutions. The desirable team is the one that has the lowest cost and the highest productivity. Once we have 2 distinct and, eventually, conflicting objectives, is convenient to use a multi-objective metaheuristic. By using this kind of algorithm facilitates the modeling of the objective functions.

We used in this work the NSGA-II [11]. We chosen this metaheuristic because it is one of the most popular and easy to implement multi-objective metaheuristic. In Figure 3, we present the multi-objective optimization module implemented in this work. The execution of this module generates a set of solutions. Each solution represents a different team.

As we discussed before in this section, we aim to allocate a team that has the lowest cost and the highest productivity. Thus, we consider the following aspects in the optimization process:

- **Productivity** - Each available developer has a value of productivity estimated in stage 2 of our approach;
- **Salary** - Each available developer has a cost that is equal to the salary and taxes spent with him.

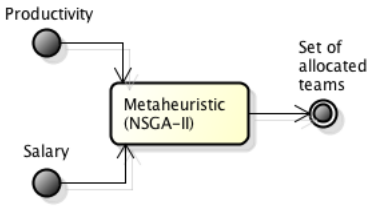


Fig. 3. Implemented NSGA-II Otimization Module

The NSGA-II algorithm needs a population of chromosomes to optimize these 2 exposed aspects. Each chromosome represents a team. The chromosomes of the population manipulated by the NSGA-II were represented using binary vectors. Each element of the chromosome represents an available developer. The length of the vector must be equal to the number of available developers. If one element of the vector is equal to 0, it is mean that the developer represented by this element is not allocated to the team represented by the respective chromosome. If one element of the vector is equal to 1, it is mean that the developer, represented by this element, is allocated to the team represented by the respective chromosome.

The 2 objective functions used by the multi-objective module of our approach are following depicted:

$$Minimize : (\sum_{i=1}^{N} W_i * D) + PF * |T - E| \quad (5)$$

$$Maximize : (\sum_{i=1}^{N} P_i * D) - PF * |T - E| \quad (6)$$

$$PF = \begin{cases} 0, if\ E \geq T \\ \alpha \textbf{ or } \beta, if\ E < T \end{cases} \quad (7)$$

Where:

- i = Developer index.
- N = Total number of available developers.
- W = salary of the developer.
- P = Productivity of the developer.
- D = 0 if the developer is not allocated to the team or 1 if he is allocated.
- E = Size of the desired team.
- T = Size of a team proposed by the algorithm.
- PF = Penalization factor.
- α = value of the PF associated with the cost of the team.
- β = value of the PF associated with the productivity of the team.

The α e β constants were specified to punish the solutions that have the number of developers bigger than the size of the desired team (E). Solutions that have the number of developers smaller than the size of the desired team are naturally punished, because these solutions are always dominated by those that has the desired size.

D. Stage 4: Evaluation of the Generated Solutions

Once the multi-objective module has generated the set of teams (Stage 3), the next stage of our approach evaluates the quality of each allocated team. This evaluation is necessary because all the teams of the final set generated in stage 3 has the same importance from the point of view of NSGA-II. In other words, the solutions of the final set are not dominated by any other solutions. Thus, to select the best team from the solution set generated by the optimization module, we must consider which of the 2 optimized aspects are most important to the project manager of the software project.

In our approach, we implemented a Mamdani fuzzy inference system to incorporate the project manager intuition in the team selection process (Figure 4). This fuzzy system qualifies each team allocated by the optimization module, guided by the intuition of the project manager. After the qualification process, each generated solution will have a quality level. The project manager must choose the team with the biggest level of quality.

The crisp input variables of the fuzzy inference system are Team Cost and Team Productivity. The values of each one of these variables are measured in stage 3. The output variable of the fuzzy inference system is Team Quality.

The input variable Team Cost can receive real values between 0 and $N * 10000$, where N is equal to size of the desired team (number of developers). The input variable Team Productivity can receive real values between 0 and $N * 10$, where N is also equal to size of the desired team (number of developers). The output variable Team Quality can receive real values between 0 and 10.

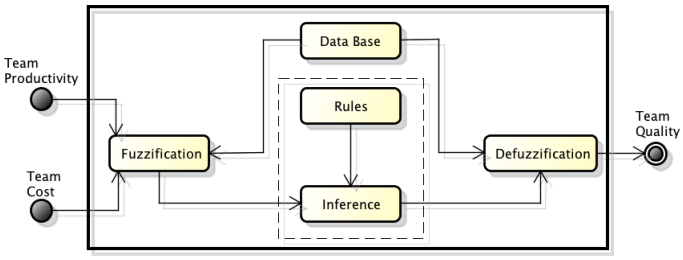


Fig. 4. Fuzzy Inference System for Qualify the Allocated Teams

We used triangular membership functions to empirically map fuzzy sets to the input variables. To each input variable 3 fuzzy sets were mapped: Low (L), Medium (M) and High (H).

We also used triangular membership functions to empirically map fuzzy sets to the output variable. We mapped 5 fuzzy sets to Team Productivity: Very Low (VL), Low (L), Medium (M), High (H) and Very High (VH).

To generate Team Quality using the two specified input variables, we must set the rule base in the proposed fuzzy inference system. The implemented rule base, which incorporates what the project manager thinks about what is a good team, has 9 rules. The rule base is presented in a matrix way, as

seen in Table II. It is important to highlight that we constructed this rule base using the knowledge of 3 project managers of the partner software factory.

TABLE II
RULE BASE OF THE FUZZY INFERENCE SYSTEM FOR QUALIFY THE ALLOCATED TEAMS

	Team Cost		
Team Productivity	L	M	H
L	L	L	VL
M	H	M	L
H	VH	H	M

As in stage 2 of our approach, we implemented the centroid defuzzification technique. It is important to claim that the implemented fuzzy inference system allows user to change the defuzzification method in a simplified way.

E. Stage 5: Selection of the Best Generated Solution

The last and most simple stage of our approach is selecting the best team. After the qualification process fulfilled in stage 4, the project manager must select the team with the highest value of Team Quality.

VII. EMPIRICAL EVALUATION

In order to evaluate the proposed approach, we have applied it in a software development company of Brazil. We have used three project managers of the company to select, in autonomous way, teams with different sizes, among different possibilities, in order to compare the results from the automated approach against the project managers intuition, since they can estimate the productivity of each member of the company.

The partner company develops software related to healthship. There are three products and three project managers that control the mentioned products and all the actions related to each one. They have to control several projects related to the improvement of the products. During the evaluation, there were 19 developers (including the project managers) in the company but the teams follow the agile prescriptions: they are self-organized and cross-functional teams. The project managers know all the collaborators, since there is a regular rotation among the projects, in order to avoid the creation of knowledge islands.

The company gave us up to four hours of each project manager to perform the study activities. We have executed a simplified version of the competence matrix. In this version, we selected only 11 questions related to Knowledge, six questions related to Skill, and five questions related to Attitude.

We asked for each project manager to answer the questionnaire with these questions about your specific team, not about the overall members. This comprises the Stage 1 of the proposed approach. The results from this stage are depicted in columns named **Dev**, **Knowledge**, **Skill** and **Attitude** of Table III. The table shows also others columns. Column **Prod** is the productivity calculated by the fuzzy system (Stage 2).

We have asked for the project managers to generate a list of the members, ordered by the productivity (descending). The project managers used only their intuition for the accomplishment of this task. The sequence of each project manager can be observed in Table III. Columns **PM1**, **PM2** and **PM3** show the position of a developer in the perception of each project manager, respectively. Column **Fuzzy** shows the position of the developer according to the input fuzzy system.

In order to facilitate the observation about the concordance among the project managers, we have created the columns **G1**, **G2**, **G3**, and **Gf**. These columns group the developers into groups according to their positions in the columns PM1, PM2, PM3 and Fuzzy. Developers with position between 1-4 are in the group A. Developers with position between 5-7 are in the group B. Developers with position between 8-10 are in the group C. Developers with position between 11-13 are in the group D. Developers with position between 14-16 are in the group E. The column **ConcD** shows the project managers concordance among themselves. This value was calculated using the following rules:

- if the developer appears in the same group for each project manager (e.g. A, A, A), the concordance is 100%;
- if the developer appears in the same group for 2 project managers (e.g. A, B, A), the concordance is 66%;
- if the developer appears in diferents groups for each project manager (e.g. A, C, B), the concordance is 0%;

The average of the concordance among project managers was 73%. In order to calculate the concordance between input fuzzy inference system and project managers, we have calculated the concordance, showed in the column **ConcDF**, using the following rules:

- if the developer appears in the same group for each evaluation (e.g. A, A, A, A), the concordance is 100%;
- if the developer appears in the same group for three evaluations (e.g. A, A, B, A), the concordance is 75%;
- if the developer appears in only two groups twice (e.g. A, B, A, B), the concordance is 66%;
- if the developer appears in the same group for two evaluations (e.g. A, A, B, C), the concordance is 50%;
- if the developer does not appears in the same group anytime (e.g. A, D, B, C), the concordance is 0%;

The average of the concordance among project managers and productivity fuzzy was also 73%. This represents an important result, since the inclusion of the fuzzy system kept the concordance in the same level. Hence, we have noticed that the input fuzzy inference system has mimicked the behavior of the domain experts.

As mentioned before, we have also asked the project managers to allocate teams with three, five and seven developers. We have done this to allow a comparison between the project managers and our approach.

We have instructed the project managers to allocate teams trying to balance the team cost and team productivity. These allocations were done using their intuition about the developers and the information about the salary of each member, available for consultation. Table IV shows the allocations performed in the study. Column **Coach** represents the responsible for the

TABLE III
EVALUATION OF THE PRODUCTIVITY BY THE PROJECT MANAGERS.

Dev	Knowledge	Skill	Attitude	Prod	Salary	PM1	G1	PM2	G2	PM3	G3	ConcD	Fuzzy	Gf	ConcDF
1	8.95	6.88	7.83	9.09	6750.00	2	A	1	A	2	A	100%	5	B	75%
2	8.36	7.21	8.13	9.34	5000.00	1	A	2	A	1	A	100%	1	A	100%
3	7.00	6.50	8.13	7.50	3750.00	8	C	11	D	7	B	0%	9	C	50%
4	7.18	5.00	5.68	5.27	5000.00	10	C	7	B	10	C	66%	12	D	50%
5	5.83	5.96	5.50	7.15	5000.00	9	C	6	B	8	C	66%	7	B	66%
6	5.76	6.43	5.50	6.91	5000.00	7	B	9	C	9	C	66%	14	E	50%
7	5.76	7.21	7.50	6.25	3750.00	6	B	5	B	5	B	100%	8	C	75%
8	5.76	5.23	6.67	5.20	5000.00	13	D	14	E	14	E	66%	10	C	50%
9	5.76	5.83	7.14	5.29	3750.00	11	D	13	D	12	D	100%	6	B	75%
10	6.79	7.50	8.44	9.31	5750.00	5	B	4	A	4	A	66%	2	A	75%
11	6.54	6.43	6.14	6.59	5000.00	12	D	10	C	11	D	66%	11	D	75%
12	7.94	8.50	8.44	9.28	5000.00	4	A	3	A	3	A	100%	4	A	100%
13	5.63	5.68	6.73	5.00	3750.00	16	E	15	E	16	E	100%	16	E	100%
14	6.39	5.42	7.83	7.24	3750.00	14	E	12	D	13	D	66%	13	D	75%
15	6.15	5.83	7.12	5.17	3000.00	15	E	16	E	15	E	100%	15	E	100%
16	7.94	8.03	9.17	9.28	5000.00	3	A	8	C	6	B	0%	3	A	50%
Average												**73%**	**PMs + Fuzzy**		**73%**

TABLE IV
TEAMS ALLOCATION BY THE PMS AND THE PROPOSED APPROACH.

Coach	Size	Developers	Prod	Salary	Quality
PM 1	3	2,1,16	27,71	16750	7,5
PM 2	3	2,7,3	23,09	12500	6,96
PM 3	3	2,12,7	24,87	13750	7,5
Approach+	3	2,10,16	27,93	15750	7,5
Approach+-	3	2,3,12	26,12	13750	7,5
Approach-	3	2,3,14	24,08	12500	7,5
PM 1	5	2,1,16,7,3	41,46	24250	7,5
PM 2	5	2,7,3,12,10	41,68	23250	7,5
PM 3	5	2,12,7,16,3	41,65	22500	7,5
Approach+	5	1,2,10,12,16	46,30	27500	7,5
Approach+-	5	2,3,10,14,16	42,67	23250	7,5
Approach-	5	2,3,7,14,16	39,61	21250	7,5
PM 1	7	2,1,16,7,3,10,12	60,05	35000	7,5
PM 2	7	2,7,3,12,10,16,5	58,11	33250	7,5
PM 3	7	2,12,7,16,3,5,10	58,11	33250	7,5
Approach+	7	1,2,3,5,10,12,16	60,95	36250	7,5
Approach+-	7	2,3,7,10,12,14,16	58,20	32000	7,5
Approach-	7	2,3,5,7,12,14,16	56,04	31250	7,5

allocation. There are three project managers (PM1, PM2, PM3) and three allocations proposed in our empirical evaluation. Columns Approach+, Approach+- and Approach- show the highest, the medium and the lowest team cost, among all the possible selections with the highest quality score.

In our approach, NSGA-II module finds a set of possible solutions (Pareto front), instead of only one solution like project managers (Stage 3). We have run the NSGA-II module 30 times for each team size (3, 5, 7) to minimize the random behavior of the implemented metaheuristic. The final Pareto fronts for each team size are shown in Figure 5. We can notice that the two variables increase in the same ratio. The better is the team productivity, the higher is the team cost.

In order to qualify the solutions (Stage 4), using team cost and productivity, we have used another fuzzy inference system. The more balanced the cost and productivity of a team, the better is the team quality in our approach. Table IV also shows the teams selected by the system, together with the team productivity, cost, and quality score.

We can notice that the teams allocated using our approach, although different from the teams allocated by the project managers, have a high degree of similarity. Besides, the teams have the same quality. This occurs probably because of the amount of members in common among the teams.

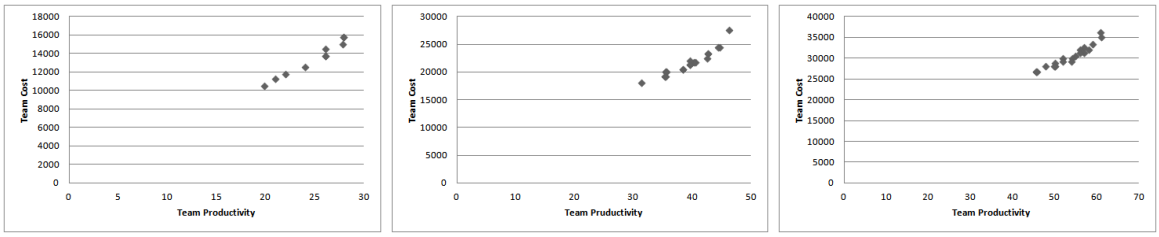
It is also possible to notice that is difficult for the project managers to choose the best team when the number of possibilities is higher. This can be seen in the teams with seven developers. The search space is big. This generates a great number of possible solutions. Because of this is hard to find the best solution without a computational help. We can notice that the suggestion of our approach, in the average case, is cheapest than the cheapest team proposed by the project managers, with almost the same productivity. However, it is important to reinforce that there is no statistical difference among the mentioned productivities.

The statistical factor is a limitation in the current stage of our work. We know that an ideal evaluation of our approach requires a project development, comparing teams allocated with and without our support. But we also know that this is very expensive in the real world. However, we are planning an experimental study following the prescriptions of the Experimental Software Engineering area [24].

VIII. CONCLUSION AND FUTURE WORKS

We presented in this work an approach to solve the agile team allocation problem. The major contributions of our work are:

- An approach to evaluate developers based on a questionnaire;
- An approach to estimate developers productivity based on a fuzzy inference system and KSA evaluation;
- An approach to qualify the generated solutions based on a fuzzy inference system;
- A hybrid approach to solve the addressed problem which incorporates in all of its stages the expertise of a project manager;



(a) Team size = 3 (b) Team size = 5 (c) Team size = 7

Fig. 5. Final Pareto Fronts of the Empirical Evaluation

- An empirical evaluation of the proposed approach, comparing its application against the intuition of some project managers.

Despite the difficulty to evaluate teams allocated both by the project managers and our approach, we performed an empirical evaluation. The main results are:

- The input fuzzy inference system, which estimates the productivity of the developers, have mimicked the behavior of the domain experts;
- The variables team cost and team productivity, used as inputs of the NSGA-II module, increase in the same ratio. The better is the team productivity, the higher is the team cost.
- The teams allocated using our approach, although different from the teams allocated by the project managers, have a high degree of similarity. Besides, the teams have the same quality.

As future works, we intend to improve the empirical evaluation, as discussed before. In order to adjust the membership functions and the rule base of our fuzzy inference systems, we intend to analyze in a statistical way the choices of the project managers to allocate the teams. Additionally, we intend to evaluate others metaheuristic techniques like PSO (Particle Swarm Optimization) or GA (Genetic Algorithm) to allocate agile teams and also to adjust the fuzzy knowledge base.

IX. Acknowledgments

We would like to thank CNPq (grant 560.128/2010) and INES, for partially supporting this work. Also, we would like to acknowledge the Infoway company by the important contributions to the research.

References

[1] R. Pressman, *Software Engineering: A Practionerís Approach*. McGraw Hill, 2006.
[2] I. Sommerville, *Software Engineering*. Addison-Wesley, 2010.
[3] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice Hall, 2001.
[4] M. R. Barrick, G. L. Stewart, M. J. Neubert, and M. K. Mount, "Relating member ability and personality to work-team processes and team effectiveness," *Journal of Applied Psychology*, vol. 83, no. 3, pp. 377–391, June 1998.
[5] K. Heldman, *PMP Project Management Professional Exam Study Guide*. Sybex, 2009.
[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.
[7] Q. Yang, G. He, and L. Li, "Application of genetic algorithm on human resources optimization," in *2010 International Conference on E-Business and E-Government*, 2010.
[8] X. Shan, G. Jiang, and T. Huang, "The optimization research on the human resource allocation planning in software projects," in *International Conference on Management and Service Science (MASS)*, 2010.
[9] D. P. Coutinho, F. G. Freitas, R. A. F. Carmo, C. L. B. Maia, and J. T. Souza, "A multi-objective approach to the team allocation problem (in portuguese)," in *2010 XLII Brazilian Symposium on Operational Search*, 2010.
[10] M. Harman and B. F. Jone, "Search-based software engineering," *Information and Software Technology*, vol. 43, pp. 833–839, 2001.
[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2001.
[12] E. H. Mamdani, "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis," *IEEE Transactions on Computers*, vol. 26, no. 12, pp. 1182–1191, 1977.
[13] M. Di Penta, M. Harman, and G. Antoniol, "The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study," *Software Practice and Experience*, vol. 41, pp. 495–519, April 2011.
[14] F. Daojin, "Research on the comprehensive evaluation of the human resource allocation based on analytic hierarchy process and fuzzy mathematics," in *2010 2nd International Conference on Industrial and Information Systems*, 2010.
[15] C. Ning, "The application of neural network to the allocation of enterprise human resources," in *2nd International Conference on e-Business and Information System Security (EBISS)*, 2010.
[16] A. Certa, M. Enea, G. Galante, and C. M. L. Fata, "Multi-objective human resources allocation in R & D projects planning," *International Journal of Production Research*, vol. 47, 2009.
[17] L. Zadeh, "Fuzzy Sets*," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.
[18] L. A. Zadeh, "Fuzzy logic = computing with words," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 2, 1996.
[19] Y. Sawaragi, H. Nakayama, and T. Tanino, *Theory of Multi-objective optimization*. Academic Press, 1985.
[20] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "Mocell: A cellular genetic algorithm for multiobjective optimization," *International Journal of Intelligent Systems*, vol. 24, pp. 726–746, 2009.
[21] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pp. 95–100, 2002.
[22] L. B. Leal, M. V. S. Lemos, R. Holanda Filho, R. A. L. Rabelo, and F. Borges, "A hybrid approach based on genetic fuzzy systems for wireless sensor networks," in *IEEE Congress on Evolutionary Computation (CEC)*, june 2011, pp. 965 –972.
[23] T. J. Ross, *Fuzzy Logic with Engineering Applications*. Wiley, 2004.
[24] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and W. A., *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, 2000.