Integrating managerial preferences into the qualitative multi-criteria evaluation of team members

Ann Barcomb, Nicolas Jullien, Patrick Meyer, Alexandru Liviu Olteanu

Integrating managerial preferences into the qualitative multi-criteria evaluation of team members

Ann Barcomb, Nicolas Jullien, Patrick Meyer and Alexandru-Liviu Olteanu

Abstract Managers can find it challenging to assess team members consistently and fairly. The ideal composition of qualities possessed by good team members depends on the organization, the team, and the manager. To enable managers to elucidate the qualities they require, we make use of an innovative methodology. This methodology is based on a multi-criteria decision aiding process, starting with the identification and definition of the dimensions that will be used to evaluate team members, then the inference of the manager's preferences through a multi-step protocol combining multiple types of preference models, and finally extracting a set of rules that can support the manager in his/her tasks. We illustrate this methodology in the case of free/libre/open source software development teams, where we were able to elicit the characteristics of a good, acceptable or bad contributor based on multiple managers' perspectives. We additionally provide an example on how to reproduce this experiment using the MCDA package for the R statistical environment.

Ann Barcomb
Lero - The Irish Software Engineering Research Centre, University of Limerick, Ireland, e-mail: ann@barcomb.org

Nicolas Jullien
IMT Atlantique, LEGO-M@rsouin, Univ. Bretagne Loire, F-29238 Brest, France e-mail: nicolas.jullien@imt-atlantique.fr

Patrick Meyer
IMT Atlantique, Lab-STICC, Univ. Bretagne Loire, F-29238 Brest, France e-mail: patrick.meyer@imt-atlantique.fr

Alexandru-Liviu Olteanu
Lab-STICC, CNRS, Université de Bretagne Sud, 17 Bd. Flandres Dunkerque, Lorient, France e-mail: alexandru.olteanu@univ-ubs.fr

1 Introduction

The field of Multiple Criteria Decision Aiding (MCDA) has been growing rapidly over the past decades, which has resulted in the proposal of various streams of thought and methodological formulations to solve complex decision problems. In particular, many so-called MCDA methods have been proposed in the literature and are very often available as software tools. Implementing a decision aiding process in practice, however, is still a difficult endeavor, often limited by the expertise of the analyst who guides it, the tools that are available, and their ability to interface with one another as well as with those from other fields of research. Nevertheless, the benefits of using MCDA approaches—integrating the preferences of the end user and providing recommendations that are easily understood and which lead to actionable outcomes—outweigh these difficulties.

The purpose of this chapter is to:

- illustrate the implementation of a decision aiding process from start to finish in the context of integrating managerial preferences into the qualitative evaluation of team members,
- promote the use of the `MCDA` package, which aims at providing a repository of MCDA methods that can easily be linked together within the statistical programming language R,
- serve as a blueprint for implementing similar decision aiding processes within other application domains.

The readers will be able to familiarize themselves, in Section 2, with the teaming problem within the free/libre/open source software (FLOSS) domain and the methodology of extracting and then selecting the criteria used to evaluate team members. In Section 3 they will gain a general understanding of the field of MCDA, the preference models and methods that will be used in the illustrative examples as well as how to set up the `MCDA` package for R. Section 4 introduces them to the iterative protocol of inferring the managerial perspective on the team member evaluations, while Section 5 follows this protocol for two different managers. The case studies from Section 5 are accompanied by the corresponding R code, which is carefully explained and may additionally be used to replay the experiment on a personal computer. Finally, the discussion from Section 6 offers additional insights into implications of the presented approach and its potential to subsequently support the managerial decision-making process. Conclusions as well as perspectives for future work are also provided in the end of this chapter.

2 A specific case: FLOSS teams

2.1 The teaming problem

The competitive advantage of firms has, since the seminal works of Kogut and Zander, and Locander et al. [35, 38], been deemed to lie in their capacity for organizing the sharing and transfer of knowledge of individuals and groups within the organization. More specifically, Grant [24] asserted that the organization does not create knowledge, but selectively identifies and coordinates the specialist knowledge embedded within its employees. This is true not only at the scale of a firm, or within the boundaries of a firm, but also at project level as the goal of the organization becomes to identify, select and coordinate, dynamically, the knowledge capacities required for each project [66] through efficient teaming [73].

An example of such dynamic teamings are free/libre/open source software projects (FLOSS), where teams are built on voluntary and competence-based criteria [21]. Today, for efficiency and flexibility, this structure is copied by firms in two ways: by outsourcing part of their intellectual production through open innovation or crowdsourcing, which requires internal reorganization [65]; or through inner source, where the firm enables voluntary cross-team collaboration within the organization in order to internally replicate the benefits of FLOSS methods of production [11, 64].

However, it may be difficult for companies to effectively implement changes to take advantage of FLOSS methodology within the firm, both because of the company culture and because of the challenge it creates for managers, who may have difficulty accounting for and evaluating their employees' work, but also in signaling which competencies they are looking for.

Information and decisions systems may be of tremendous help to pre-select the best candidates, in terms of technical adequacy and personal match to a team and a project, as shown by Malinowski et al. [39]. But, as these authors also stressed, the final decision may remain in the hand of the team leaders, or team managers, as they are those who make the team work; leadership is expressed through selection and motivation of good team members [25]. In fact, Wageman [71] found that while design choices and hands–on coaching both impact the well–being of self–managing teams, only leaders' design activities affect task performance. It is also important that any automating pre-selecting staffing system be accepted by the manager, as "One reason for greater resistance to global staffing systems is the 'human factor.' Managers tend to be resistant to anything that might override or lead to a different conclusion than their own judgment on a 'people issue'" [58].

In the view of the firm as a system to manage knowledge via ad-hoc, and increasingly virtual, teams, the project manager is the "chief executive" of the "temporary organization" [68]. The manager is responsible not only for motivating team members—which can be especially challenging in a global, distributed environment [5, 52]—but also selecting them, as team assembly mechanisms can determine team performance [25]. This is especially true in creative teams such as those engaged in building knowledge commons [26]. For instance, both the quality of the team

and good management are known as important motivation factors in virtual software development teams [6], while poor feedback, inequity and unfair rewards are demotivating and of particular concern in this environment [16]. Consequently, the best way to improve team efficiency may be to provide the leaders of teams with a methodology to systematically and transparently describe the characteristics which matter to them in the evaluation, selection, and development of team members.

The precise elicitation of key characteristics for team members, from the perspective of the manager, is at the core of what we propose in this chapter, rather than other factors contributing to efficiency. In this we follow the shift, especially in virtual team research, from skills and abilities of people working in virtual teams to team composition [22], specifically the skills and attributes a specific manager values in the composition of a particular team.

Having a methodology which allows the manager to weigh the relative benefit of the characteristics they really look for when assessing their team members would have several potential managerial benefits. First, in a self-selected team, similar to the FLOSS scenario, individuals could determine if they were a good match for the team based on how well they aligned with the desired profile. Alternately, in an assigned team, dynamic team composition could be constructed based on necessary qualities.

Second, project managers could explicitly identify the qualities they value in their team and use this to steer development. Coaching individuals in areas where improving their skills would most benefit their participation in the team, articulating why certain individuals are problematic, and developing plans for improvement or expulsion are examples of steering enabled by assessment. In a self-selected team, as in the FLOSS scenario, members may not be compelled to join or required to improve their performance, but they may still be asked to leave by a leader. This method could be used to help articulate why a person's contributions are unsatisfactory. Finally, a transparent exposition of desired attributes provides the manager with a non-arbitrary way of expressing preferences, leading to greater opportunities for consistency and fairness.

2.2 The specific case of FLOSS teaming

Before proceeding to the main contribution of this work, we offer an explanation as to why we used the case of free/libre/open source software (FLOSS) software development teams to illustrate the application of the method.

Firstly, FLOSS is considered a model for firms employing virtual teaming for the production of knowledge (in this case, software), especially in the Information Systems literature [21]. The fact that social skills in conjunction with leadership behavior affect team motivation and performance has already been described in general and are highly applicable to the context of software development (see, for instance, [43]). As we have previously stated, FLOSS development methodology is also highly applicable to companies.

Secondly, as already said, it is known that managers vary regarding their criteria of evaluation of team members. As our claim is that our technique helps detecting those differences, it was necessary to choose a domain where it has been clearly established that there is something to detect, and FLOSS is a place where such differences have been documented.

Taking the example of communication skills, certain behaviors can be deemed 'toxic' in some projects, but completely acceptable in others [12]. The difference in perception can be seen in public statements by FLOSS team leaders. For example, the Dreamwidth community prides itself on its inclusivity[1], while Linus Torvalds, the founder of Linux, has famously said, "Talk is cheap. Show me the code,"[2] suggesting that technical capability/proposals are highly valued in his community over other attributes.

Finally, existing processes for evaluating team members in FLOSS groups have clear limitations. Most widespread is the idea of meritocracy, where the best developer is easily recognized by the quality of contribution, and gains the greatest recognition through evaluation by peers [60]. However, meritocratic cultures have been demonstrated to deliver biased observations [13], and FLOSS communities have been specifically criticized for this shortcoming [50, 55]. FLOSS community managers wrestling with the difficulty of identifying "poisonous" behaviors [12], or facing decisions about the allocation of voting rights—a scenario posed by one of our participants—can benefit from a methodology which aids in articulating the ways in which a person succeeds or fails to perform within the team.

Consequently, for the rest of the chapter, FLOSS contributors will be described as team members, and community managers as decision makers (DM).

2.3 Universal team member characteristics

Characteristics of team members can be categorized into technical "multifunctional knowledge" and the "teamwork capability of team members" such as communication skills which combine with other factors—such as the "working relationship" between teammates—to bring knowledge together [14]. The study of Bassellier and Benbasat [4] uses the categories "business competence" and "IT competence" to describe, respectively, organization-specific knowledge and more generic, transferable skills. We opted to use the classification system which distinguishes between universal human qualities, namely psychological factors, and domain-specific technical skills.

The work of Chen and Lin [14] proposed five indicators of team member characteristics based on the Myers-Briggs [48] psychological indicators. However, these are

[1] Denise Paolucci and Mark Smith, Web 2.0 Expo SF, 2010, `https://www.slideshare.net/dreamwidth/build-your-own-contributors-one-part-at-a-time`.

[2] Linux kernel mailing list. Google Groups, `https://groups.google.com/d/msg/fa.linux.kernel/iQtWFALi4JA/eSzv64_tOvoJ`. Other quotations from L. Torvalds may be found in Wikiquote: `https://en.wikiquote.org/wiki/Linus_Torvalds`.

coarse-grained, and incorporate some measure of technical skills with descriptors such as "functional expertise." Consequently, we relied on the personality indicator method described by Driskell et al. [20]. This work examined the relationship between team member personality traits and team effectiveness, basing the traits on the ten personality indicators (BFI-10) [40]. The five indicators which describe the relationship between team members and team effectiveness are: *communication*, *commitment*, *collaboration*, *handling of pressure*, and *creativity* [20]. The set of psychological factors is considered complete for describing traits associated with group collaboration, and has been used not only to describe work teaming, but also a wide range of human interactions from team sports [2] to social behavior among musicians [69]. Technical skills, on the other hand, are far more domain-specific, and must thus be customized to allow the solution to be applied to different teaming contexts.

Measuring these psychometric indicators using the abbreviated BFI-10 scale normally requires a number of questions for each indicators. However, Rammstedt and John [54] showed that a ten-question instrument provides a robust measure of the indicators. The questions actually represent five indicators (the big five) in both negative and positive incarnations. It can be reduced to a set of five indicators with negative and positive descriptions, when time is a constraint and the research does not rely on a precise measure [23], as in our case. We will return to these indicators, with examples of their negative and positive descriptions, in Table 2 when they are merged with the technical indicators to create the final set of criteria.

Because technical abilities are more variable and domain-specific, they are detailed in Section 2.4.

2.4 Domain-specific team member characteristics

In order for a DM to assess a team member on an ordinal scale, it is first necessary to identify the characteristics on which this evaluation is based. In Section 2.3 we listed five psycho-sociological dimensions which can be universally applied to teaming situations. These characteristics are meant to be used in conjunction with the technical attributes, which vary by domain.

Our method for identifying the technical attributes consists of three steps. This methodology begins with a mainstream step but the third step is distinct to the problem we are addressing with MCDA techniques.

1. **Defining the team members' evaluation dimensions according to the literature**
   The literature on skills required for FLOSS development is sparse, and none of the three main attempts we identified [3, 15, 34] was suitable for our purpose, for two reasons. First, they are not at the same level of granularity as the psychological factors, which we incorporated because they are universal and well-established by research, and therefore can be applied to any teaming situation. Second, the

lists are too lengthy to allow participants to keep all attributes in mind. Considering other domains risked diluting the specific situation we wished to observe. Therefore, we synthesized the aforementioned literature to nine proposed dimensions summarizing the technical skills. Since the next stage involved validating the criteria, we were not concerned with overlooking a critical factor.

The proposed technical dimensions were: *produces code of quality*; *understands the architecture of the code (modular code, code dependencies)*; *proposes implementation of new features without disturbing others*; *contributes a lot of commits*; *documents the work produced*; *writes clearly, in good English*; *implements the feedback received*; *is good at describing bugs*; and *is good at specifying features*.

2. **Confirming these findings with the managerial point of view**
In the second step, a number of DMs with experience in the domain are asked which attributes they consider relevant for identifying a team member's value to the group. We make use of interviews in order to gain an understanding of the DM, the language and context of the decision [49].

After giving their responses, DMs are then supplied with a list of characteristics the researchers have gleaned from the literature and asked to indicate how relevant these characteristics are. Finally, they are asked if they want to add any characteristics to their initial list. This allows the participant to first consider the question without being primed, then to evaluate the criteria from literature, and finally to reflect on the initial thoughts.

The interview responses are compared to the initial list derived from the literature. Where the DMs described existing criteria using different terms, the names or descriptions of the criteria are updated to reflect the language of the participants. If any new criteria are identified, this may warrant additional study to understand why it was not previously described in the literature. When a particular criterion is not valued by any participants, the criterion could be dropped to reduce the number of dimensions to be considered.

Based on the interviews, we concluded that our list was sufficiently comprehensive. Furthermore, four of the technical criteria were considered unimportant by all participants. Because our experts did not consider these dimensions relevant to the evaluation of team members, we dropped them from our list.

This left us with a set of ten dimensions—five psycho-social and five technical—which applied to all DMs. These are listed in Table 1. The terms in this table, and which are used hereafter, reflect the language of our participants, and are therefore different from the formal terms given in Section 2.3 or identified in the previous step.

3. **Eliciting the managerial preference model on the team members evaluation based on the defined dimensions**
   The set of ten dimensions reflected the full set of factors considered important to all DMs we interviewed. However, not all DMs were interested in all the dimensions. Therefore, for each DM, we identified a working subset of dimensions of interest, taking those criteria ranked as important or very important. As each DM ranked at least half of the factors as relevant, it would not have been possible for the unselected factors to collectively deliver more relevance than the chosen factors even if all were ranked equally. The primary reason for limiting factors was to improve the DM's ability to effectively consider model profiles.

   The exact choices of the DMs are discussed in more detail in the next section and described in Table 2. This result advocates for a manager-based methodology, where each DM is able to identify a tailored subset of key factors from a longer list of attributes relevant to the domain.

Table 1 Detail of the full ten dimensions used to evaluate the team members

Variable	**Description**	
	Negative	*Positive*
Psycho-Social		
Communication skills (signal over noise ratio)	Too much noise / not enough information	Is good at providing the right level of information
Commitment to the project	Unmotivated/passive in seeking answers	Is motivated and does a thorough job
Working with others	Tends to find fault with others	Is generally trusting, patient with people
Pressure and stress related managing capacity	Gets nervous & stressed easily (ex.: when things do not go as expected, when there are delays or due deliverables)	Is relaxed, handles stress, technical limitations, setbacks well
Creativity	Not very creative in terms of solution	Has an active imagination, proposes creative ideas/solutions
Technical		
Quantity of code contributed	few lines of code	an impressive quantity of code
Quality of code contributed	Tends to provide incomplete or inferior solutions	Produces efficient & well written code, without disturbing other part of the code
Global picture: understands the tools/ technology/ domain, & processes behind the project	Low, does not understand beyond the talks/the modules addressed	Understands the technical & non-technical fundamentals of the project
Documentation & testing	Does not document/test the code produced, or does so in a way not understandable by others	Documents/test well and clearly the code produced
Contribution on other aspects than code (new features, bug description)	Does not contribute beyond code production	Very active in proposing new features, tracking and documenting bugs, etc.

2.5 The manager's perspective

A total of six DMs participated in the first part of our experiment. Table 2 illustrates the criteria each of them selected as 'important' or 'very important' according to the second step of the proposed methodology.

Table 2 Important Criteria for DMs.

Criteria	Decision makers					
	DM1	DM2	DM3	DM4	DM5	DM6
Communication skills			✓		✓	✓
Commitment to the project	✓	✓	✓	✓	✓	✓
Working with others	✓	✓	✓	✓	✓	✓
Pressure and stress related managing capacity				✓	✓	✓
Creativity						✓
Quantity of code produced				✓		
Quality of code produced	✓	✓	✓		✓	✓
Global picture	✓	✓	✓	✓	✓	✓
Documentation and testing	✓	✓	✓	✓	✓	✓
Contributions on other aspects			✓	✓		

Each of the dimensions were considered useful by at least one DM, while no DM felt that all of the criteria were relevant. We observe that the dimensions related to the commitment of contributors, their capacity to work together, their perspective of the project as a whole and their ability to document their work were unanimously considered important factors by all DMs. Furthermore, most were also concerned with the quality of code produced.

This result supports the premise of the article: if a set of variables can describe what is observable and important for evaluating a team member, managers will only use a subset of these variables. Furthermore, the selected subsets are rarely the same, and even when two DMs choose the same criteria, the resulting preference models are unlikely to be the same.

3 MCDA techniques

In a case where the criteria are known and are the same for every unit of analysis, classical data analysis methodologies such as linear regression or decision tree modeling can be used. The reader who is interested in a review of these findings may consult Driskell et al. and Stewart [20, 63]. However, these results do not deliver actionable managerial recommendations because there is no single ideal team composition which can be applied to all groups working on all tasks, even within a single domain. As we demonstrated earlier, managers value a different subset of criteria. Due to our interest in team composition, which is typically the purview of the manager, our method strives to elicit one perspective—the manager's—rather

than incorporating impressions from multiple team members as in Sarker et al.'s [59] approach involving identifying the extent to which a team member is valued by her or his peers.

In other words, to improve the efficiency of teaming, we propose to develop a non arbitrary, non ad-hoc evaluation scale based on multiple criteria, which encapsulates the perspective of the manager. When multiple, potentially conflicting evaluation criteria must be taken into account, and the individual preferences of the manager should be captured, it is natural to turn towards Multi-Criteria Decision Aiding (MCDA) techniques.

3.1 An introduction to MCDA

MCDA is the study of decision problems, methods and tools which may be used in order to assist a DM in reaching a decision when faced with a set of alternatives, described via multiple—often conflicting—properties or criteria.

Usually, three types of decision problems are put forward in this context [57]:

- the *choice problem*, which aims to recommend a subset of alternatives, as restricted as possible, containing the "satisfactory" ones;
- the *sorting problem*, whose goal is to assign each alternative into predefined ordered categories;
- the *ranking problem*, which orders the alternatives by decreasing degree of preferences.

Various models have been proposed to support DMs facing a multi-criteria decision problem [10, 33, 57] and to represent their preferences. Roughly speaking, they originate from two methodological schools. First, in the *outranking* methodologies, any two alternatives are compared pair-wisely on basis of their evaluations on the set of criteria, according to a majority rule. More precisely, a DM could consider that an alternative a *outranks* an alternative b when a *weighted majority* of criteria validates the fact that a is performing at least as good as b and there is no criterion where b seriously outperforms a. The majority-related condition is called concordance, whereas the second condition (veto) is called discordance. From a computational point of view, various implementations of these conditions, and their conjunction, have been proposed in the literature (see for example [57]). Second, methods based on *multiattribute value theory* aim to construct a numerical representation of the DM's preference on the set of alternatives. The main difference between these two methodological schools lies in the way in which the alternatives are compared and the type of information required from the DM. Outranking methods are preferred if the evaluations of the alternatives are primarily qualitative, if the DM would like to include a measure of imprecision about personal preferences in the model, and when a human-readable evaluation model is desired. Value-based methods can be favored if a compensatory behavior of the DM should be modeled, and when the evaluation of the alternatives should be summarized by a single value (as in the case

of accounting, for instance). These methodologies are usually integrated in a more general decision aiding process, as described in Tsoukiàs [67].

In the context of this chapter, the **DM** is the **team manager**, and the **alternatives** are the **team members**, while the evaluation **criteria** are the **attributes used to evaluate the team members**. Our aim is to evaluate the team members by building a transparent and understandable evaluation model which integrates the manager's preferences. The evaluation of team members relies on qualitative criteria. Furthermore, we require that the overall evaluation also delivers a qualitative scale. Last but not least, in order to develop a functional tool for team member evaluation, we require a readable evaluation model which enables the manager to understand the strengths and weaknesses of the team members, while providing details on improvement. As argued by [31], where possible, simple preference models should be used. All of these arguments are in favor of outranking preference models, more specifically an **outranking sorting technique**.

3.2 The Majority-Rule Sorting model

Among the possible outranking sorting methods, we chose Majority-Rule Sorting (MR-Sort) [37], which is a simplified version of the classic ElectreTRI [56]. This technique is close to the version axiomatized in Bouyssou and Marchant [8, 9] and does not consider the previously mentioned discordance or veto principle. The method allows us to build an overall qualitative evaluation scale for the evaluation of the contributors, while presenting a very readable and operational model.

The MR-Sort procedure takes thus into account the preferences of the DM, which are represented through the following preferential parameters :

- **criteria weights** (w), which give the relative importance of criteria;
- a **majority threshold** (λ), which indicates the weight of a coalition of criteria in order to be considered sufficient;
- **category limits** (b), which are used for comparing with an alternative leading to its assignment.

Let us consider a finite set of alternatives A, a finite set of criteria indexes J and a set of category limits separating profiles $B = \{b_1, \dots b_{k-1}\}$. Each alternative and each category limit is a vector of evaluations with respect to all criteria. The evaluation with respect to criterion j can be viewed as a function $g_j : A \cup B \to \mathbb{R}$, where $g_j(a)$ denotes the evaluation of alternative $a \in A$ on criterion j and $g_j(b_h)$ denotes the evaluation of category limit $b_h, \forall h \in \{1, \dots, k-1\}$, on criterion j. The set of category limits are used to define a set of k categories $\{c_1, \dots, c_k\}$, ordered by their desirability, from c_1 being the worst category to c_k being the best one. Each category c_h is defined through its upper limit, b_h, and its lower limit, b_{h-1}, with the exception of the worst and best categories, which have only one limit. We assume, without loss of generality, that the performances are supposed to be such that a higher

value denotes a better performance. Furthermore the performances of the category limits are non-decreasing, i.e. $\forall j \in J, 1 < h < k : g_j(b_{h-1}) \leq g_j(b_h)$.

An alternative a is said to outrank a category limit b_{h-1} if and only if there is a sufficient coalition of criteria supporting the assertion "a is at least as good as b_{h-1}". A coalition of criteria corresponds to a subset of criteria which "agree" on how an alternative compares to a category limit, either being at least a good or strictly worse. To measure this, we define for each criterion j a function $C_j : A \times B \to \{0, 1\}$ which assesses whether criterion j supports that statement or not:

$$\forall j \in J, a \in A, 1 \leq h \leq k : C_j(a, b_{h-1}) = \begin{cases} 1, \text{ if } g_j(a) \geq g_j(b_{h-1}), \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

To assess whether a coalition of criteria is in favor of the outranking or not, $\forall a \in A, 1 \leq h \leq k$, we first define the overall concordance as:

$$C(a, b_{h-1}) = \sum_{j \in J} w_j C_j(a, b_{h-1}), \tag{2}$$

where w_j is the weight of criterion j. The weights are defined so that they are positive ($w_j \geqslant 0, \forall j \in J$) and sum up to one ($\sum_{j \in J} w_j = 1$). This overall concordance is then compared to a majority threshold $\lambda \in [0.5, 1]$ extracted from the decision-maker's preferences along with the weights. As we do not consider any veto rule here, the outranking relation S is then defined as:

$$a \,\mathrm{S}\, b_{h-1} \iff C(a, b_{h-1}) \geqslant \lambda. \tag{3}$$

If $C(a, b_{h-1}) < \lambda$, the coalition of criteria is not sufficient, the alternative does not outrank the frontier b_{h-1} and will therefore be assigned in a category lower than c_h.

Alternative a is assigned to the highest category it outranks, hence this rule can be written as:

$$a \in c_h \iff a \,\mathrm{S}\, b_{h-1} \text{ and } a \not\mathrm{S}\, b_h. \tag{4}$$

In order for this assignment rule to be used for the limit categories, two dummy profiles b_0 and b_k need to be added to B, the first holding the lowest possible evaluations on all criteria, while the second holding the highest possible ones.

We provide a simple illustrative example for this assignment rule in Figure 1. We consider three team members denoted with TM_1, TM_2 and TM_3, who are evaluated with respect to five criteria with ordinal scales corresponding to their performance levels: good (g), neutral (n) and bad (b). These scales are numerically encoded as -1 for b, 0 for n and 1 for g. We wish to assign these team members to either one of three categories: c_1 (bad), c_2 (medium), c_3 (good).

At the top we observe the parameters of the sorting model. The first parameter (λ) is the majority threshold, followed by the vector of criteria weights (w) and two category separating profiles (b_1 and b_2). Below the parameters we illustrate the assignment of three alternatives using the assignment rule from Equation (4). In the case of the first and last categories we have added two fictive limits in order to simplify

$\lambda = \frac{3}{5} \quad w = (\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}) \quad b_1 = (n, n, n, n, n) \quad b_2 = (g, g, g, g, g)$

A	g_1	g_2	g_3	g_4	g_5	$a_i \,\mathrm{S}\, b_0$	$a_i \,\mathrm{S}\, b_1$	$a_i \,\mathrm{S}\, b_2$	$a_i \,\mathrm{S}\, b_3$	assignment
SC_1	g	g	g	n	b	✓	✓	✓	×	c_3 (good)
SC_2	g	n	n	n	g	✓	✓	×	×	c_2 (medium)
SC_3	b	b	g	b	n	✓	×	×	×	c_1 (bad)

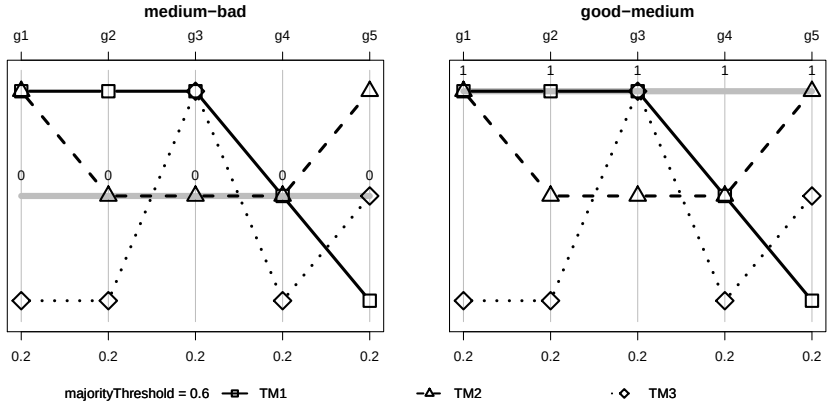

Fig. 1 Illustrative example for the MR-Sort assignment procedure.

the example. The lower category limit of the worst category (b_0) is considered to have the lowest possible evaluations on all criteria (in this case b), while the upper profile of the best category (b_3) is considered have the evaluations that are always better than all alternatives in A on each criterion. TM_1 is at least as good as b_1 on 4 out of the 5 criteria, and it is at least as good as b_2 on 3. Therefore TM_1 outranks both b_1 and b_2, and is therefore assigned to the highest category. The second team member is at least as good as b_1 on all criteria, however it is at least as good as b_2 on only the two. Therefore TM_2 outranks b_1 but not b_2 and is therefore assigned to the second category. Finally, the last alternative is at least as good as b_1 on only two criteria and is therefore assigned to the lowest category. The illustration serves as a visual aid, containing two plots of the relation between the three team members and the profiles separating each consecutive pair of categories. The separating profiles are plotted in black with filled round markers, while the team members are colored based on their category assignment and each contain a different marker.

3.3 The Majority-Rule Sorting model with large performance differences

In order to model more accurately the preferential behavior of the DM, several extensions of the MR-Sort model have been proposed in literature. We focus on the work of Meyer and Olteanu [41], who extended this approach in order to handle large positive performances (resulting in what is called dictator effects). This work shows that various options of combining the veto and dictator effects lead to different preference models. These extensions lead to more flexible and more complex models as their number of parameters increases. Consequently, they allow the modeling of increasingly complex preferential statements of the DM.

The first type of large performance difference corresponds to the classical veto concept used in outranking relation. As we have previously stated, an alternative a is said to outrank a category limit b_{h-1} if and only if there is a sufficient coalition of criteria supporting the assertion "a is at least as good as b_{h-1}". However, even when the coalition is strong enough, a criterion may veto the outranking situation. An alternative a is in a veto relation (denoted with V) with a profile b_{h-1} when:

$$a \,\mathrm{V}\, b_{h-1} \iff \exists j \in J : g_j(a) \leqslant g_j(b^v_{h-1}). \tag{5}$$

A veto profile b^v_{h-1} is added to the MR-Sort method, and represents the maximal level of performance that that is considered unacceptable for an alternative to be allowed into category c_h via the weighted coalition of criteria in favor of this assignment. If on any criterion an alternative has at most this level of performance, then it is forbidden from being assigned to c_h or above. The following constraint on the evaluations of a veto profile b^v_{h-1} needs to hold: $g_j(b^v_{h-1}) < g_j(b_{h-1}), \forall j \in J, h \in 1..k$. Furthermore, the performances on the veto profiles are non-decreasing, i.e. $\forall j \in J, h \in 1..k+1 : g_j(b^v_{h-1}) \leq g_j(b^v_h)$. In addition, the performances of the top most veto profile will be fixed to be equal to those of the top most category profile, so that no veto situation may be triggered in this case.

To summarize, alternative a outranks the category limit b_{h-1} (and therefore is assigned to at least the category c_h) if and only if:

$$a \,\mathrm{S}\, b_{h-1} \iff C(a, b_{h-1}) \geqslant \lambda \text{ and not } a \,\mathrm{V}\, b_{h-1}. \tag{6}$$

We take the illustrative example from Figure 1 and add two additional veto profiles in order to illustrate the assignment rule of the previously presented model, as seen in Figure 2. Only a few of the veto profiles evaluations are defined in this case, while the graphical illustration shades in black the value ranges that these profiles define as unacceptable for the upper category they separate.

The addition of the veto profiles leads to the first contributor being assigned to the medium category due to the fact that, despite having three good evaluations, it has a bad evaluation on the last criterion. b^v_2 indicates that such an evaluation is unacceptable for being assigned to the top category. All the other contributors assignments do not change, however, it should be noted that, in order for the third

$\lambda = \frac{3}{5} \quad w = (\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}) \quad b_1 = (n, n, n, n, n) \quad b_2 = (g, g, g, g, g)$
$b_1^v = (\ ,\ ,\ , b,\ ) \quad b_2^v = (\ ,\ ,\ , b, b)$

A	g_1	g_2	g_3	g_4	g_5	$a_i \,\mathrm{S}\, b_0$	$a_i \,\mathrm{S}\, b_1$	$a_i \,\mathrm{S}\, b_2$	$a_i \,\mathrm{S}\, b_3$	assignment
TM_1	g	g	g	n	b	✓	✓	×	×	c_2 (medium)
TM_2	g	n	n	n	g	✓	✓	×	×	c_2 (medium)
TM_3	b	b	g	b	n	✓	×	×	×	c_1 (bad)

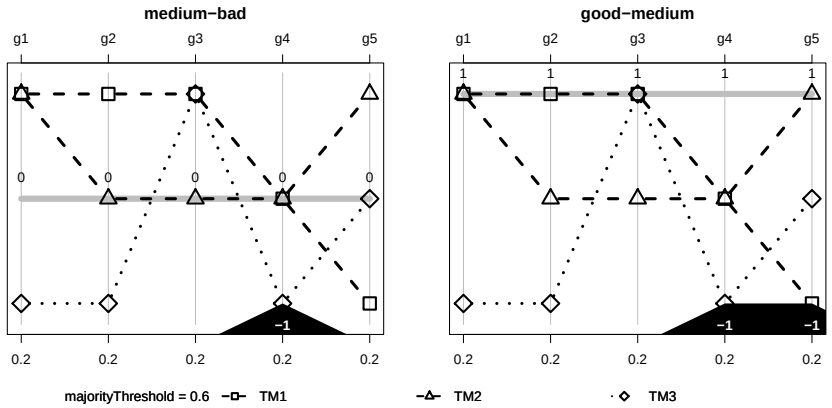

Fig. 2 Illustrative example for the MR-Sort assignment procedure with vetoes.

contributor to be allowed into the medium category, he/she should improve on the fourth criterion first, as otherwise a veto would continue to be triggered regardless of any improvements on the first two criteria (see veto profile b_1^v).

We continue by considering the dictator effect [41], which may be seen as the opposite concept of a veto. When only an insufficient coalition of criteria supports the assertion "a is at least as good as b_{h-1}", having a much better performance of a with respect to b_{h-1} on even one criterion allows us to validate the outranking relation. An alternative a is in a dictator relation (denoted with D) with a profile b_{h-1} when:

$$a \,\mathrm{D}\, b_{h-1} \iff \exists j \in J : g_j(a) \geqslant g_j(b_{h-1}^d). \tag{7}$$

A dictator profile b_{h-1}^d is added to the MR-Sort method, and represents the minimum level of performance that an alternative needs to have in order to be guaranteed assignment into at least category c_h. The following constraint on the evaluations of a dictator profile b_{h-1}^d needs to hold: $g_j(b_{h-1}) < g_j(b_{h-1}^d), \forall j \in J, h \in 1..k$. Furthermore, the performances on the veto profiles are non-decreasing, i.e. $\forall j \in J, h \in 1..k+1 : g_j(b_{h-1}^d) \leq g_j(b_h^d)$.

To summarize, alternative a outranks the category limit b_{h-1} (and therefore is assigned to at least the category c_h) if and only if:

$$a \,\mathrm{S}\, b_{h-1} \iff C(a, b_{h-1}) \geqslant \lambda \text{ or } a \,\mathrm{D}\, b_{h-1}. \tag{8}$$

Equations (5) and (7) correspond to MR-Sort models with vetoes and with dictators respectively. In [41], multiple assignment rules combining the veto and dictator relations have been proposed, each using a slightly modified outranking relation:

- MR-Sort with a dictator effect weakened by a veto effect:

$$a \,\mathrm{S}\, b_{h-1} \iff C(a, b_{h-1}) \geqslant \lambda \text{ or } \big(a \,\mathrm{D}\, b_{h-1} \text{ and not } a \,\mathrm{V}\, b_{h-1}\big). \tag{9}$$

- MR-Sort with a veto effect weakened by a dictator effect:

$$a \,\mathrm{S}\, b_{h-1} \iff C(a, b_{h-1}) \geqslant \lambda \text{ and not } \big(a \,\mathrm{V}\, b_{h-1} \text{ and not } a \,\mathrm{D}\, b_{h-1}\big). \tag{10}$$

- MR-Sort with a dictator effect and a dominating veto effect:

$$a \,\mathrm{S}\, b_{h-1} \iff \big(C(a, b_{h-1}) \geqslant \lambda \text{ or } a \,\mathrm{D}\, b_{h-1}\big) \text{ and not } a \,\mathrm{V}\, b_{h-1}. \tag{11}$$

- MR-Sort with a veto effect and a dominating dictator effect:

$$a \,\mathrm{S}\, b_{h-1} \iff a \,\mathrm{D}\, b_{h-1} \text{ or } \big(C(a, b_{h-1}) \geqslant \lambda \text{ and not } a \,\mathrm{V}\, b_{h-1}\big). \tag{12}$$

- MR-Sort with conflicting dictator and veto effects:

$$a \,\mathrm{S}\, b_{h-1} \iff \Big(C(a, b_{h-1}) \geqslant \lambda \text{ and not } \big(a \,\mathrm{V}\, b_{h-1} \text{ and not } a \,\mathrm{D}\, b_{h-1}\big)\Big) \text{ or } \big(a \,\mathrm{D}\, b_{h-1} \text{ and not } a \,\mathrm{V}\, b_{h-1}\big). \tag{13}$$

The first model is similar to MR-Sort with dictators, except that the dictator effect may be invalidated by a veto effect occurring at the same time. However, the veto on its own does not impact in any way the outranking relation. The second rule corresponds to MR-Sort with vetoes where the dictator invalidates the veto, but does not do anything on its own. The following three models benefit from both veto and dictator effects, but handle cases where both occur at the same time differently. On their own, a veto invalidates an outranking relation while a dictator validates it. When both occur at the same time, the model with dominating dictator gives precedence to the dictator effect, the one with dominating veto gives precedence to the veto, while the one with conflicting dictator and veto invalidates both and considers the weighted coalition of criteria in favor of the outranking.

3.4 Learning a Majority-Rule Sorting model

Several techniques have been proposed in the literature to learn the parameters of outranking-based multi-criteria sorting models as an alternative to directly asking the DM to provide them.

Mousseau and Słowiński [47] have proposed to find the entire model through the use of *assignment examples*. More specifically, the DM is asked in a first step to

assign a few *well known* alternatives to the predefined categories. Mousseau et al. [46] only sought to find the criteria importance weights with the other parameters being supposedly known, while Ngo The and Mousseau [51] only looked for the category limits. Other more robust approaches compute for each alternative a range of possible categories to which they may be assigned when the parameters of the model are not completely determined [17, 18, 19]. Approaches that deal with inconsistent sets of assignment examples leading to non existing preference model solutions have also been explored in Mousseau et al. [44, 45].

In most cases, the approaches of inferring the parameters of majority-rule sorting models use mathematical programming techniques involving binary variables, such as in Leroy et al. [37]. As these approaches find the optimal solution, they may also require large amounts of computational resources and time, making their use limited when large sets of assignment examples are considered. Sobrie et al. [62] have suggested to use a technique based on a metaheuristic to learn the parameters of the sorting model, which has been adapted and extended by Olteanu and Meyer [53] in order to additionally take into account veto thresholds. Finally, Olteanu and Meyer [41] proposed mixed integer programming techniques to learn the parameters of the previously presented extensions of the MR-Sort method.

In the context of our example, the team members are the alternatives, and the DM will assign them to categories such as "good", "medium" or "bad", based on their performance on the selected evaluation criteria (i.e. their skills). Then, from these assignment examples, the model parameters are extracted using a given algorithmic approach.

3.5 The MCDA R package as a support tool

In this section we present the software which can be used to support the previously presented method, namely the MCDA package for R. R is an open-source functional programming language and environment mainly centered around data analysis [29, 70]. In recent years it has grown in popularity with the IEEE identifying it as the 9th most popular programming language in 2014, the 6th most popular in 2015 and the 5th most popular in 2016 [28]. Due to the large community of R users, many tools in the form of functions within packages have been proposed, many dealing with handling different data formats, data pre-processing, data filtering and interactive visualizations. An interactive tutorial for learning R may be found at [1].

The MCDA package [7] that we propose to use follows the philosophy of R, by encompassing a growing array of MCDA algorithms that may be used to decompose a decision aiding process into sub-steps. The package mainly targets MCDA practitioners that are familiar with the decision aiding process, giving them the possibility to construct this process as they see fit. As very often during a decision aiding process the DM does not have a clear picture of his/her problem [61], being able to quickly adapt the process as new information is made available is of great importance. Finally, the MCDA package may benefit both MCDA practitioners and

data analysts, as MCDA practitioners could further apply methods linked to data analysis throughout the decision aiding process, while data analysts could use their data for reaching an objective in addition to analyzing it.

At the time of writing, the package is still young and consequently is far from covering all of the algorithms from the classical MCDA literature. However, functions supporting various steps of the MCDA process have been implemented in the MCDA R package. They can be categorized as follows:

- state of the art aggregation algorithms;
- state of the art preference elicitation algorithms;
- data manipulation functions;
- plot functions.

In order to be able to use this package, the user should first install either `GLPK`, `CPLEX` or both on their machine. `GLPK` is a free and open-source, while a free version of `CPLEX` can be easily acquired through the Academic Initiative of IBM. We recommend using `CPLEX` for testing the case studies that we present in Section 5, as it is much faster than `GLPK`.

On a Debian-based Linux machine, installing `GLPK` is as simple as: `sudo apt-get install libglpk-dev`.

Within R, the MCDA package and its dependencies can be installed using the following command:

```
# install the MCDA package

install.packages("MCDA")
```

The user should carefully check whether all of the package dependencies were able to be installed correctly, and otherwise proceed to installing each of them separately.

Additionally, for users wishing to make use of `CPLEX`, the `cplexAPI` package should also be installed, as the MCDA package does not list it as a direct dependency:

```
# install the cplexAPI package

install.packages("cplexAPI")
```

In certain cases, this command will not work and it might be necessary to also point towards the path where `CPLEX` is installed on your machine:

```
# install the cplexAPI package

install.packages("cplexAPI",configure.args = "--with-cplex-dir=
                                                /path/to/cplex")
```

4 A method to integrate managerial preferences into the ordinal evaluation of contributors

4.1 Selecting the criteria for the MCDA analysis

Selecting the criteria for an MCDA process requires reducing the options to a manageable level. According to Miller [42], there are limits to human capacity for processing information, which explains why the maximum number of aspects which can be simultaneously considered is around seven, plus or minus two. From our literature review and initial interviews, we were able to narrow the full set of dimensions to ten by excluding factors which were not valued by any DM.

In order to extract the subset of dimensions relevant for each DM, we asked them to rate each dimension based on their perception of its importance with one of five levels, ranging from *very important* to *not important at all*. We then included all criteria they ranked as very important or important. A balanced three-factor scale would have captured the same information, but as part of our exploration we wanted to demonstrate to the DM the similarities and differences between the off-the-cuff assessment of the relative importance of items and the more nuanced model we created.

It is worth noting that in other types of analysis of preferences, such as Q-methodology, the standard is to incorporate those factors which can explain at least 50% of the sample variance [72]. If we considered each factor equally important to the DM, incorporating five dimensions would already capture half of the variance. However, based on the DM's rankings, it was certain that by dropping the less important factors we were reducing the demands on the DM without significant loss of detail.

4.2 Inferring the DM's preferences

In Section 3, we explained why we advocated using the MR-Sort technique to model the DM preferences. To infer the parameters of such an MR-Sort preference model, the following protocol, illustrated in Figure 3, is applied.

The protocol may be used for interactively inferring an MR-Sort model for any problem. As we are interested here on applying it to the problem of inferring the preferences of a manager with respect to team members, we will consider that the alternatives (the decision objects in MCDA) correspond to team member profiles. A team member profile represents the series of evaluations of a team member across several criteria. The protocol consists of the following series of steps:

1. Generate an initial set of team member profiles A.
   In the first step, fictitious team members are generated by randomly constructing vectors of evaluations on the previously selected criteria. These team members profiles correspond to the alternatives used in MR-Sort.

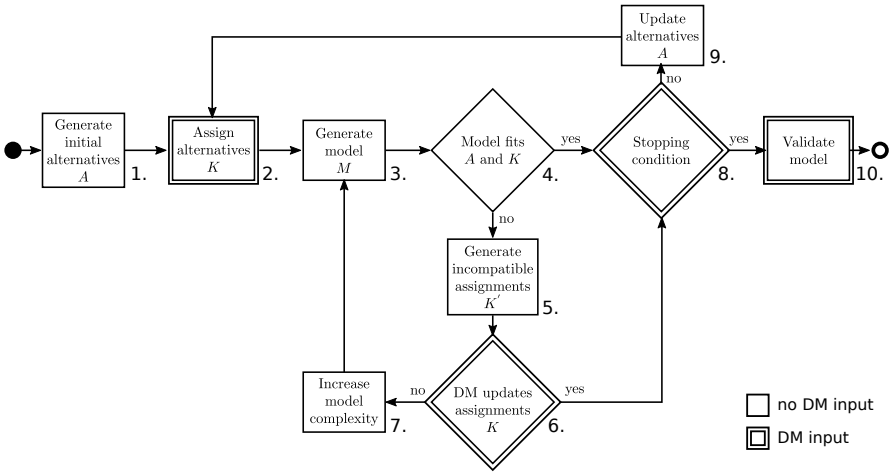

Fig. 3 Protocol for inferring the parameters of an MR-Sort model

2. The DM assigns the team member profiles to categories K.
   This corresponds to the construction of the assignment examples used during the subsequent steps to infer the MR-Sort model. The DM assigns each of the previously constructed team member profiles to one category.
3. Generate an MR-Sort model M.
   An MR-Sort model is constructed using an inference algorithm that takes as input the provided assignment examples. The initial attempt is to look for the most simple model (the classical MR-Sort model) which is able to assign the team member profiles in the way provided by the DM.
4. If the model fits then go to step 8.
   The fit corresponds to the model, through its inferred parameters and assignment rule, assigning the team member profiles in exactly the same way as the DM.
5. Generate all minimal sets of incompatible assignments.
   If the MR-Sort inference algorithm is not able to output a model that fulfills all of the assignment examples, we extract the minimal sets of team member profiles which would need to be assigned to different categories in order for a valid output to be provided.
6. If the DM accepts changing the assignments in accordance to at least one incompatible assignment set then go to step 8.
   The previously generated sets of incompatible assignment examples are presented to the DM in order to make sure that no error or hesitation were introduced. If the DM agrees to change their assignments then the MR-Sort model that was initially generated can be used further.
7. Select a more complex model and go to step 3.
   If the DM does not agree to changing their assignments in accordance to the presented incompatible assignment sets, we proceed by considering a more complex MR-Sort model. The choice of models is elaborated further below.

8. If a stopping condition is met then go to 10.
   After we are able to select a model that fits the assignment examples of the DM, or at least one that minimizes the number of incompatible assignments, we check whether a stopping condition is met. This condition may be based on subjective factors linked to the willingness of the DM to proceed further, or on factors linked to the extracted assignment rules that accompany an MR-Sort model.
9. Update the set of team member profiles A by generating additional ones.
   Additional team member profiles are randomly generated in the same way as in step 1.
10. Validate the model and finish.
    The DM is presented with the final MR-Sort model, as well as with an illustration and explanation of its assignment rules.

Regarding the choice of models in steps 3 and 7, we consider first the MR-Sort model with a simple majority rule, followed by the models which take into account a single large performance difference effect (MR-Sort with vetoes or with dictators) and only then models that take into account both. Among the latter models, we consider that the models that take into account a single effect but which may be weakened by another (MR-Sort with vetoes weakened by dictators and MR-Sort with dictators weakened by vetoes) are the least complex, followed by models that have one dominating effect (MR-Sort with vetoes dominating dictators and MR-Sort with dictators dominating vetoes), and finally the model which equally balances the two effects together (MR-Sort with conflicting vetoes and dictators). When no model completely fits the assignments, we choose based on which model is better able to fit the set of assignment examples, i.e. the one which minimizes the number of incorrectly assigned team member profiles. When this criterion is not sufficient in order to differentiate between such models, or when multiple complex models fit, we select one at random.

This incremental approach has been chosen in order to reduce the cognitive effort of the DM and also to test for hesitations or inconsistencies in their assignments. Furthermore, by starting with a simple preference model and increasing its complexity as warranted by the input of the DM allows us to lower the needed computational resources as simpler models no longer need to be tested once their are deemed insufficiently flexible for modeling the perspective of the DM.

5 Experimental case: FLOSS

Although six DMs agreed to participate in our research, three of them only participated in the criteria selection step, whereas one of the remaining three stopped mid-way through the presented protocol. We therefore illustrate below the proposed methodology for the two DMs who reached the end of our proposed methodology.

Both of these DMs selected the same five criteria in order to evaluate their contributors, therefore we present them below:

- Commitment to the project (g_1), psycho-sociological;
- Working with others (g_2), psycho-sociological;
- Quality of code produced (g_3), technical;
- Global picture: understands the tools/technology/domain, & processes behind the project (g_4), technical;
- Documentation & testing (g_5), technical.

For each of these criteria, a five-level ordinal scale ranging from very bad to very good ($\{vb, b, n, g, vg\}$) was selected, as well as a final three-level ordinal evaluation scale ($K = \{Bad, Neutral, Good\}$). The scale evaluations are numerically denoted from -2 for vb up to 2 for vg.

5.1 Inferring the preferences of DM1

We begin with illustrating the inference of the first decision maker's (DM1) preferences. We will additionally show how the proposed protocol was implemented in R.

The entire R script for this case study is available in the MCDA package using:

```
# path to the R script of the example

system.file("examples","exampleFLOSS.R",package="MCDA")
```

We started by defining the criteria and the categories:

```
# load the MCDA library

library(MCDA)

# define the criteria and their preference directions

criteria <- c("c1","c2","c3","c4","c5")

criteriaMinMax <- c("max","max","max","max","max")

# define the categories and their ranks

categories <- c("Good","Neutral","Bad")

categoriesRanks <- c(1,2,3)

names(categoriesRanks) <- categories

names(criteriaMinMax) <- criteria
```

We then continued with the first step of the model inference protocol by generating an initial set of 25 team member profiles. The code that generates these profiles is presented below.

```
# generate the initial set of unique alternatives
```

```
pT <- unique(matrix(sample.int(5, 1000, T), 100, 5))[1:25,] - 3

rownames(pT) <- 1:dim(pT)[1]

colnames(pT) <- criteria
```

In order for the reader to be able to reproduce our experiment, the provided R script does not include this code, but simply loads step by step the profiles that were generated using this method.

Given the 25 generated team member profiles, we asked DM1 to assign them to one of the three categories, while explaining his reasoning to one of the chapter's authors. DM1's assignments are illustrated in Table 3. This corresponds to step 2 in the proposed protocol.

Table 3 The initial set of contributor profiles and their assignment by DM1;

Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Assigned category	Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Assigned category
1	vg	g	vb	vg	n	Bad	14	vg	n	vb	b	g	Bad
2	b	vg	n	vb	n	Neutral	15	n	b	b	vb	n	Bad
3	b	b	b	b	g	Bad	16	b	vg	g	vg	vb	Bad
4	b	b	vb	vg	n	Bad	17	n	b	n	g	n	Bad
5	g	vb	vg	b	b	Neutral	18	vg	vb	vg	g	b	Neutral
6	vg	g	vg	n	vg	Good	19	vg	vb	n	n	vb	Bad
7	g	n	b	n	vg	Neutral	20	vb	vg	vg	b	vg	Neutral
8	n	n	g	b	g	Good	21	vb	n	vb	n	vb	Bad
9	n	vg	n	g	b	Bad	22	vb	b	vb	vg	g	Bad
10	vb	g	vg	vb	b	Bad	23	vb	vb	g	g	vg	Neutral
11	g	g	g	vb	vg	Good	24	g	vb	b	g	vb	Bad
12	n	vg	g	vb	g	Neutral	25	b	n	b	vg	b	Bad
13	g	g	n	n	vb	Bad							

Using this information, we then tried to construct a MR-Sort model with a simple majority rule (step 3 of the protocol).

```
# define the category assignments of the generated alternatives

assignments <-c("Bad", "Neutral", "Bad", "Bad", "Neutral",
                "Good", "Neutral", "Good", "Bad", "Bad",
                "Good", "Neutral", "Bad", "Bad", "Bad", "Bad",
                "Bad", "Neutral", "Bad", "Neutral", "Bad",
                "Bad", "Neutral", "Bad", "Bad")

names(assignments) <- rownames(pT)

m1 <- MRSortInferenceExact(pT, assignments, categoriesRanks,
                           criteriaMinMax, veto = FALSE,
                           readableWeights = TRUE,
```

```
                                readableProfiles = TRUE)
```

Running the R code up to this point gives us the following result contained in `m1`:

```
> m1

$solverStatus
[1] "Failed ( invalid bounds )"
```

This means that no MR-Sort model with only the majority rule is able to capture the assignments given by DM1 on the 25 profiles (step 4). In order to determine whether we need to use a more expressive model, we first generate the minimal sets of incompatible assignments (step 5).

```
incomp <- MRSortIdentifyIncompatibleAssignments(pT, assignments,
                                                categoriesRanks,
                                                criteriaMinMax,
                                                veto = FALSE)
```

The result, containing a list of sets of indexes corresponding to the profiles that would need to be assigned to different categories in order for us to use an MR-Sort model, is given by:

```
> incomp
$incompatibleSets
$incompatibleSets[[1]]
[1] "2"  "12"

$incompatibleSets[[2]]
[1] "2" "8"

$incompatibleSets[[3]]
[1] "2"  "11"

$solverStatus
[1] "Success"
```

The `solverStatus` output reassures us that we have found all incompatible sets of alternatives assignments. In order to also provide DM1 with the alternative category assignments for each incompatible assignment example, we removed one by one each set of team member profiles and generated a new MR-Sort model (see the `alternativesIDs` parameter below).

```
> m1pr <- MRSortInferenceExact(pT, assignments, categoriesRanks,
                               criteriaMinMax, veto = FALSE,
                               readableWeights = TRUE,
                               readableProfiles = TRUE,
                               alternativesIDs = rownames(pT)
                               [!rownames(pT) %in% c(2, 12)])
```

The result of this function is given by `m1pr`:

```
> m1pr
$lambda
[1] 0.6004
```

```
$weights
    c1     c2     c3     c4     c5
0.2008 0.1988 0.2008 0.1998 0.1998

$profilesPerformances
            c1      c2    c3     c4    c5
Good     0.000  0.0000 1.000  1.001 1.501
Neutral -0.999 -1.0005 0.001 -1.000 1.001
Bad         NA      NA    NA     NA    NA

$vetoPerformances
NULL

$solverStatus
[1] "Solution found"
```

`lambda` corresponds to the majority threshold of the MR-Sort model, the vector `weights` contains the criteria weights, while `profilesPerformances` contains a matrix with the evaluations of the category limits. The first line of this matrix corresponds to the limit between categories Good and Neutral, the second line corresponds to the limit between categories Neutral and Bad, while the last is simple a dummy profile corresponding to the lower limit of category Bad. No values are needed for this profile (`NA`), however it is nevertheless provided as other function of the package require it to be present. We also find that no veto profiles are provided and that the model corresponds to the optimal solution fulfilling all provided assignment examples.

Using the model from `m1pr`, we then applied the MR-Sort assignment procedure on the alternatives that we previously removed, in order to extract their category assignments which would allow us to continue using an MR-Sort model.

```
> MRSort(pT, m1pr$profilesPerformances, categoriesRanks,
         m1pr$weights, criteriaMinMax, m1pr$lambda,
         alternativesIDs = c(2, 12))

     2     12
 "Bad" "Good"
```

The above output indicates that alternative 2 should be assigned to the Bad category, while alternative 12 should be assigned to Good (DM1 initially assigned them both to Neutral).

We repeated this procedure for the remaining two sets of incompatible assignments ($\{2, 8\}$ and $\{2, 11\}$) and obtained the results illustrated in Table 4.

Using these sets, we devised a series of questions for DM1 to simplify his task of considering these alternative assignments (step 6). As the second team member profile appeared in all three sets, we asked if he would agree to change his assignment from Neutral to Bad. DM1 agreed to change his assignment, and confirmed that he had initially hesitated between these two categories.

We continued by asking if he would be willing to change the assignment for 8, 11, or 12, in order to complete one of the three sets of assignment profiles. DM1 did not agree with changing any of these other assignments. Because we were unable

Table 4 The alternative assignments for an MR-Sort model within the first iteration of the inference protocol for DM1.

	Profile	Criteria					Category	
	number	g_1	g_2	g_3	g_4	g_5	original	alternative
First set	2	b	vg	n	vb	n	Neutral	Bad
	8	n	n	g	b	g	Good	Neutral
Second set	2	b	vg	n	vb	n	Neutral	Bad
	11	g	g	g	vb	vg	Good	Neutral
Third set	2	b	vg	n	vb	n	Neutral	Bad
	12	n	vg	g	vb	g	Neutral	Good

to apply any of the alternative assignments in full, we did not apply the change to profile 2.

Consequently we increased the complexity of the model (step 7) and found that an MR-Sort model with vetoes was able to capture all 25 profile assignments (again steps 3 and 4). The code for constructing an MR-Sort model with vetoes is presented below:

```
> m1 <- MRSortInferenceExact(pT, assignments, categoriesRanks,
                             criteriaMinMax, veto = TRUE,
                             readableWeights = TRUE,
                             readableProfiles = TRUE)
```

Notice that we have used the same function `MRSortInferenceExact`, however, by making the `veto` parameter `TRUE`, the function will now try to construct an MR-Sort model with vetoes. The result of this step is presented below.

```
> m1
$lambda
[1] 0.5996

$weights
    c1     c2     c3     c4     c5
0.1992 0.2002 0.2012 0.1992 0.2002

$profilesPerformances
            c1     c2    c3       c4     c5
Good     0.001  0.000 1.001 -1.0000  0.667
Neutral -1.000 -0.999 0.001 -1.5005 -0.667
Bad         NA     NA    NA       NA     NA

$vetoPerformances
        c1 c2 c3 c4 c5
Good    -2 NA -1 NA -1
Neutral NA NA -2 NA -2
Bad     NA NA NA NA NA

$solverStatus
[1] "Solution found"
```

We now find that an MR-Sort model with vetoes fits the provided assignments examples, since the `MRSortInferenceExact` function found an optimal solution. We observe that the `vetoPerformances` parameter is now not empty, containing the veto profiles for each of the three categories. The veto profiles only contain values where minimum performances for a category are necessary. Naturally, the veto profile of the Bad category will always be empty, and is only provided for interfacing with other functions of the package.

This ended the first iteration of the protocol. In order to output an illustration of the model given by `m1`, we may use the following R code:

```
# defining the plot lower and upper bounds

criteriaLBs=c(-2,-2,-2,-2,-2)

names(criteriaLBs) <- criteria

criteriaUBs=c(2,2,2,2,2)

names(criteriaUBs) <- criteria

plotMRSortingProblem(pT, m2$profilesPerformances,
                     categoriesRanks, assignments,
                     criteriaMinMax, criteriaUBs,
                     criteriaLBs, NULL, m2$vetoPerformances,
                     'V', m2$weights, m2$lambda,
                     alternativesIDs = "no alternatives")
```

We use the `plotMRSortingProblem` function in order to draw the generated MR-Sort model. This function also requires that we specify the lower and upper bounds of the criteria evaluation scales, which we provide thought `criteriaLBs` and `criteriaUBs` variables. The function also allows to plot several alternatives along with their assignment, however, at this point we are only interested in seeing the model. The output of this code is given in Figure 4. The first diagram shows the limit between the Bad and Neutral categories, while the second shows the boundary between Neutral and Good. The lines correspond to the category limits, while the dark regions represent the range of values which would trigger a veto.

At this point, the model was not yet presented to DM1, as at least a couple of iterations of the protocol need to be followed. An additional set of 10 profiles were generated and presented to DM1 (step 9). The code below illustrates the generation procedure, however, the script that can be loaded from the MCDA package uses the same evaluations that were used in this experiment, instead of regenerating them, in order for us to be able to reproduce the original results.

```
# generate new alternatives and add them to the performance table

pT <- unique(rbind(pT,matrix(sample.int(5, 1000, T), 100, 5)
                                                    - 3))[1:35,]

rownames(pT) <- 1:35
```

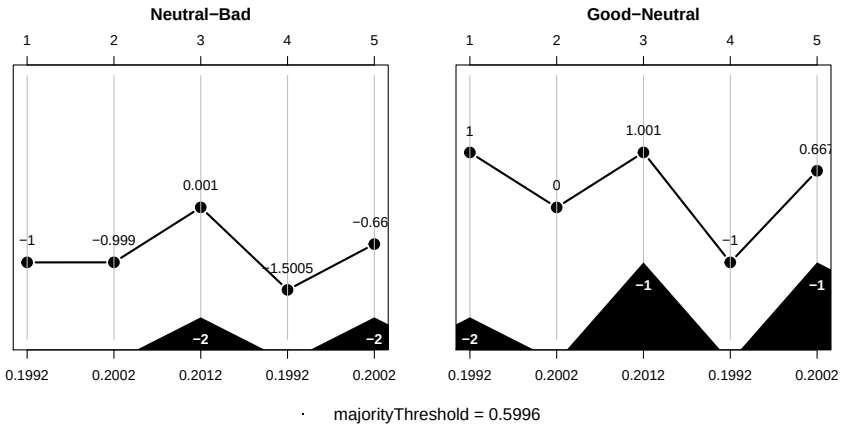

Fig. 4 First iteration preference model of DM1 (MR-Sort with vetoes).

DM1 assigned the new set of alternatives (step 2) as shown in Table 5. This was again done interactively with a researcher present.

Table 5 The second set of contributor profiles and their assignment by DM1.

Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Assigned category	Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Assigned category
26	vg	vg	n	b	b	Bad	31	vb	b	vg	vg	b	Bad
27	vg	b	n	vg	b	Bad	32	vb	b	n	vg	vg	Bad
28	vg	b	n	b	vg	Neutral	33	vg	vg	vb	vg	vg	Neutral
29	vb	vg	n	vg	b	Bad	34	vg	vg	vg	vg	vb	Good
30	vb	vg	n	b	vg	Bad	35	b	vb	g	vb	n	Neutral

Adding the new assignments in R is done as follows:

```
# define the category assignments of the generated alternatives

assignments <-c(assignments, "Bad", "Bad", "Neutral", "Bad",
                "Bad", "Bad", "Bad", "Neutral", "Good", "Neutral")

names(assignments) <- rownames(pT)
```

Using these new alternatives and their assignments, we again tested whether an MR-Sort model with vetoes was able to capture them (step 3):

```
> m2 <- MRSortInferenceExact(pT, assignments, categoriesRanks,
                             criteriaMinMax, veto = TRUE,
                             readableWeights = TRUE,
                             readableProfiles = TRUE)
```

```
> m2
$solverStatus
[1] "Failed ( invalid bounds )"
```

As the code above illustrates, this model did not completely fit the assignments. The `MRSortInferenceExact` function reported that the mixed-integer linear solver did not find a feasible solution. We therefore proceeded to check if DM1 had any hesitations in his assignments which would allow this model to be used. We again identified the sets of alternatives with incompatible assignments (step 5):

```
incomp <- MRSortIdentifyIncompatibleAssignments(pT, assignments,
                                                categoriesRanks,
                                                criteriaMinMax,
                                                veto = TRUE)
> incomp
$incompatibleSets
$incompatibleSets[[1]]
[1] "2"  "16"

$incompatibleSets[[2]]
[1] "2"  "34"

$incompatibleSets[[3]]
[1] "2"  "35"

$incompatibleSets[[4]]
[1] "2"  "33"

$incompatibleSets[[5]]
[1] "30" "33"

$solverStatus
[1] "Success"
```

Five sets of alternatives were found (`incompatibleSets`). We use the same approach of inferring an MR-Sort model using all assignment examples except those from each incompatible set and then using this model in order to find their new assignments for each of the five incompatible sets. We illustrate the results in Table 6.

We observed that the first four sets contained the second team member profile, which DM1 had already agreed to change. Therefore, we continued by first asking if he would also agree to an alternative assignment for one of the remaining profiles in these sets (step 6). DM1 did not accept changing the assignment of profiles 16, 33 or 34, especially for the third set where the alternative assignment strongly contradicted the initial assignment. However, DM1 agreed to change the assignment of profile 35 from Neutral to Bad. We make these changes using the following code:

```
assignments[2] <- "Bad"

assignments[35] <- "Bad"
```

With these changes made, we now construct an MR-Sort model with vetoes:

Table 6 The alternative assignments for an MR-Sort model with vetoes within the second iteration of the inference protocol for DM1.

	Profile	Criteria					Category	
	number	g_1	g_2	g_3	g_4	g_5	original	alternative
First set	2	b	vg	n	vb	n	Neutral	Bad
	16	b	vg	g	vg	vb	Bad	Neutral
Second set	2	b	vg	n	vb	n	Neutral	Bad
	33	vg	vg	vb	vg	vg	Neutral	Bad
Third set	2	b	vg	n	vb	n	Neutral	Bad
	34	vg	vg	vg	vg	vb	Good	Bad
Fourth set	2	b	vg	n	vb	n	Neutral	Bad
	35	b	vb	g	vb	n	Neutral	Bad
Fifth set	30	vb	vg	n	b	vg	Bad	Neutral
	33	vg	vg	vb	vg	vg	Bad	Bad

```
m2 <- MRSortInferenceExact(pT, assignments, categoriesRanks,
                           criteriaMinMax, veto = TRUE,
                           readableWeights = TRUE,
                           readableProfiles = TRUE)
```

The resulting model given by `m2` is:

```
> m2
$lambda
[1] 0.6006

$weights
    c1     c2     c3     c4     c5
0.2002 0.1992 0.2002 0.2002 0.2002

$profilesPerformances
           c1     c2    c3 c4    c5
Good    0.000  0.667 1.000 -1 1.501
Neutral -0.999 -0.667 0.001 -2 1.001
Bad         NA     NA    NA NA    NA

$vetoPerformances
        c1 c2 c3 c4 c5
Good    -2 -1 -1 NA NA
Neutral NA NA NA NA NA
Bad     NA NA NA NA NA

$solverStatus
[1] "Solution found"
```

We depict this model using the `plotMRSortingProblem` function in Figure 5.

With the second iteration of preference modeling completed, DM1 agreed to continue with another iteration (step 8). Adding more assignment examples allows us to more accurately infer the model of DM1's preferences. We have not used in

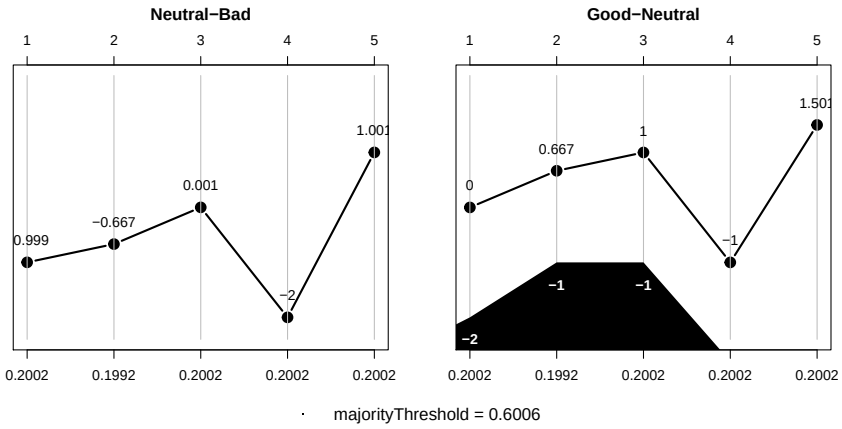

Fig. 5 Second preference model of DM1 (MR-Sort with vetoes).

these experiments any measure of convergence of the model, but only the opinion of DM1 with respect to its output, in the form of assignment rules, as we will see shortly.

Starting a new iteration, we generated an additional 10 profiles (step 9), and asked DM1 to assign them during an interview (step 2). The results of this assignment are presented in Table 7.

Table 7 The third set of contributor profiles and their assignment by DM1.

Profile	Criteria					
number	g_1	g_2	g_3	g_4	g_5	Category
36	b	vg	n	vg	g	Bad
37	n	b	g	vb	vg	Neutral
38	n	vb	vb	b	vg	Bad
39	vb	vb	g	b	vg	Neutral
40	vg	vb	vg	vg	vg	Good

Profile	Criteria					
number	g_1	g_2	g_3	g_4	g_5	Category
41	n	vb	g	vb	vb	Bad
42	n	b	g	b	vb	Bad
43	b	vg	n	vb	vg	Good
44	b	vg	vb	g	b	Bad
45	vb	g	vb	vb	vg	Bad

After adding the new profiles and their assignments to the existing ones, we found that an MR-Sort model with vetoes was not able to represent all of the assignments (step 3 and 4). We generated four possible profile changes, as shown in Table 8.

```
incomp <- MRSortIdentifyIncompatibleAssignments(pT, assignments,
                                                categoriesRanks,
                                                criteriaMinMax,
                                                veto = TRUE)
> incomp
$incompatibleSets
$incompatibleSets[[1]]
```

```
[1] "8"  "34"

$incompatibleSets[[2]]
[1] "14" "43"

$incompatibleSets[[3]]
[1] "33" "43"

$incompatibleSets[[4]]
[1] "34" "43"

$solverStatus
[1] "Success"
```

Table 8 The alternative assignments for an MR-Sort model with vetoes within the third iteration of the inference protocol for DM1.

	Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Category original	alternative
First set	8	n	n	g	b	g	Good	Neutral
	34	vg	vg	vg	vg	vb	Good	Bad
Second set	14	vg	n	vb	b	g	Bad	Neutral
	43	b	vg	n	vb	vg	Good	Bad
Third set	33	vg	vg	vb	vg	vg	Neutral	Bad
	43	b	vg	n	vb	vg	Good	Bad
Fourth set	34	vg	vg	vg	vg	vb	Good	Bad
	43	b	vg	n	vb	vg	Good	Bad

Profiles 8, 33 and 34 reoccurred, but we were already aware that DM1 did not want to change their assignments. Therefore we only inquired on the possibility of changing profiles 14 and 43. DM1 felt strongly about retaining his initial assessment, motivating us to test a more complex model (step 6).

We tested all other MR-Sort variants and found that only the one with conflicting vetoes and dictators was able to fulfill DM1's assignments.

```
m3 <- LPDMRSortInferenceExact(pT, assignments, categoriesRanks,
                              criteriaMinMax, majorityRule = "dv",
                              readableWeights = TRUE,
                              readableProfiles = TRUE,
                              minmaxLPD = TRUE)
> m3
$lambda
[1] 0.601

$weights
   c1    c2    c3    c4    c5
0.200 0.199 0.201 0.200 0.200
```

```
$profilesPerformances
            c1     c2    c3 c4    c5
Good     0.000 -0.001 1.000 -1 2.001
Neutral -0.999 -1.001 0.001 -2 1.001
Bad         NA     NA    NA NA    NA

$vetoPerformances
        c1 c2 c3 c4 c5
Good    -2 NA -1 NA  1
Neutral -2 NA -2 NA  1
Bad     NA NA NA NA NA

$dictatorPerformances
        c1 c2 c3 c4 c5
Good    NA  0 NA NA NA
Neutral NA  0  2 NA  2
Bad     NA NA NA NA NA

$solverStatus
[1] "Solution found"
```

This model is illustrated in Figure 6. Vetoes are shown in black, while dictators are lightly shaded.

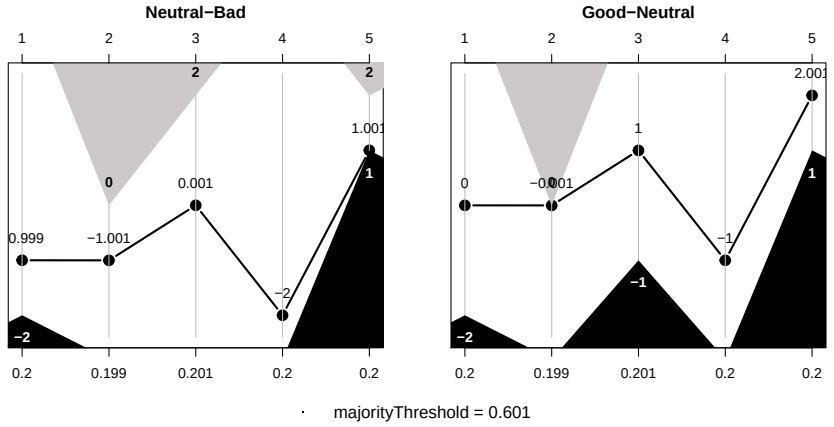

Fig. 6 Third preference model of DM1 (MR-Sort with conflicting vetoes and dictators).

At this point, DM1 expressed his interest in reviewing the model we had generated. In order to allow DM1 to validate the model, we presented a series of rules derived from it, depicted in Figure 7. The rules were displayed as a series of graphs showing all combinations of evaluations that a good or bad team member could have. The rules for the neutral category were not included, as they can be inferred as a complement of the rules for the good and bad team members. At the time of this writing, the

functionalities of generating and plotting the assignment rules are currently not implemented in the MCDA package. In order to minimize the number of rules per category, we allowed for some overlap to occur. It may be noticed, for instance, that the ideal team member profile (containing the highest evaluations on all criteria) is included in all rules corresponding to the Good contributors.

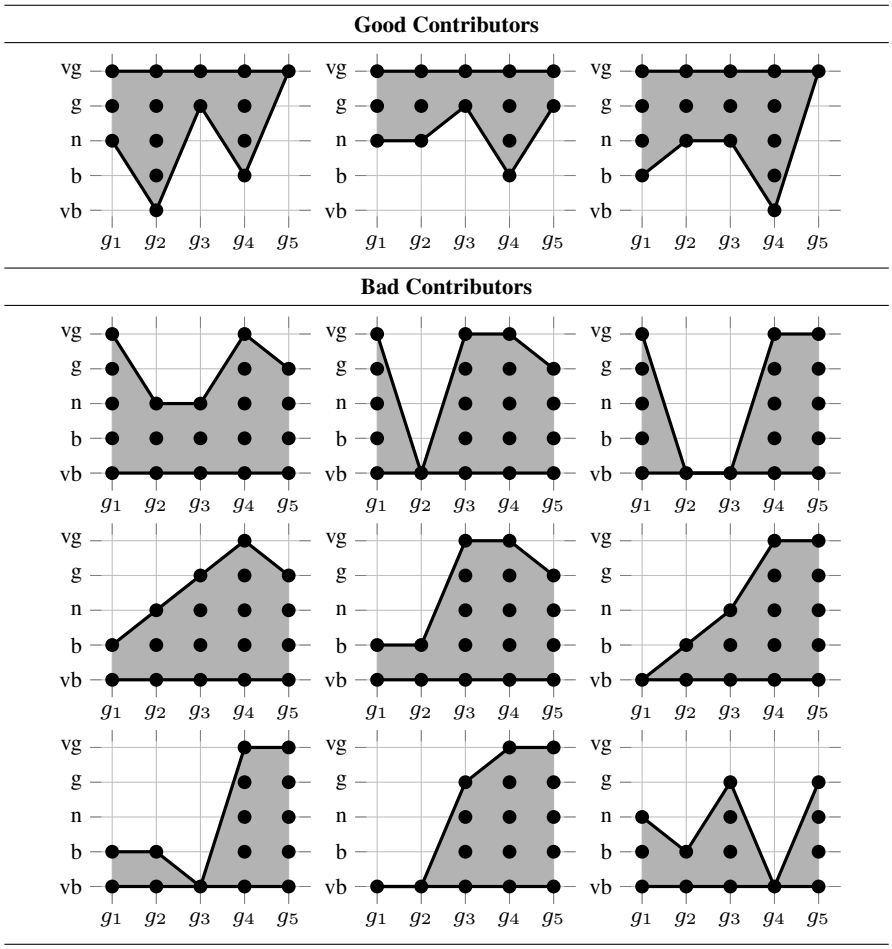

Fig. 7 Assignment rules for good and bad contributors from the perspective of DM1.

We explained the interpretation of the rules verbally to DM1, who validated the illustrated rules without making any changes to them. For example, the first rule tells us that a Good contributor is at least very good in documentation and testing, good in coding, neutral in commitment, bad in his/her global vision on the project and very bad in working with others. The second rule tells us that a Good contributor can alternatively be at least good in coding, documentation and testing,

neutral in commitment and in working with others and bad in his/her global vision on the project. Finally, a Good contributor may also be someone at least very good in documentation and testing, neutral in working with others and in coding, bad in commitment and very bad in his/her global vision on the project. Similar descriptions of the rules describing a Bad contributor can be made, except that in that case the boundaries correspond to "at most" assertions, instead of "at least".

5.2 Inferring the preferences of DM2

We continue with the second DM, who selected the same 5 criteria as the first one. Below, we will only illustrate the different steps of the protocol and their results, without including this time the R code.

Because DM2 selected the same subset of dimensions as DM1, we asked him to assign the same initial 25 model alternatives as DM1 in order to better illustrate any similarities or differences in assignment choices. DM2's assignments were quite different from those of DM1, and are shown in Table 9.

Table 9 The initial set of contributor profiles and their assignment by DM2.

Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Assigned category	Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Assigned category
1	vg	g	vb	vg	n	Good	14	vg	n	vb	b	g	Good
2	b	vg	n	vb	n	Bad	15	n	b	b	vb	n	Bad
3	b	b	b	b	g	Bad	16	b	vg	g	vg	vb	Neutral
4	b	b	vb	vg	n	Bad	17	n	b	n	g	n	Neutral
5	g	vb	vg	b	b	Neutral	18	vg	vb	vg	g	b	Neutral
6	vg	g	vg	n	vg	Good	19	vg	vb	n	n	vb	Neutral
7	g	n	b	n	vg	Good	20	vb	vg	vg	b	vg	Bad
8	n	n	g	b	g	Neutral	21	vb	n	vb	n	vb	Bad
9	n	vg	n	g	b	Good	22	vb	b	vb	vg	g	Bad
10	vb	g	vg	vb	b	Bad	23	vb	vb	g	g	vg	Bad
11	g	g	g	vb	vg	Good	24	g	vb	b	g	vb	Neutral
12	n	g	g	vb	g	Good	25	b	n	b	vg	b	Bad
13	g	g	n	n	vb	Good							

We tried to fit the data with an MR-Sort model and found that at most 24 of the 25 assignment examples could be captured with this type of model. Therefore we provided DM2 with the possible sets of alternatives, shown in Table 10, which would allow for an MR-Sort model to be used should the assignments be changed.

DM2 had already verbally expressed hesitation in assigning the eighth profile, and therefore agreed to change his assignment. The resulting MR-Sort model is presented in Figure 8.

Table 10 The alternative assignments for an MR-Sort model within the first iteration of the inference protocol with DM2.

	Profile	Criteria					Category	
	number	g_1	g_2	g_3	g_4	g_5	original	alternative
First set	8	n	n	g	b	g	Neutral	Good
Second set	14	vg	n	vb	b	g	Good	Neutral

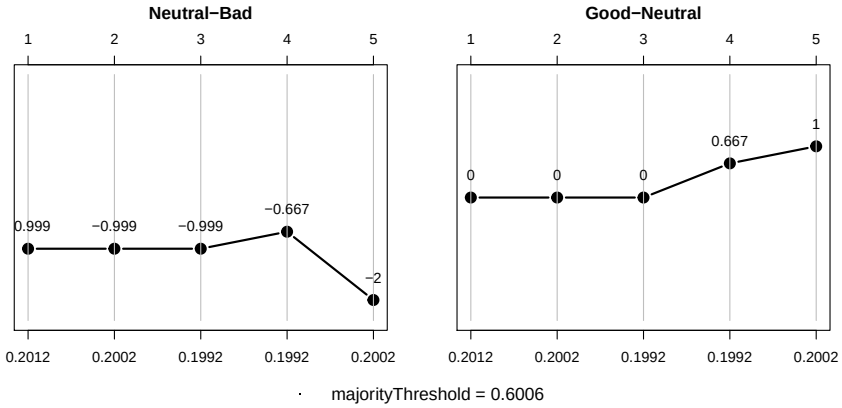

Fig. 8 First preference model for DM2 (MR-Sort using a simple majority rule).

We continued with a second iteration by generating an additional 10 profiles. This new set was presented to DM2, who assigned them as shown in Table 11.

Table 11 The second set of contributor profiles and their assignment by DM2.

Profile	Criteria					Assigned	Profile	Criteria					Assigned
number	g_1	g_2	g_3	g_4	g_5	category	number	g_1	g_2	g_3	g_4	g_5	category
26	b	b	b	vg	vg	Bad	31	n	n	g	vb	vb	Neutral
27	vb	n	g	n	g	Bad	32	n	vb	n	vb	vb	Bad
28	vg	b	n	b	n	Neutral	33	vb	n	n	n	vb	Bad
29	n	n	vb	vb	g	Neutral	34	n	n	vb	n	vb	Bad
30	b	b	vg	b	vg	Bad	35	n	vb	vb	n	g	Neutral

We combined the initial 25 profiles with the 10 new profiles and checked their fit with an MR-Sort model. As only 33 out of the 35 profiles could be represented by this model, we proceeded to determine if DM2 would accept adapting a minimal number of his assignments. Only one set of profile changes would make this possible. This option is shown in Table 12.

Table 12 The alternative assignments for an MR-Sort model within the second iteration of the inference protocol (DM2).

	Profile	Criteria					Category	
	number	g_1	g_2	g_3	g_4	g_5	original	alternative
First set	16	b	vg	g	vg	vb	Neutral	Bad
	34	n	n	vb	n	vb	Bad	Good

DM2 expressed a willingness to change the assignment of the first profile from this set. However, the change in assignment for the second profile was too drastic. Therefore we looked at a more complex model, in particular an MR-Sort with vetoes or an MR-Sort with dictators, could describe the assignments. Both of these models were also unable fit the data completely, but the model with dictators required two profiles to be changed, whereas the model with vetoes only needed one profile change. Therefore we focused on the MR-Sort with vetoes. The possible profile changes are found in Table 13.

Table 13 The alternative assignments for an MR-Sort model with vetoes within the second iteration of the inference protocol (DM2).

	Profile	Criteria					Category	
	number	g_1	g_2	g_3	g_4	g_5	original	alternative
First set	16	b	vg	g	vg	vb	Neutral	Bad
Second set	32	n	vb	n	vb	vb	Bad	Neutral
Third set	34	n	n	vb	n	vb	Bad	Neutral

As DM2 had previously expressed a hesitation on the assignment of profile 16, he quickly agreed to the proposed changed. The model that reflects all assignments at the end of the second iteration is depicted in Figure 9.

The model appeared to further illustrate the existence of vetoes on the first two criteria, which we had already posited from the reasoning DM2 expressed in the assignment interview. DM2 agreed to continue with another iteration of the protocol.

We generated an additional set of 10 profiles, which were assigned as illustrated in Table 14.

After adding the 10 new profiles and their assignments to the existing ones, we found that an MR-Sort model with vetoes was not able to represent all of these assignments and that at least three profile assignments needed to be altered to allow for this type of model to be used. These sets are presented in Table 15.

As profile 39 appeared in all three sets, we began with asking DM2 if he hesitated in assigning this profile. As the answer was negative, we considered a more complex model. An MR-Sort model with veto weakened by a dictator was able to fully reflect all of the DM2's assignments. Figure 10 presents this model.

At this point, DM2 expressed an interest in reviewing the model, which, after three iterations, already reflected a large portion of his perspective. A set of rules were

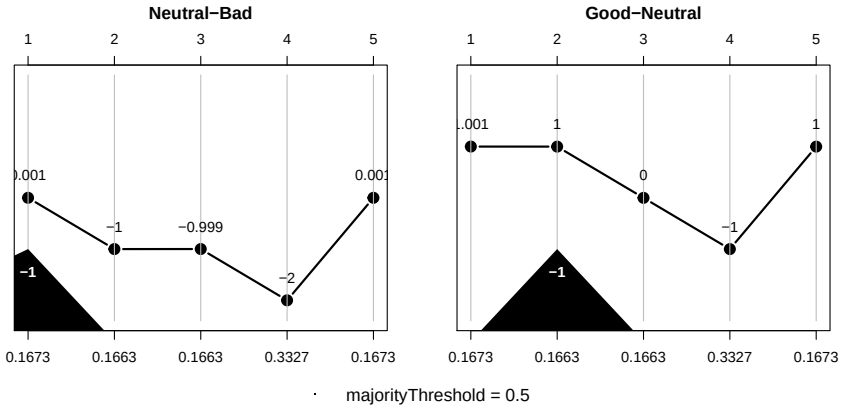

Fig. 9 Second preference model for DM2 (MR-Sort with vetoes).

Table 14 The third set of contributor profiles and their assignment by DM2.

Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Assigned category	Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Assigned category
36	n	vb	vb	vb	g	Neutral	41	g	g	n	vb	vb	Good
37	n	g	n	vb	g	Neutral	42	n	vb	b	vg	vg	Neutral
38	vg	vb	vg	vg	vg	Good	43	g	b	vb	vb	vb	Neutral
39	b	vg	vg	vg	vg	Neutral	44	n	b	n	vb	vb	Neutral
40	n	n	b	vg	vg	Neutral	45	vg	vb	b	vg	n	Good

Table 15 The alternative assignments for an MR-Sort model with vetoes within the third iteration of the inference protocol (DM2).

	Profile number	Criteria g_1	g_2	g_3	g_4	g_5	Category original	alternative
	9	n	vg	n	g	b	Bad	Neutral
First set	32	n	vb	n	vb	vb	Bad	Neutral
	39	b	vg	vg	vg	vg	Neutral	Bad
	9	n	vg	n	g	b	Bad	Neutral
Second set	12	n	vg	g	vb	g	Good	Neutral
	39	b	vg	vg	vg	vg	Neutral	Bad
	38	vg	vb	vg	vg	vg	Good	Neutral
Third set	39	b	vg	vg	vg	vg	Neutral	Bad
	45	vg	vb	b	vg	n	Good	Neutral

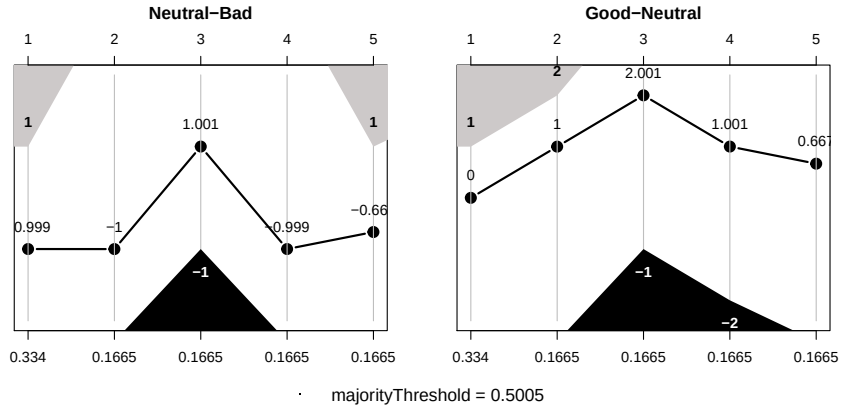

Fig. 10 Third preference model for DM2 (MR-Sort with vetoes weakened by dictators).

generated from the model, as shown in Figure 11, and we explained the implications in an interview with DM2.

DM2 felt that the model was slightly inaccurate, and decided to tweak the first two rules of the Good category by raising the boundary of the second criterion from $very\ bad$ to bad, as he felt that being neutrally committed to the project and having a clear inability to work with others would not be a characteristic of a good team member, regardless of all other factors. In the Bad category, DM2 also felt that a very committed team member should not be in the Bad category, regardless of other poor evaluations, however if the commitment were to fall to neutral, this would indeed be a Bad team member. We therefore lowered the evaluation on the first criterion of the first Bad rule from $very\ good$ to $neutral$. The second and third rules in the Bad category were also adjusted by lowering slightly the good evaluation on the second criterion to a neutral one. The adjusted assignment rules, which were still consistent with the profiles assignments of DM2, are presented in Figure 12.

The rules can again be interpreted as lower limits for a contributor to be considered good, a single rule among the five being needed by a contributor to validate, and as upper limits for a contributor to be considered bad respectively.

In the next section we will discuss the findings from these experiments, and what it implies to practitioners and scholars.

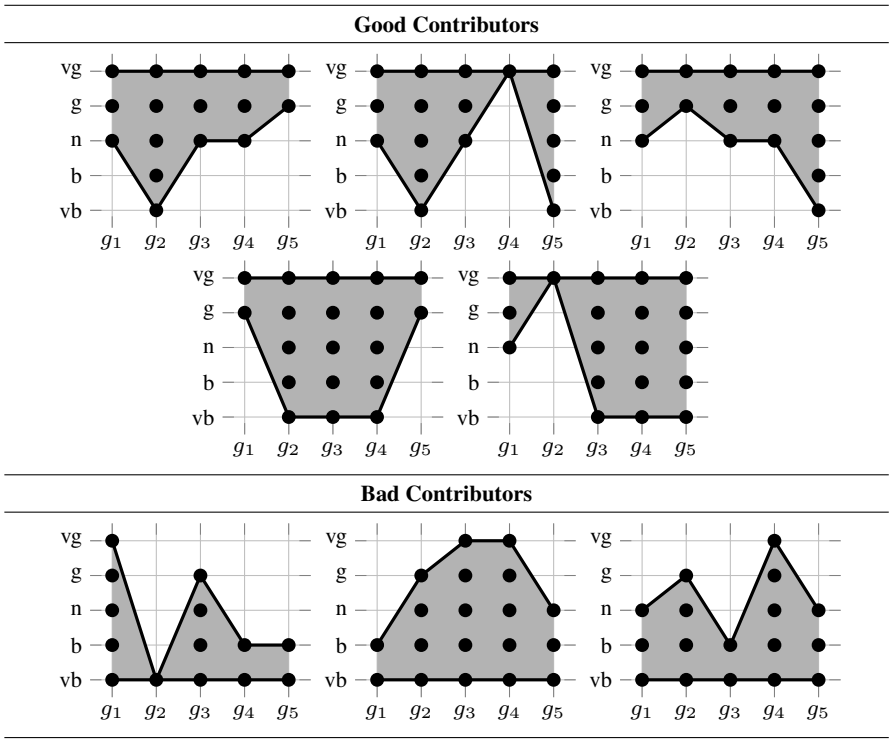

Fig. 11 Assignment rules for good and bad contributors for DM2.

6 Discussion

6.1 Using the MCDA R package as a support tool

In this chapter, we proposed to support the multi-criteria evaluation process throughout all of its steps by use of a single environment, the R statistical software. This prevents analysts and DMs from having to resort to using multiple tools at different stages of the decision aiding process, adding an additional level of difficulty. The choice of using R throughout the process was motivated by its focus towards data analysis, its open source and package-based philosophy, as well as its large community of users and contributors.

While the MCDA package currently contains only a few MCDA algorithms, its authors wish to continue developing it in the future so that as many of the MCDA algorithms can be found within it. Furthermore, functions linked to the presentation of the results, for instance graphically, will also be added to complement the existing ones.

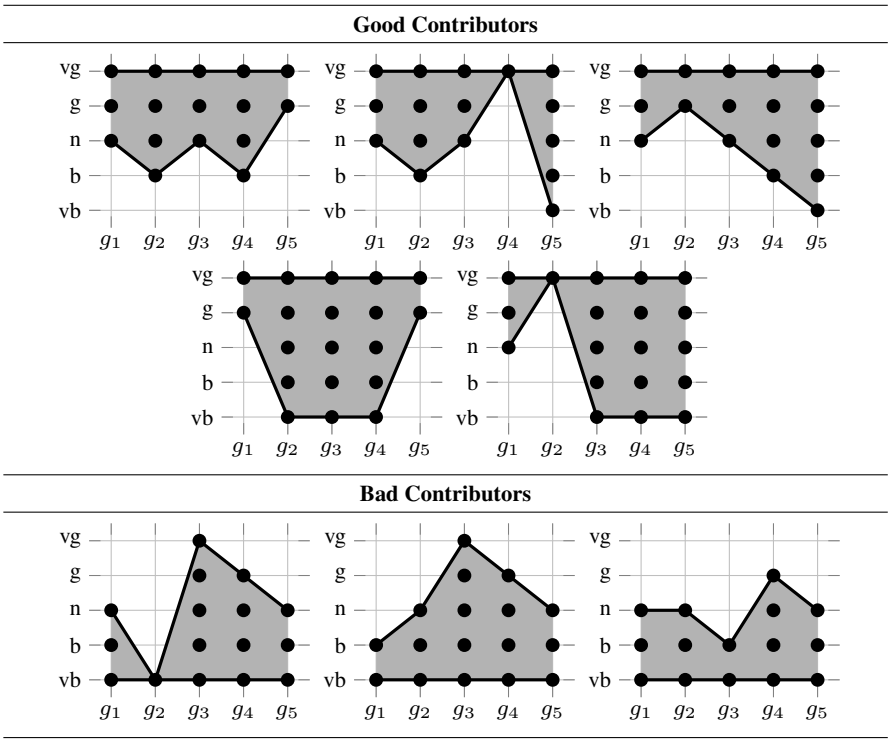

Fig. 12 Adjusted assignment rules for good and bad contributors (DM2).

This application of MCDA in a real-world case has also shown that, even when the decision aiding process is complex, by using R and the MCDA package we are able to successfully obtain a convincing solution.

6.2 Managerial and theoretical implications

In proposing this methodology, we contributed towards filling the gap between operational research techniques and information management sciences regarding the expression of different points of view, and the automation of the process, here in the manager/team relationship. We are confident that our method can be applied to different situations exhibiting teaming, and are key to the building of an efficient staffing support system, especially in the case of team building.

It proposes a way to go beyond the paradox raised by Kayworth and Leidner [32] about both the non-statistical link between team manager characteristics and team performance, but also team leader importance in trust-building by exposing that there is no direct link because managers are different and have different expectations. From a theoretical point of view, it opens a new area of research into the collection

of data on a manager's expectations regarding team members in particular, and principal-agent modeling in general. Instead of trying to identify a good agent for the majority of managers through statistical inference, we propose to construct an individual preference model for every manager.

From a practical point of view, the importance of trust-building and the role of the manager in doing so [32], through integrity and benevolence [30], has been identified. Therefore the first contact between the manager and the team, and the manager's communication of expectations are of utmost importance. This is where our work may provide solutions to practitioners. We have proposed a method and showed that this allows managers to better define what their evaluation criteria are. This model does not identify a single acceptable profile, but allows for team members to have different strengths and weaknesses, yet understand the relative importance of different attributes. Because the results are presented in a structured way, ambiguity is reduced and transparency is increased.

We are confident that this approach would result in solid and actionable results regarding efficient teaming questions, not only in the team building phase, but also in subsequent coaching and in explaining the team manager's expectations.

6.3 Application of assignment rules

The assignment rules provided as the output of our preference elicitation protocol can firstly serve the manager in better understanding his/her perspective on the important characteristics that make for a good team member, and similarly, those which make for a bad one. The use of these rules is also linked to more consistent evaluation of the team members, since the manager will refer to the generated assignment rules for all subsequent evaluations of new team members or re-evaluation of existing ones. While the evaluations of a given team member on every criterion are currently subjectively provided by the manager, using the assignment rules provides a level of objectivity. Additionally, the criteria evaluations could potentially be linked to less subjective means of evaluating team members, by linking them to systems that automatically asses their performance, or by having other members from the managerial structure intervene in constructing them.

A second benefit of the assignment rules is the possibility to provide recommendations to team members on which criteria they need to improve and on which they should be careful not to worsen. Let us consider the illustrative example presented in Fig. 13. We use, in this case, the assignment rules extracted for DM2, as well as a team member who was evaluated by DM2 as *neutral* in commitment (first criterion), *bad* in working with other and produced code quality (the following two criteria), *very bad* in his or her vision on the whole project (fourth criterion), but *very good* in documentation and testing (last criterion).

We notice that this contributor is evaluated by DM2 as neutral, since none of the assignment rules from the Good category are validated, nor any from the Bad category. The plots, however, give insights into how the contributor may improve in

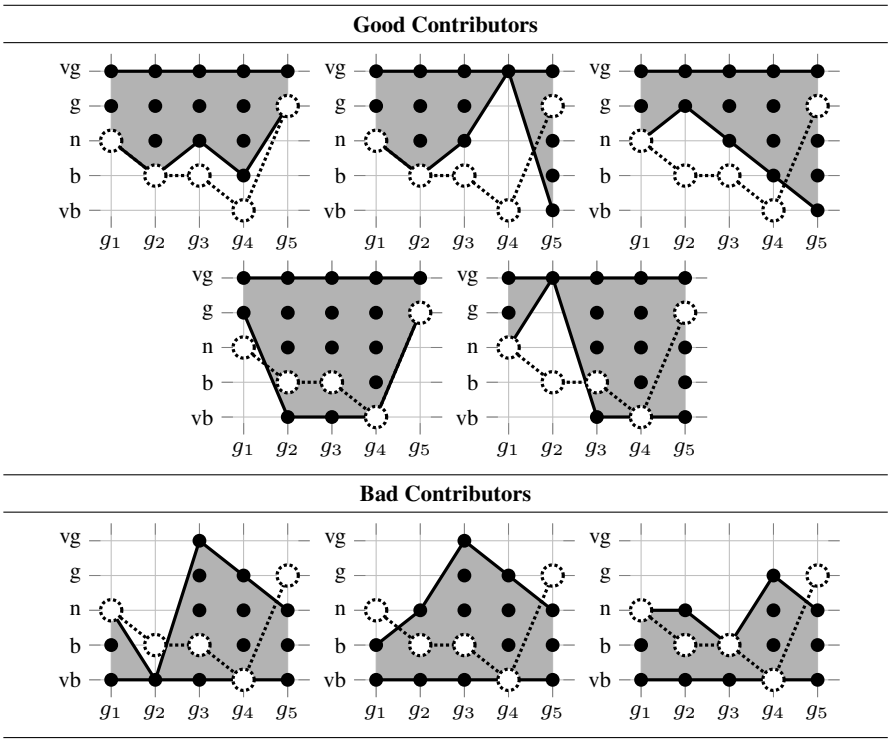

Fig. 13 Assignment rules of DM2 overlaid with contributor profile.

order to become a good contributor. With respect to the first assignment rule, he/she could improve slightly in produced code quality (g_3) in order to become good on this criterion and in the overall vision of the project (g_4) in order to not be *very bad*. The second assignment rule requires more drastic changes on the fourth criterion and will not likely be taken into account by DM2. The third rule requires the same changes as the first, but also an improvement from *bad* to *good* in teamwork skills. The fourth rule tells us that slight improvements in commitment (g_1) from *neutral* to *good* and in documentation and testing from *good* to *very good* may also make this contributor a good one. Finally, only improving in teamwork, however quite radically (from *bad* to *very good*) could also be proposed.

While DM2 would naturally ask all contributors to strive to improve on all criteria, ideally wishing all contributors to be *very good* on all of them, the previous recommendations may serve as the paths of least resistance, allowing contributors to become globally good quickly and without too much effort, As the recommendations with respect to the second and third assignment rules include those for the first rule, DM2 would consider recommending only the latter as well as those from rules 4 and 5. While we may be inclined to select only a subset of recommendations pertaining to the minimum number of characteristics to improve, or the least amounts, it may

be ultimately up to the contributor to decide on what to improve, as he/she may have a different perspective on the difficulty of improving on every criterion.

A similar analysis may be made with respect to the three rules corresponding to the Bad category, assessing in this case the risk for a contributor to become bad. The first rule tells us that a decrease from *bad* to *very bad* in teamwork and a decrease from *good* to *neutral* in documentation and testing would make this contributor a bad one. The same decrease on the last criterion paired with a decrease from *neutral* to *bad* on the first one also correlates with this contributor becoming bad. Finally, the third rule only requires the same performance decrease in documentation and testing, and since it is included in the other recommendations, we can safely state that the highest risk of becoming bad is associated with a performance drop on this criterion. DM2 may in this case suggest to the contributor to be careful to not decrease their efforts on this aspect.

7 Conclusion

This chapter's goal was to advocate for a manager-specific approach to evaluating the composition of a good team member, especially in the case of FLOSS inspired organizations and to illustrate a methodology to make this assessment explicit with existing tools, available in a standard platform such as R. We do not pretend to have provided any categorization of types of managers on the basis of their preferences. However, our experiment clearly shows that while a set of variables relevant to the evaluation of team members can be identified within a domain, different managers rely on different subsets of factors, and weigh the factors differently.

Furthermore, we demonstrated that our 3-step methodology can be used to create a model which was accepted by the FLOSS managers we included in our experiment. The three steps consist in defining the variables according to the literature and to validate them with practitioners, selecting an appropriate subset of criteria for each individual manager, and finally eliciting a model of the manager's contributors evaluation through an interactive and iterative process using fictitious contributors profiles.

The presented methodology and the illustrated implementation within R may serve as a blueprint for readers interested in applying it to a different context, where an ordered evaluation of entities (objects, people, strategies) is required, and where this evaluation needs to be done in accordance to the preferences of one or multiple decision-makers.

We wish to mention, however, several limitations of our approach, which open several perspectives for improvement. One such limitation is the number of interactions with the DM, which were substantial and may be streamlined by making the interactions automatic through a dedicated system. One direct factor for having multiple interactions with the DM was our use of exact inference algorithms, which begin to require large amounts of time and computational resources even for relatively modest numbers of contributor profiles. Because of this, we also had to deal

with the effects of bounded rationality, where the DM's judgments could become slightly inconsistent, changing perspective on the problem from one session to the next. In order to overcome this by accelerating the whole process, approximative inference methods could be explored. This would allow us to perform more iterations of the protocol and therefore arrive to models that do not vary consistently between the final iterations.

The next steps regarding the use of this methodology for the analysis of teams would be to do the same at group level, and then to study the connexions between team member qualities and group performance, contributing to the discussion opened by [36] on that matter. We could additionally explore how business processes and existing information systems within companies can be used to automatically evaluate the performance of a contributor on the technical criteria as well as how procedures used in recruitment can be used to evaluate them on the psychological criteria, leading to the perspective of integrating our methodology directly into such systems.

Finally, we only explored the family of models based on outranking relations, MR-Sort and its extensions. The motivation behind the use of these models was based on the ordinal scales used to evaluate team members on the different criteria, and the fact that the DM often describes a good team member based on the member's outstanding characteristics rather than the average of all attributes. We also assumed that the DM has accurate knowledge of the performance of each team member in all dimensions, although quantitative measures linked to each dimensions could be additionally considered, allowing the DM to judge unknown team members, for instance, in the context of a recruitment procedure.

In that sense it would provide a new piece to a recruitment system workflow, after a first selection of the candidate according to their technical skills and their social/trust links as in Malinowski et al. [39], whose data would be proposed as input for the manager in that context.

Acknowledgements This work was supported, in part, by Science Foundation Ireland grants 10/CE/I1855 and 13/RC/2094 to Lero - the Irish Software Research Centre (`www.lero.ie`).

We would like to thank the six FLOSS community managers who participated in this research and Pr. Mathieu Simonnet, who provided input on the modeling of psychological traits.

References

[1] Introduction to R. `https://www.datacamp.com/courses/free-introduction-to-r`. Accessed: 2018-03-15.

[2] Mark S Allen, Iain Greenlees, and Marc Jones. Personality in sport: A comprehensive review. *International Review of Sport and Exercise Psychology*, 6(1):184–208, 2013.

[3] Ann Barcomb, Michael Grottke, Jan-Philipp Stauffert, Dirk Riehle, and Sabrina Jahn. How developers acquire FLOSS skills. In *Proceedings of the 11th In-*

ternational Conference on Open Source Systems (OSS 2015). Springer Verlag, 2015.

[4] Genevieve Bassellier and Izak Benbasat. Business competence of information technology professionals: Conceptual development and influence on IT-business partnerships. *MIS quarterly*, pages 673–694, 2004.

[5] Sarah Beecham. Motivating software engineers working in virtual teams across the globe. In *Software Project Management in a Changing World*, pages 247–273. Springer, 2014.

[6] Sarah Beecham and John Noll. What motivates software engineers working in global software development? In Pekka Abrahamsson, Luis Corral, Markku Oivo, and Barbara Russo, editors, *International Conference on Product-Focused Software Process Improvement*, pages 193–209. Springer, 2015.

[7] Sébastien Bigaret, Richard Hodgett, Patrick Meyer, Tatyana Mironova, and Alexandru-Liviu Olteanu. Supporting the multi-criteria decision aiding process: R and the MCDA package. *EURO journal on decision processes*, 5(1–4):169 – 194, November 2017.

[8] D. Bouyssou and T. Marchant. An axiomatic approach to noncompensatory sorting methods in MCDM, I: the case of two categories. *European Journal of Operational Research*, 178(1):217–245, April 2007.

[9] D. Bouyssou and T. Marchant. An axiomatic approach to noncompensatory sorting methods in MCDM, II: more than two categories. *European Journal of Operational Research*, 178(1):246–276, April 2007.

[10] D. Bouyssou, T. Marchant, M. Pirlot, A. Tsoukiàs, and P. Vincke. *Evaluation and decision models with multiple criteria: Stepping stones for the analyst*. International Series in Operations Research and Management Science, Volume 86. Boston, 1st edition, 2006.

[11] Maximilian Capraro and Dirk Riehle. Inner source definition, benefits, and challenges. *ACM Computing Surveys (CSUR)*, 49(4):67, 2017.

[12] Kevin Daniel André Carillo and Josianne Marsan. "The dose makes the poison"—exploring the toxicity phenomenon in online communities. In *ICIS*, 2016.

[13] Emilio J Castilla and Stephen Benard. The paradox of meritocracy in organizations. *Administrative Science Quarterly*, 55(4):543–676, 2010.

[14] Shi-Jie Chen and Li Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Transactions on Engineering Management*, 51(2):111–124, 2004.

[15] Paul David, Rishab A. Ghosh, Rüdiger Glott, Jesús M. González-Barahona, Federico Heinz, and Joseph Shapiro. Free/Libre and Open Source Software: Worldwide Impact Study. FLOSS World D31: Track 1 International Report, Jun 2007.

[16] Sadhana Deshpande and Ita Richardson. Management at the outsourcing destination-global software development in India. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, pages 217–225. IEEE, 2009.

[17] L.C. Dias and J.N. Clímaco. On computing ELECTRE's credibility indices under partial information. *Journal of Multi-Criteria Decision Analysis*, 8(2):74–92, 1999.

[18] L.C. Dias and J.N. Clímaco. ELECTRE TRI for groups with imprecise information on parameter values. *Group Decision and Negotiation*, 9(5):355–377, September 2000.

[19] L.C. Dias, V. Mousseau, J. Figueira, and J.N. Clímaco. An aggregation/disaggregation approach to obtain robust conclusions with ELECTRE TRI. *European Journal of Operational Research*, 138(2):332–348, April 2002.

[20] James E Driskell, Gerald F Goodwin, Eduardo Salas, and Patrick Gavan O'Shea. What makes a good team player? Personality and team effectiveness. *Group Dynamics: Theory, Research, and Practice*, 10(4):249, 2006.

[21] Brian Fitzgerald. The transformation of open source software. *MIS Quarterly*, 30(3):587–598, 2006.

[22] Lucy L Gilson, M Travis Maynard, Nicole C Jones Young, Matti Vartiainen, and Marko Hakonen. Virtual teams research: 10 years, 10 themes, and 10 opportunities. *Journal of Management*, 41(5):1313–1337, 2015.

[23] Samuel D Gosling, Peter J Rentfrow, and William B Swann Jr. A very brief measure of the big-five personality domains. *Journal of Research in Personality*, 37(6):504 – 528, 2003.

[24] R.M. Grant. Toward a knowledge-based theory of the firm. *Strategic Management Journal*, 17(Winter):109–122, 1996.

[25] Roger Guimera, Brian Uzzi, Jarrett Spiro, and Luis A Nunes Amaral. Team assembly mechanisms determine collaboration network structure and team performance. *Science*, 308(5722):697–702, 2005.

[26] Charlotte Hess and Elinor Ostrom. Introduction: An Overview of the Knowledge Commons. In [27], editor, *Understanding Knowledge as a Commons. From Theory to Practice*, pages 3–26. 2006.

[27] Charlotte Hess and Elinor Ostrom, editors. *Understanding Knowledge as a Commons. From Theory to Practice*. MIT Press, december 2006.

[28] IEEE Spectrum. The 2016 top programming languages. Available from: http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages, 2016.

[29] Ross Ihaka and Robert Gentleman. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.

[30] Sirkka L. Jarvenpaa, Kathleen Knoll, and Dorothy E. Leidner. Is anybody out there? Antecedents of trust in global virtual teams. *Journal of Management Information Systems*, 14(4):29–64, 1998.

[31] Konstantinos V. Katsikopoulos, Ian N. Durbach, and Theodor J. Stewart. When should we use simple decision models? A synthesis of various research strands. *Omega*, 2017.

[32] Timothy R. Kayworth and Dorothy E. Leidner. Leadership effectiveness in global virtual teams. *Journal of Management Information Systems*, 18(3):7–40, 2002.

[33] R. L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[34] Nicole Kimmelmann. Career in Open Source? Relevant Competencies for Successful Open Source Developers. *IT–Information Technology*, 55(5):204–212, 2013.

[35] Bruce Kogut and Udo Zander. Knowledge of the firm, combinative capabilities, and the replication of technology. *Organization science*, 3(3):383–397, 1992.

[36] Srinivas Kudaravalli, Samer Faraj, and Steven L Johnson. A configural approach to coordinating expertise in software development teams. *MIS Quarterly*, 41(1):43–64, 2017.

[37] A. Leroy, V. Mousseau, and M. Pirlot. Learning the parameters of a multiple criteria sorting method. In R. Brafman, F. Roberts, and A. Tsoukiàs, editors, *Algorithmic Decision Theory*, volume 6992, pages 219–233. Springer, 2011.

[38] William B Locander, H Albert Napier, and Richard W Scamell. A team approach to managing the development of a decision support system. *MIS Quarterly*, pages 53–63, 1979.

[39] Jochen Malinowski, Tim Weitzel, and Tobias Keim. Decision support for team staffing: An automated relational recommendation approach. *Decision Support Systems*, 45(3):429–447, 2008.

[40] Robert R. Mc Crae and Paul T. Costa. Reinterpreting the Myers-Briggs Type Indicator From the Perspective of the Five-Factor Model of Personality. *Journal of Personality*, 57(1):17–40, 1989.

[41] Patrick Meyer and Alexandru-Liviu Olteanu. Integrating large positive and negative performance differences into multicriteria majority-rule sorting models. *Computers & Operations Research*, 81:216 – 230, 2017.

[42] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–97, March 1956.

[43] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. Examining the Relationship between Socialization and Improved Software Development Skills in the Scratch Code Learning Environment. *Journal of Universal Computer Science*, 22(12):1533–1557, December 2016.

[44] V. Mousseau, L.C. Dias, and J. Figueira. Dealing with inconsistent judgments in multiple criteria sorting models. *4OR*, 4(3):145–158, 2006.

[45] V. Mousseau, L.C. Dias, J. Figueira, C. Gomes, and J.N. Clímaco. Resolving inconsistencies among constraints on the parameters of an MCDA model. *European Journal of Operational Research*, 147(1):72–93, 2003.

[46] V. Mousseau, J. Figueira, and J.P. Naux. Using assignment examples to infer weights for ELECTRE TRI method: Some experimental results. *European Journal of Operational Research*, 130(2):263–275, April 2001.

[47] V. Mousseau and R. Słowiński. Inferring an ELECTRE TRI model from assignment examples. *Journal of Global Optimization*, 12(2):157–174, 1998.

[48] Isabel Briggs Myers, Mary H McCaulley, and Robert Most. *Manual: A guide to the development and use of the Myers-Briggs Type Indicator*, volume 1985. Consulting Psychologists Press, Palo Alto, CA, 1985.

[49] Michael D Myers. Qualitative research in information systems. *Management Information Systems Quarterly*, 21(2):241–242, 1997.

[50] Dawn Nafus. 'Patches don't have gender': What is not open in open source software. *New Media & Society*, 14(4):669–683, 2012.

[51] A. Ngo The and V. Mousseau. Using assignment examples to infer category limits for the ELECTRE TRI method. *JMCDA*, 11(1):29–43, November 2002.

[52] John Noll, Sarah Beecham, Ita Richardson, and Clodagh Nic Canna. A global teaming model for global software development governance: A case study. In *Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference on*, pages 179–188. IEEE, 2016.

[53] A-L. Olteanu and P. Meyer. Inferring the parameters of a majority rule sorting model with vetoes on large datasets. In *DA2PL 2014: From Multicriteria Decision Aid to Preference Learning*, pages 87–94, 2014.

[54] Beatrice Rammstedt and Oliver P John. Measuring personality in one minute or less: A 10-item short version of the big five inventory in English and German. *Journal of research in Personality*, 41(1):203–212, 2007.

[55] Joseph Reagle. "Free as in sexist?" Free culture and the gender gap. *First Monday*, 18(1), 2012.

[56] B. Roy. The outranking approach and the foundations of ELECTRE methods. *Theory and Decision*, 31:49–73, 1991.

[57] B. Roy. *Multicriteria Methodology for Decision Aiding*. Kluwer Academic, Dordrecht, 1996.

[58] Ann Marie Ryan, Darin Wiechmann, and Monica Hemingway. Designing and implementing global staffing systems: Part II - best practices. *Human Resource Management*, 42(1):85–94, 2003.

[59] Saonee Sarker, Manju Ahuja, Suprateek Sarker, and Sarah Kirkeby. The role of communication and trust in global virtual teams: A social network perspective. *Journal of Management Information Systems*, 28(1):273–310, 2011.

[60] Walt Scacchi. Free/open source software development: Recent research results and methods. *Advances in Computers*, 69:243–295, 2007.

[61] Herbert Alexander Simon. Administrative behavior; a study of decision-making processes in administrative organization-3. 1976.

[62] O. Sobrie, V. Mousseau, and M. Pirlot. Learning a majority rule model from large sets of assignment examples. In *ADT*, volume 8176 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2013.

[63] Greg L Stewart. A meta-analytic review of relationships between team design features and team performance. *Journal of management*, 32(1):29–55, 2006.

[64] Klaas-Jan Stol, Muhammad Ali Babar, Paris Avgeriou, and Brian Fitzgerald. A comparative study of challenges in integrating open source software and inner source software. *Information and Software Technology*, 53(12):1319–1336, 2011.

[65] Klaas-Jan Stol and Brian Fitzgerald. Two's Company, Three's a Crowd: A Case Study of Crowdsourcing Software Development. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 187–198, New York, NY, USA, 2014. ACM.

[66] D.J. Teece, G. Pisano, and A. Shuen. Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7):509–533, 1997.

[67] Alexis Tsoukiàs. On the concept of decision aiding process: an operational perspective. *Annals of Operations Research*, 154(1):3 – 27, October 2007.

[68] J Rodney Turner and Ralf Müller. On the nature of the project as a temporary organization. *International journal of project management*, 21(1):1–8, 2003.

[69] Giovanna Varni, Gualtiero Volpe, and Antonio Camurri. A system for real-time multimodal analysis of nonverbal affective social interaction in user-centric media. *IEEE Transactions on Multimedia*, 12(6):576–590, 2010.

[70] Bill Venables, Dave Smith, Robert Gentleman, and Ross Ihaka. *Notes on R: a programming environment for data analysis and graphics*. University of Auckland, 1998.

[71] Ruth Wageman. How Leaders Foster Self-Managing Team Effectiveness: Design Choices Versus Hands-on Coaching. *Organization Science*, 12(5):559–577, 2001.

[72] Stephen C Wingreen, J Ellis Blanton, Sandra K Newton, and Madeline Domino. Assessing the IT training and development climate: an application of the Q-methodology. In *Proceedings of the 2005 ACM SIGMIS CPR conference on Computer personnel research*, pages 12–23. ACM, 2005.

[73] Stefan Wuchty, Ben Jones, and Brian Uzzi. The Increasing Dominance of Teams in the Production of Knowledge. *Science*, 316:1036–1039, May 2007.