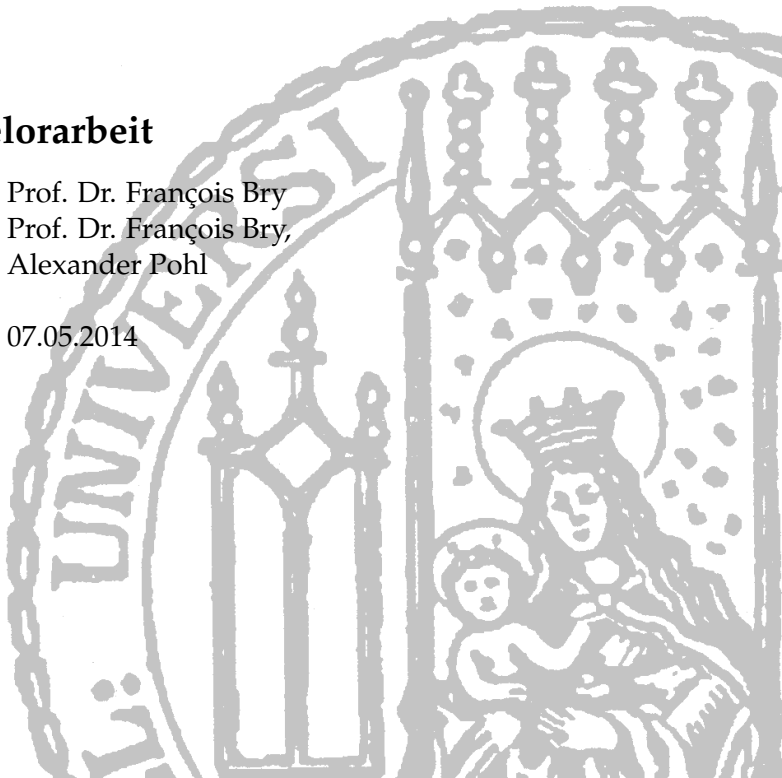
INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

TEAM FORMATION IN E-LEARNING

Marco Hoffmann



Bachelorarbeit

Aufgabensteller	Prof. Dr. François Bry
Betreuer	Prof. Dr. François Bry, Alexander Pohl
Abgabe am	07.05.2014

Acknowledgement

This thesis has benefited greatly from the support of various people, whom I would like to express my gratitude to.
I would like to thank Prof. Dr. François Bry for giving me the chance to investigate the interesting topic of team formation as well as for his inspiring ideas and comments. I would also like to thank him for his guidance on scientific writing that has allowed me to hone my writing skills.
I am also very grateful to my advisor Alexander Pohl for his highly competent notes and suggestions and the extremely productive discussions we had. This work has greatly gained from them. Thank you for your openness and insightful thoughts, which have often allowed me to see things from another perspective.
A special thanks goes to my friends, who have supported me morally throughout the past eight months. I explicitly want to thank Caroline Zygar for providing expertise and references on certain psychology topics.
Last, but not least, I want to thank my parents for providing moral and financial support throughout my studies and for allowing me to enjoy the invaluable broad education that has helped me reach the point where I am now.

Declaration

I hereby state that I wrote this document without any other help and without making use of any other sources than explicitely mentioned.

München, 07.05.2014 Marco Hoffmann

Abstract

This thesis is devoted to team building in collaborative eLearning environments. It proposes a novel approach to team building and its implementation in a partially functional proof-of-concept prototype that assists students and lecturers in forming teams by suggesting appropriate team members or whole team compositions. The appropriateness is based on predefined formation parameters that can be adjusted by students and lecturers, determining whether a group is more homogenous or heterogeneous in regard to a student's personality or knowledge. For study groups, an approach is introduced that considers the zone of proximal development. Depending on the parameters, personality traits, knowledge, the socioenvironment and temporal commitment preferences of a student are taken into consideration by the team formation algorithm to calculate a compatibility score between potential team candidates. These data categories are weighted taking into account the usage scenario at hand. For group compositions, an iterative method and a constraint optimization-based approach are presented. This work also elaborates on potential social media and gamification extensions to the tool in order to increase the users' motivation and engagement.

Contents

CHAPTER 1

Overview

Active learning, stating that students are supposed to actively discuss, create meaning, and solve problems together with peers, has received much attention. Hence, many researchers and practicioners recently focus on the establishment of active learning in higher education. This endeavor is exciting, since traditional higher education can be characterized by large-class lectures and a thorough passivity of students. Therefore, support through software is inevitable in enabling group work in large classes. This thesis deals with the design of an automatic group formation system that assists learners and teachers in forming "better" groups in terms of performance, e.g. productivity or creativity depending of the goals to be achieved. The procedure takes metrics of personal traits into account. These metrics have recently shown to play a role in team formation [SRV+13, SRS14]. Therefore, this thesis suggests a weighted scoring algorithm, the Compatibility Score Algorithm, according to which students' suitability to groups is measured. This iterative procedure is then compared to a constraint-based approach. To increase engagement and motivation of students to learn in groups, this thesis suggests to extend the team formation with elements known from social media.
The outline of this thesis is as follows:
The second chapter *Assisting Learners and Teachers: A Model For Team Formation* deals with an appropriate data model for group formation.
The third chapter *Establishing Data Models: Student and Team Attributes* investigates on the nature of said individual aspects and tries to find answers to what information about students can be used and gathered without bothering them too much.
The fourth chapter *Optimizing Team Compositions: Formation Parameters* elaborates on how the information available can be processed. Since teams can have varying purposes, the manner in how students are optimally grouped may vary as well. Thus, the gathered information can be of varying importance, and team composition algorithms may need to be adjusted.
In the fifth chapter *Creating Teams: Compatibility Score Algorithms* an algorithm that calculates a so-called compatibility score between two students is discussed. This

score is used for determining optimal team setups. The advantages and drawbacks of an iterative algorithm to a constraint-solving based approach are also compared in this chapter.
The sixth chapter *Increasing The Environment's Attractiveness and Liveliness* explores the necessity of being able to provide feedback to peers in the learning group. Moreover the chapter illustrates the employment of game mechanisms designed to keep users playing. Such mechanisms can be used on the team formation platform. Thus overall the chapter looks into incentives for students to actively take part in the team formation platform's community, making it a lively and interesting platform that is deemed useful by its users.
The seventh chapter *Extending Versatility Of The Team Formation Platform* presents some additional extensions for the social media part of the platform. It also takes into consideration additional features that make the platform an interesting tool for team formation scientists, as it can be used to experiment with different team formation setups.
The eighth chapter *Prototype* explains the capabilities and design of the prototype that was implemented in the course of this thesis. In this chapter a download reference for the source code and binaries can be found by interested readers.
The last chapter *Outlook and Future Work* provides an overlook about some aspects of this work that could be continued on working, as they require deeper and more extensive consideration as well as additional research.

CHAPTER 2

Assisting Students and Teachers: A Model For Team Formation

In a university context, teams are often formed in two ways: Either the lecturer groups students until every student is in a team, or it is left to the students themselves to find peers and form teams. In the latter case, there are often students left which did not find teammates and are then assigned by the lecturer. Both approaches differ significantly in regard to the liberties granted to the student, and in terms of information available to a possible formation tool: in case of the teams being formed by the lecturer, student autonomy is limited and information about the students may be demanded. However, when students form teams themselves, they should be granted autonomy and it should be left to them to decide which information they provide. This is especially important, because in the case of the system forcing them to provide information, the readiness to use the tool will most likely decrease. For both approaches, the tool should assist the user (lecturer or students) in forming teams by suggesting appropriate members.

2.1. Assisting Student Team Formation

Students form teams for a variety of reasons. They might want to tackle the course's problem sets with peers or generally discuss course-related material. Maybe they want to solve specific self-defined problems or practice a very specific topic in depth (and nothing else). Others may want to start a big project for which they need the skill, help or creative input of peers.
Regardless of the goal of a specific team, the formation process of student groups often proceeds as described in the following figure 2.1.: It starts with a student A asking a student B to form a team.

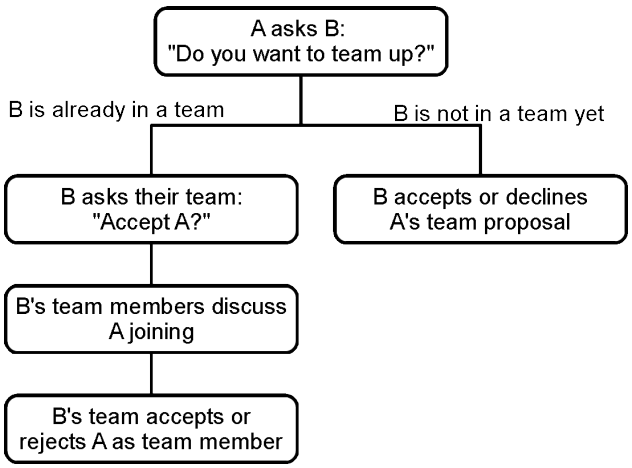


Figure 2.1.: A typical team formation procedure.

Often, B is either a friend, acquaintance or to some extent perceived similar person to A [Bek05]. If that is not the case, A will probably ask a random student. In a pool of students with few experience in team formation and very limited to no knowledge about each other, it is likely that suboptimal teams in regard to team performance will be formed. Given student A looking for team members, the team formation tool can assist A by suggesting team member candidates that are most fitting using the information that the tool has about both students. This information could be on

- the students' personalities. The tool could suggest candidates with specific personality traits so as to create personality-wise diverse or homogenous teams depending on the purpose and task-related goals of team formation.
- the students' knowledge. Depending on the goal of the team, the tool could suggest candidates with different knowledge fields compared to student A to create a universally skilled team, or suggest candidates with the same knowledge fields so as to create a very specialized team.
- the students' personal preferences. The tool could suggest candidates that have similar personal preferences so that conflict and organizational overhead within a group is minimized.

The student should be able to adjust the suggestion parameters as they deem it fit. Granting this freedom also means that an implementation should have a strong focus on explaining available settings and their probable effects, so that it can assist even when a student decides to when a students decides to deviate from the default settings. As mentioned before, teams can have different purposes. When creating a new team, it is thus necessary to specify what the team is actually about and what is required to join, so that interested students can quickly decide if they want to apply for the team.

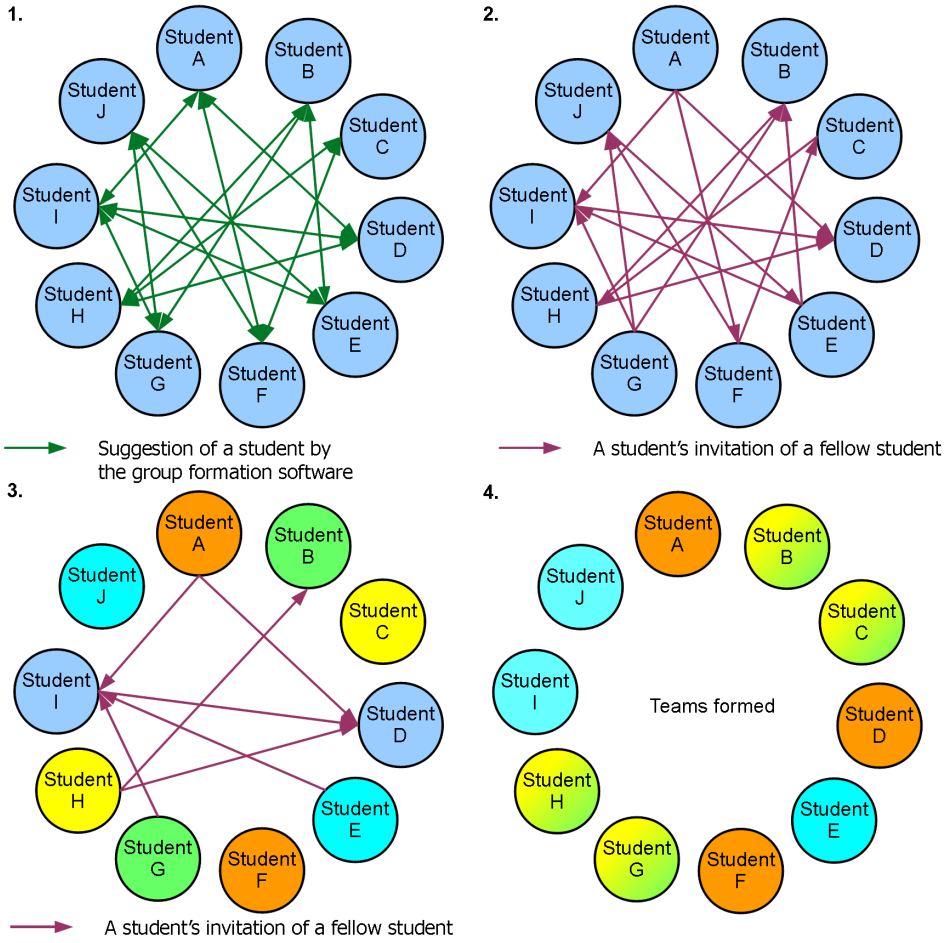


Figure 2.2.: An assisted lecture-based team formation procedure for students.

After suggestions have been made, it will ultimately be left to the student whether they want to go along with the suggestions or ignore them. If they accept the tool's suggestion, an invitation is sent to suggested student B, who may then evaluate whether A is a fit candidate for their team. Student B may either decline the invitation or accept it, in which case A and B are grouped together. All other pending invitations of A and B are put on hold for a certain amount of time, during which the new team member can veto against them. In cases where A or B are not individual students, but existing teams and comprise of more than one member, all

actions that concern the whole team will require a voting. Team actions might be

- Inviting new members
- Accepting join requests
- Expelling members
- Disbanding the group

However, this is only possible when either A or B represents a team, not both. Merging two teams is only reasonable when both teams have similar goals and requirements: it would not make sense to merge a team which wants to study analysis with a team that wants to improve their Japanese skills. Nevertheless, in some cases, it is desirable to merge two teams. The following example demonstrates a potential problem that could arise if team merging was not allowed.

Example 2.1.1. 20 students in a lecture are advised to form study groups with a team size of 5. They all begin to look for team members, adding their desk neighbor to their team first. After a short while, there are 10 teams with 2 students each. No individual student remains, and the team formation procedure terminates.

To tackle such problems, two team flags are introduced: *Lecture-based Team* and *Specific Student Team*. A Lecture-based Team is a default team template provided by the lecturer for types of teams that are to be expected to be created frequently. Study groups without a specific focus on a topic or teams that are needed in the context of the course while leaving the freedom of team formation to the students are typical use cases for such a Lecture-based Teams. Lecture-based Teams do have the same requirements and goals, merging teams is therefore possible. On the other hand, a Specific Student Team can be completely customized by students, but does not allow team merging, as it is hard to determine whether two custom teams really do have the same goals and requirements.

2.2. Assisting Lecturer Team Formation

In some cases, e.g. practicals or lab exercises, teams are created on a mandatory basis and thus team formation is not left to the students, but it is the lecturer who arranges teams. In many cases, the lecturer has to handle a big audience and does not have the time to get to know every student personally. Thus, it is likely that teams will be arranged randomly, or at least using a method that does not take into account each individual student's personality, knowledge and preferences. In such scenarios, the proposed team formation tool can assist the lecturer by suggesting complete team arrangements.

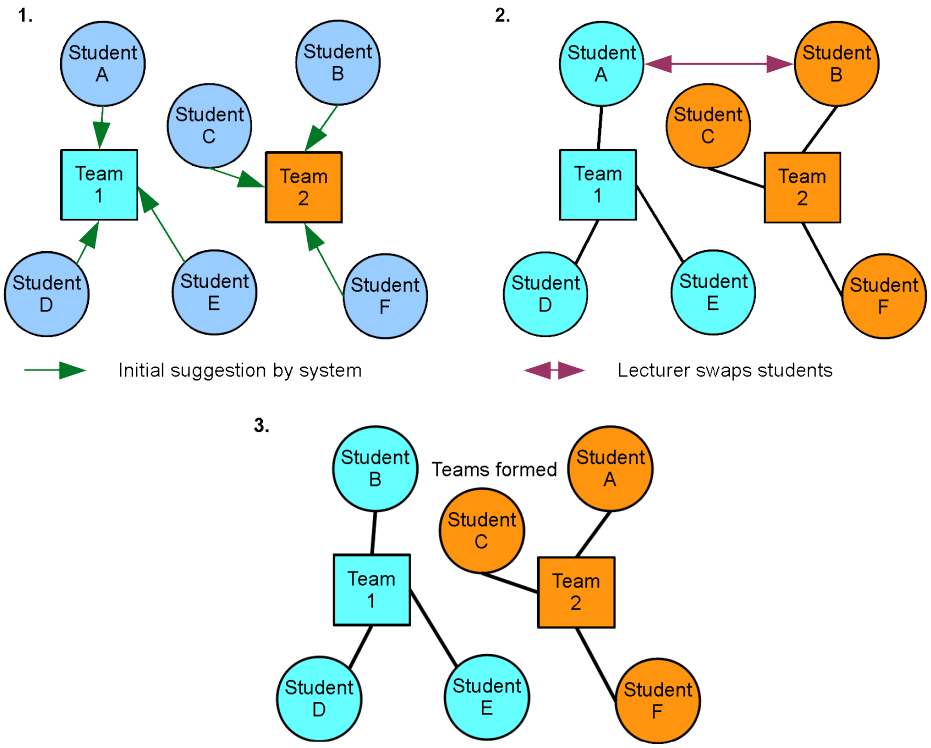


Figure 2.3.: An assisted team formation procedure for lecturers. The lecturer may apply changes to the team setup.

After the suggested assignment, the lecturer has the freedom to rearrange teams as they deem fit. If the lecturer is satisfied with the result, the students are informed about the team assignments.
The normal approach is to partition the class by teams, generating as much teams as needed so that a preselected team size is met. In some cases though, only a limited number of students can be accepted into a course or practical. In such a scenario, it is likely that more students apply than team slots are available. In this case, a lecturer would often employ a "first come, first served"-strategy. However, the tool can further assist the lecturer by offering a "best of"-preselection, which ranks students according to predefined attributes, such as previous grades, semester count or even a specific personality trait, using a predefined ranking mode (ascending/descending). It is thus not necessary to block applications after a certain number of students have applied: Instead, everyone may apply during the application period and after that, reasonable attributes for a ranking are selected. The top students in the generated ranking can then be assigned to teams.

CHAPTER 3

Establishing Data Models: Student and Team Attributes

Before proceeding to elaborate the team formation procedures, this chapter discusses what data can be reasonably used to actually make teams more productive, creative or efficient than random teams.
The first section outlines data connected to the entity "student" that may be relevant to team formation. It is inspired by the findings of Spoelstra et al., who introduced the categories *Personality*, *Knowledge* and *Preferences* as three important factors to team formation, which supports the initial assumption that specific personality traits and different knowledge fields within a group do affect team performance and can thus be used to optimize the students' team experience [SRV+13, SRS14].
The second section will on the one hand deal with parameters that affect group-related characteristics of a team, such as team size or designated team longevity. Adjustable team attributes are necessary to adapt to different environments and team purposes.

Example 3.0.1. A study group may have an ideal group size that is rather small, whereas huge software development projects may require more students within a team.

On the other hand, some team attributes may be the sum of the corresponding attributes of the team's members (student attributes) and can be automatically derived.

Example 3.0.2. The average team member age can be derived by calculating the mean average of all the students' ages. The average team semester count can be derived likewise.

3.1. Student Attributes

In this section, the key attributes *student personality traits*, *knowledge* and *preferences* are examined and elaborated. They need to be processible by the team formation

algorithms, so the following questions need to be answered for each of them.

- How, if at all, can information about this attribute be gained?
- Are there any scales or metrics that offer a sufficient representation of the extracted information? Representations need to be processible by team formation algorithms, but at the same time must be as accurate and close to reality as possible, such as the Big Five metrics for personality characteristics[JS99].

There are other student attributes that might also play a role in team formation, but have not been not included in this thesis because no conclusive evidence could be found on how they improve teams in any way given an attribute-specific team composition. The subsection *Attributes Not Considered In Team Formation* elaborates on these attributes, as they might gain importance as new research is done.

3.1.1. Personality Traits

The most inherent attribute of a human being regarding the way they interact with their surroundings and the world is probably their personality. Personality is important for the data model, because teamwork is mainly about interaction between individuals. In teams some people tend to be leaders, while others prefer to assume supportive roles. Some are extroverted, while others prefer to get their work done in silence. Since personality plays a role when it comes to forming effective teams, personal characteristics should be considered in team formation [NWC99, HC07].
A sufficient representation of this trait is provided by the well established and often used Big Five metric, which measures the following dimensions [JS99]:

- *Openness*: The tendency to be imaginative or open to unusual ideas or adventure; a higher score correlates to a preference of a variety of activities in contrast to a strict routine.
- *Conscientiousness*: The tendency to be aware of one's duties and be self-disciplined; a higher score correlates to planned behavior and dependability.
- *Extraversion*: The tendency to be talkative, outgoing and assertive, to seek the company of others.
- *Agreeableness*: The tendency to be cooperative and trustful rather than having a hostile basic attitude.
- *Neuroticism*: The tendency to fall victim to negative emotions such as anger, depression or anxiety easily; a low score correlates to high emotional stability.

These dimensions are calculated with the help of a survey that students have to take, yielding a 5-dimensional numeric vector. There are many surveys available with a varying number of questions such as the NEO-PI-R (240 items) or the shorter version, the NEO-FFI (60 items) [CM85]. While more items certainly increase the accuracy of the resulting Big Five scores, it should be noted that long,

time-consuming tests will probably decrease acceptance among students or provoke thoughtless answers in order to shorten the time needed to complete the test. A middle course needs to be found here to acquire reconciliation.
A longitudinal study about personality change in young adulthood carried out by Robert et al. [RFRT01] suggests that although changes are possible, there is great continuity in personality traits. It can be concluded that updating personality data does not need to happen often. Therefore, it is suggested to offer to retake the test on a voluntary basis or to annually prompt the user to retake it.

3.1.2. Knowledge

When referring to the term knowledge, a student's skills, experience and general familiarity within a given topic or field is meant in this thesis.
It is reasonable to take student knowledge and skill into account in the process of assigning students to teams, because some kinds of teams, e.g. project teams with a specific task, may require a minimum level of knowledge on a topic in order to ensure that no team member slows their peers down too much and no peer is left behind. To supervisors, it may also serve as a guarantee that the assigned team actually does have a chance to jointly complete a task successfully. For other teams, a team formation algorithm might not regard minimum or maximum scores of knowledge, but rather seek to group students with complementary skill sets, so that they can learn from each other.
The required knowledge fields have to be provided by another entity, for example a lecturer, their assistants or a student who created a project team announcement. Once the topics, or knowledge areas, have been defined, students can self-assess their knowledge levels on a numeric scale. A longitudinal study among medical students conducted by Fitzgerald et al. [FWG03] states that self-assessment accuracy is not significantly related to gender, ethnicity or academic variables. Thus, self-assessment seems to be a suitable means of gathering knowledge-related data. The following scale that ranges from 0-5 is implemented.

- 0 – unheard of, absolutely no knowledge
- 1 – shallow knowledge, severe difficulties to apply knowledge
- 2 – theoretical knowledge, but severe difficulties to apply knowledge
- 3 – can apply knowledge, often requires rechecking/looking up further information
- 4 – can apply knowledge, most of the time without mistakes/additional help
- 5 – deep understanding of this knowledge area

An even-numbered Likert scale is chosen so as to have students express a tendency towards their levels of proficiencies. This will yield acceptable results even when employed in a very demotivated community, or a pool of students with a very homogenous proficiency distribution. If teams are matched based on heterogeneous

knowledge areas, i.e. a team consists of members with different skill sets, it is favorable to have no middle ground within the scale. It should also be noted that self-assessment is not the only way to gather data on a student's knowledge. A team formation tool used alongside digital classroom software such as *Backstage* [PGBB11] could profit from Learning Analytics - the analysis, measurement and reporting of data about learners and their context - as additional data based on quiz results or questions asked can be gathered. Also, past grades or problem set ratings yield further conclusions about an applicant's depth of knowledge within a given field. Additional refinement of the self-assessment results could be achieved like this.
Knowledge levels are likely to change; within a short period of time, familiarity with a certain topic can be increased drastically. Furthermore, knowledge areas have to be defined individually for every team announcement. It is therefore necessary to prompt the student to provide a self-assessment every time they express their wish to join a team. Knowledge areas should be defined precisely and kept to a minimum necessary number. Redundancy between areas should be avoided so as to avoid causing too much of a hassle.

3.1.3. Preferences

In the work of Spoelstra et al. [SRV+13], preferences include attributes like availability and time zones, and serve as hard constraints. This subsection will deviate from that definition: here, preferences are defined as additional attributes that, when factored in, should further optimize the student's group experience.

3.1.3.1. Pre-existing Relationships

It is quite common that students get to know each other and form friendships. Considering friendships when forming teams, however, does not seem to be a good idea. Although it is conceivable that friendship-based groups are able to communicate better, in general, non-friend groups seem to be more efficient [Sco60]. Bekele's analysis shows that (self-selected) groups of friends tend to take work less seriously, and prioritize chat over studying or group work [Bek05]. Spoelstra et al. note that friendship-based groups may hamper the exchange of different ideas, and self-selection in general leads to people with similar abilities flocking together, ultimately preventing students with different learning styles from learning from each other [SRS14]. Finally, Chapman et al. notes that filling up a group of friends with random students may cause the random students to not contribute as much due to lack of cohesiveness between the initial group and the new students [CMTW06]. However, it can not be avoided that friendship-based groups are formed: Students could easily figure out if the team formation tool separates friends into different teams. In that case they would not use it for voluntary groups, and would not admit their friends to the system when the lecturer is in control of forming teams. Even so, the other side of the coin should be considered. If there are hostilities between two students due to personal reasons, they should be given the opportunity to express their desire to not be grouped with each other. It is conceivable

that otherwise conflicts within the group or even sabotage might occur, ultimately compromising the efficiency of the whole team. An avoidance list is therefore implemented that may be used to keep a list of students one does not wish to be grouped with.

3.1.3.2. Temporal Preferences

It is important for study group members to be roughly similarly dedicated. Students willing to spend a lot of time will probably be disappointed if their peers lose motivation quickly, and vice versa, a student with a low desired expenditure of time in a team with very dedicated people will eventually be left behind. In order to take this into account, the attribute *expenditure of time* is introduced: Upon applying for a study group, students will simply be asked how many hours per week they intend to spend on the subject. Linked to expenditure of time is the manner in which this time is spent: *division of time*. This attribute is about whether the student prefers more meetings with less time expenditure, or less meetings with more time expenditure.

3.1.4. Organizational Information

The organizational information elaborated here does not directly contribute to the team formation process, but may on the one hand serve as a filter, on the other hand allow the student to make a more informed decision about potential team members.
Students are in a specific semester and have a specific age. The semester count does increase biannually, but taking off semesters is not too uncommon. At the start of each semester, students should thus be reminded to update their semester when logging in. Recording this attribute might be of interest to see if there is any correlation between semester count and readiness to form certain kinds of teams. It may also serve as a filter for a best-fit approach.

Example 3.1.1. At the LMU, workshop and seminar slots in computer science are sometimes given to those applicants who have the highest semester count. In such cases, assigning applications to the available slots can be done by the team formation tool. It should be noted that the semester count needs to be verified somehow, so as to avoid students cheating the system by deliberately entering semester counts that are too high. In regard to the LMU computer science faculty, necessary data can be retrieved from UniWorX.

Finally, the property "is online" is introduced. That boolean property is set to true automatically when being logged in. It may serve as a requirement when lecturers want to form teams spontaneously during a lecture. After all, it would not make sense to form teams of temporary nature with students that are actually not attending.

3.1.5. Attributes Not Considered In Team Formation

When examining other aspects that distinguish one student from another and might be relevant to team formation, it is standing to reason that demographic information might be a candidate, such as

- Country of Origin
- Spoken Languages
- Cultural Background
- Religion

However, as findings by Watson et al. [WKM93] suggest, cultural differences seem to only play a role at the beginning of a task, and culturally heterogeneous team performance converges towards that of culturally homogenous teams, yielding no significant difference after about three months. Although this indicates importance for teams that are meant to last for a shorter amount of time - which is the case for most student groups - other work [MN05, WJKC98] lead to the conclusion that optimizing a team based on demographic aspects is not an easy task, requiring complex moderators, otherwise yielding only marginal to no benefits. In this work, given that the teams consist of students, i.e. inexperienced individuals in regard to leadership and team moderation, and due to the uncertainty regarding practical relevance, demographics will not play a role, although they should definitely be kept in mind for future revisions.
Languages are usually very important for team formation: if team members are unable to communicate, collaborating is hardly possible. However, it is assumed that students in a university course can at least, as the lowest common denominator, speak the course's primary language.

3.2. Team Attributes

Having outlined all fundamental attributes of potential team members, required team attributes are elaborated in this section. Some of them will be predefined and thus set before an actual team has been formed, serving as formation parameters. These include the maximum and minimum team size or disbandment options. Other attributes will be derived from the team members.

3.2.1. Team Size

Team Size is an attribute represented by a numeric interval that determines the minimum number of individuals needed to form a team as well as the maximum team size. Many studies suggest the optimal team size for study related work lies between three to five team members [Bek05, OFBE04]. Although it is also suggested that the lower end of this interval, three members, are more optimal for study groups and the higher end, five members, are better for project work, the formation tool will, by default, implement the [3,5] interval. An interval is necessary because a fixed value would lead to unassigned students if the number of participants is not divisible by said fixed value. For a number of participants greater than or equal to x, an interval $[x, 2x-1]$ ensures that all students can be assigned to a group. This assumes that all participants should be assigned to a group. For student-assisting team compositions, as in when the tool only suggests new members to the teams, this value may be ignored and is only shown as a hint. With a limited number of total teams, the minimum value should be equal to the maximum value of the interval, or else it will not be clear how many student slots are available.

3.2.2. Longevity and Disbandment

Study groups will most likely end after the course's exam, while project teams will in most cases cease after the project has been finished. Other team purposes may be very specific, depending on the requirements of lecturers or students proposing groups. The team formation tool proposed here will offer several methods of handling time-based group disbandment.

- *Manual Disbandment* will terminate and evaluate a team when all team members express their wish to end the group. This is suitable for teams without a known termination date.
- *Automatic Disbandment* requires a team life span that the lecturer or team leader has to provide. This allows for spontaneous team formation during lectures as well as for project groups with a known termination date.
- *Oakley's Method* will disband the team automatically after a set period of time unless every team member explicitly expresses their wish to remain together [OFBE04]. Although Oakley suggests 4-6 weeks as a time interval, it is possible to override this value.

Apart from these methods, it should always be possible for the lecturer to disband a group as an administrative function.

3.2.3. Cumulative Attributes: Deriving from Team Members

Some team attributes emerge as the sum of attributes from it's members.

- *Team Knowledge Sum* is the sum of all members' knowledge values.
- *Team Knowledge Average* is the average of all members' knowledge values.
- *Average Age* is the arithmetic average of all members' ages.
- *Minimum Age* is the age of the youngest team member.
- *Maximum Age* is the age of the oldest team member.
- *Average Semester* is the arithmetic average of all members' semester counts.
- *Minimum Semester* is the lowest semester count of all team members.
- *Maximum Semester* is the highest semester count of all team members.
- *Team Member Count* is the number of team members.
- *Team Time Dedication* is the arithmetic average of the hours members want to spend on group work.
- *Team Age* is the timespan since the creation of the team.

These attributes do not need to be gathered manually. They are derived automatically from the base data provided. It should be noted that an increase in *Team Knowledge* can also be used as an indicator for the success of a team, although updates may happen too irregularly to be of any use.

3.3. Chapter Summary

A summary of this chapter with the different attributes that are taken into consideration is shown in figure 3.1.

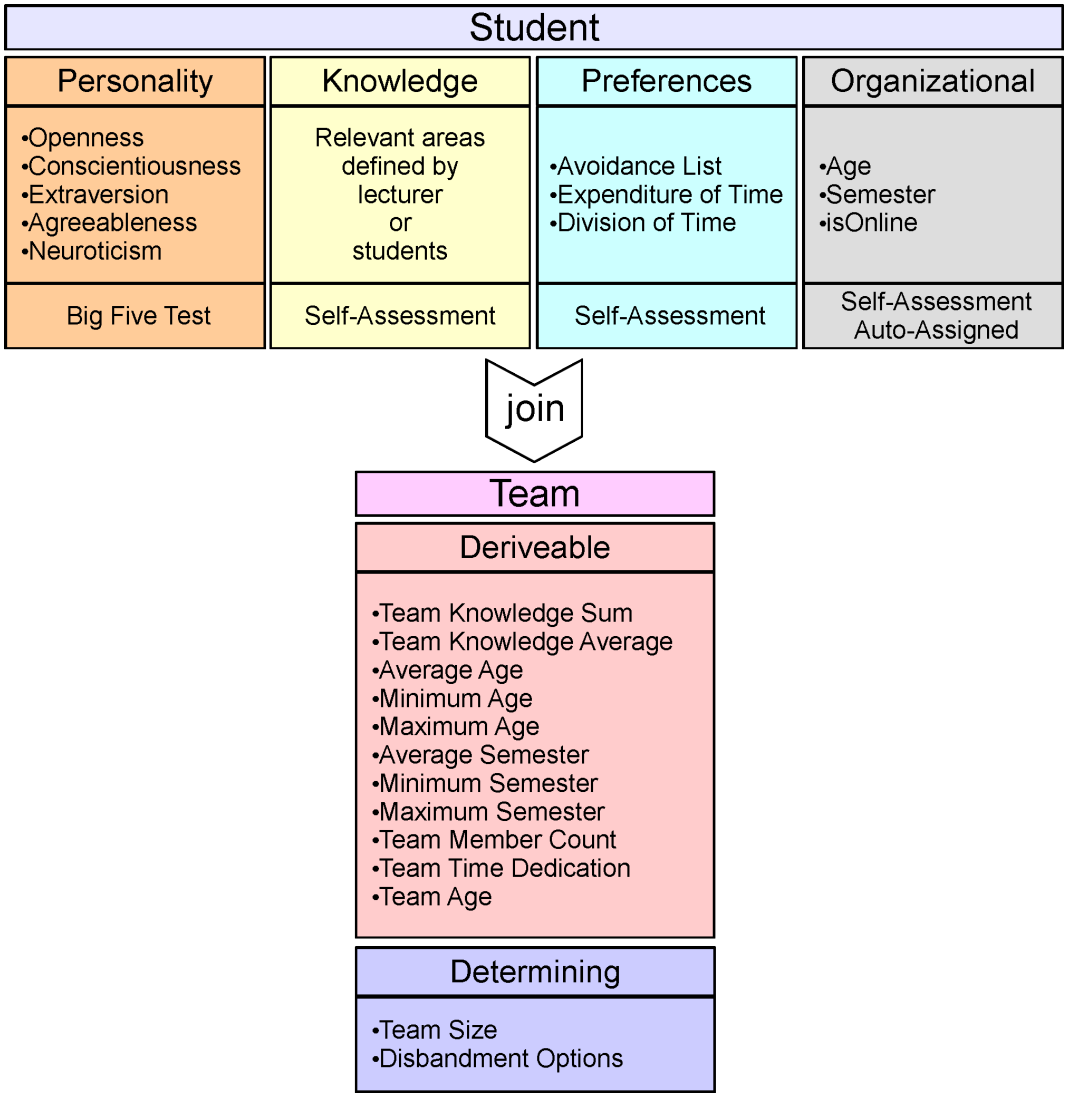


Figure 3.1.: All attributes used for team formation.

CHAPTER 4

Optimizing Team Compositions: Formation Parameters

In chapter 3, the available data for an optimized team formation was elaborated. It is, however, still not clear how an optimized team composition can be achieved. This thesis suggests to calculate a compatibility score for each category between the following entities:

- Between two students
- Between two teams
- Between a team and a student

Per category, this score can either be high when the attributes of the two compared entities are very close to each other (supplementary approach) or when there is a lot of variation when comparing two students' attributes (complementary approach). Spoelstra et al. suggest that depending on the envisioned outcome of project work, several team formation rules can be designed [SRS14]. This means that it may be possible to optimize a team in regard to, for example, creativity or productivity[SRV+13, Wes97]. Thus, parameters that determine how a compatibility score is calculated are adjustable to cover a wide range of potential applications. This also allows to experiment with different team formation setups.
After the compatibility scores have been calculated, they are weighted and summed up, yielding the final compatibility score and thus the order of suggested students.

4.1. Team Composition

Three major attribute categories have been defined: *Personality* (the personality trait results of the Big Five personality test), *Knowledge* (proficiency levels gathered by self-assessment) and *Preferences* (preferences regarding peers or time dedication). For the categories *Personality* and *Knowledge*, two approaches can be used to calculate a compatibility score:

- Group entities that have similar scores: This creates a homogenous team and will be referred to as *supplementary fit*.

- Group entities which scores differ the most: This creates a heterogeneous team and will be referred to as *complementary fit*.

The settings for *Personality* and *Knowledge* are considered independently of each other. It is thus possible to create groups that are homogenous in respect of personality and heterogeneous in regards to knowledge and vice versa. The following figure shows different team compositions depending on the approach used. The bars represent knowledge scores for the exemplary knowledge areas "A","B","C" and "D". The higher a bar is, the greater is the corresponding self-assessed knowledge value.

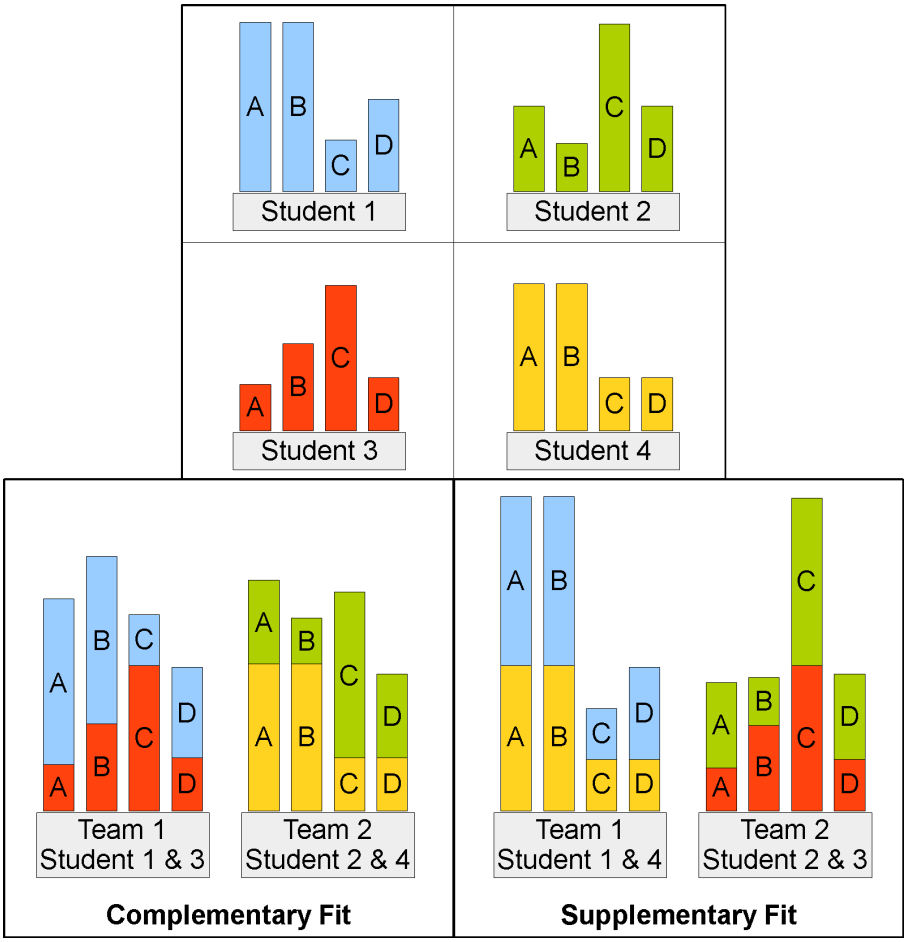


Figure 4.1.: Example *Knowledge*: A,B,C and D refer to certain knowledge areas - the higher a bar is, the higher the proficiency level is. Complementary fit yields a broadly skilled group, supplementary fit yields groups that excel in certain topics.

Preferences will always be approached using a supplementary method (closer values get higher compatibility scores): expenditure of time should roughly be the same. For other boolean options such as "is not in avoidance list", the score will be maximized if it matches and minimized if it does not.

A special case needs to be considered when it comes to the formation of study groups: the zone of proximal development [Vyg78]. This term describes the gap in skill levels between two learners: Students benefit from their peer's knowledge when it exceeds their own knowledge to a certain extent. However if a student is a lot more skilled within a certain topic, they might feel slowed down by their peers and vice versa: the lesser skilled peers might be demotivated. This is only reasonable to be applied when using a complementary approach, as a supplementary approach already punishes great differences in attribute values. Thus, a third mode is introduced: *Complementary Limited Deviation*, which increases scores as values differ up to a certain distance (maximum distance); if two values' difference exceeds a certain threshold, the score will be decreased again. This feature is highly experimental, as there currently is no research about operationalizations for knowledge and no means available to map a maximum distance in knowledge as described above onto such an operationalization. Figure 4.2 exemplifies an approximated score behavior when using different parameters.

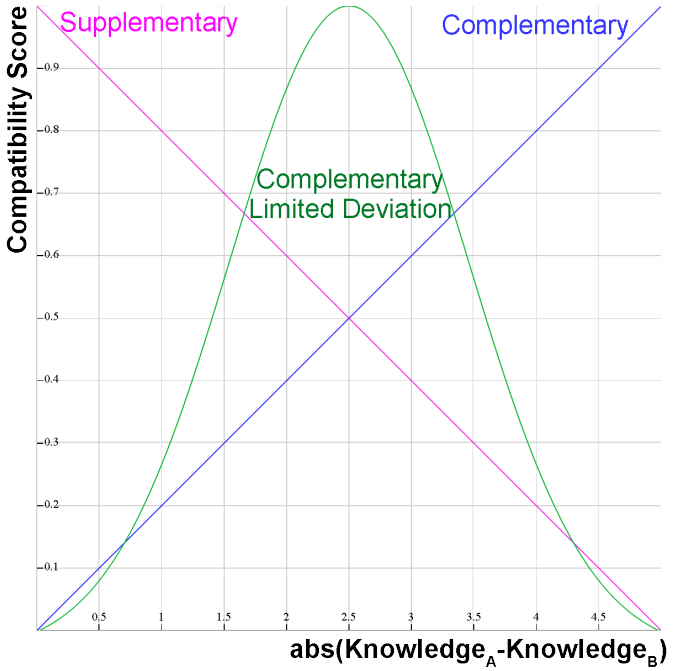


Figure 4.2.: Visualization of an exemplary knowledge score behaviour between a student A and a student B using different parameters. Please note that all other attributes apart from the operationalization of knowledge are considered in the same manner as for supplementary fit (i.e. the more similar the better)

4.2. Team Formation Mode

Two scenarios can be distinguished in the supervised team formation process, in which teams are formed by a central authority, e.g. a lecturer: Either a whole class or pool of students needs to be divided into teams, with the result that every student is assigned to a team; or the number of team slots is limited and thus the most suitable students out of a pool of students are to be admitted. Spoelstra et al. introduce the terms *best possible average solution* and *best fit team solution* [SRS14]. In this work, these terms are adapted and elaborated as following. The *best fit team solution* method adds a preprocessing step that shrinks the pool of students according to lecturer-defined weighted filters. For each attribute, a filter can be added that sorts students ascending or descending in regard to that attribute.

Example 4.2.1. If a lecturer wants a creative team, it is suggested to look for a low conscientiousness personality score [SRS14]. They can then instruct the system to sort students from a low conscientiousness value to a high conscientiousness value. Others may want to favor students with the highest semester count. In this case, students are sorted descending in regard to their semester count.

After the students have been sorted, the n top students are assigned to teams, with n being the number of available slots for that course.
If more than one filter is applied, they can be given a weight $weighting_{Category} \in [0,1]$. The final ranking is then calculated by multiplying the position of each applicant by $weighting_{Category}$.

Example 4.2.2.

Semester Count decending weighting=0.3			Conscientiousness ascending weighting=0.7			Final Ranking	
#	Student	Semester	#	Student	Conscientiousness	#	Student
1	A	7	1	B	24	2,3	B
2	D	6	2	C	26	2,3	C
3	C	5	3	A	30	2,4	A
4	B	3	4	D	33	3,4	D
5	E	1	5	E	35	5	E

The ranking positions are calculated as following:

$$X : weighting_{Cat1} \cdot position_{Cat1} + weighting_{Cat2} \cdot position_{Cat2}$$
$$A : 0,3 \cdot 1 + 0,7 \cdot 3 = 2,4 \quad B : 0,3 \cdot 4 + 0,7 \cdot 1 = 2,3 \quad C : 0,3 \cdot 3 + 0,7 \cdot 2 = 2,3$$
$$D : 0,3 \cdot 2 + 0,7 \cdot 4 = 3,4 \quad E : 0,3 \cdot 5 + 0,7 \cdot 5 = 5$$

In some cases, students will be on the exact same position, e.g. in the above figure, student B and C. If not all students on that same position can be admitted and changing the weights is not an option, a random selection regarding the equally ranked students is probably the most fair approach.

After that, the main formation algorithm is executed: It already incorporates the idea of *best possible average team*. That means that selecting *best possible average team* differs only by the absence of a preprocessing step.

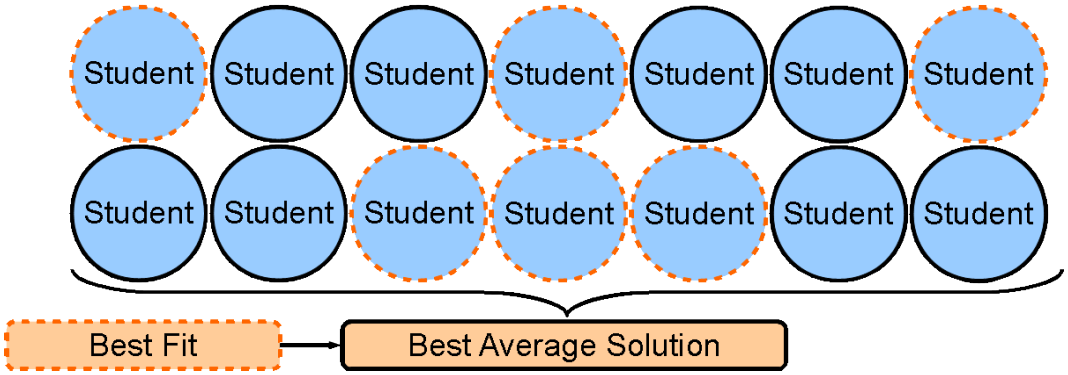


Figure 4.3.: A modular staged approach. When using *best possible average team*, all students are considered. When using *best fit*, only the most fitting students (outlined in orange) are considered. It should be noted that the lecturer defines what "most fitting" means.

4.3. Category Weighting

Each of the discussed categories yield a compatibility score between entities (students or teams). If added up, this will result in the final compatibility score. However, different categories may very well be more or less important. Thus, all category compatibility scores are multiplied with a weighting factor $\chi \in [0, 1]$ that can be freely redefined and thus allow the user to experiment with different weightings. It is, at the time, not clear which weighting configuration yields the best team results. A survey among "teachers and designers of project-based learning, who were also team formation practitioners" conducted by Spoelstra et al. revealed that while 64% consider *Knowledge* as very important or important, only 12 do the same regarding *Personality*. However, it is not clear if this might be because the respondents were unable to assess personality and thus did not regard it as very important [SRS14]. Indeed, other authors suggest that personality does play a significant role in team formation theory [Bek05, NWC99, BH97]. Each category consists of several attributes, which can also additionally be weighted using a subweighting $\zeta \in [0, 1]$. By default, the attributes within the *Personality* and *Knowledge* category are equally weighted. For the category *Preferences*, it seems intuitive that hostilities should weigh heavier than temporal preferences. All weighting and subweighting factors can be adjusted manually if specific formation demands need to be met.

CHAPTER 5

Creating Teams: Compatibility Score Algorithms

In the previous chapters, the underlying data for team formation was established and a compatibility score that is used to determine whether two peers are a good match in context of a desired team specification was introduced. This chapter finally elaborates on how such a score is calculated in this draft. A compatibility score can generally exist between two students, a student and a team or two teams. Students and teams are both compiled into objects of the class `Team`, which holds all the required attributes for team formation. Individual students are regarded as a singleton team; a team with only one member.

```
class Team{
  //Personality Attributes
  List of numbers personality; //contains the big five scores

  //Knowledge Attributes
  List of numbers knowledge; //contains the knowledge scores

  //Preference Attributes
  Set<Student> avoidanceList;
  number timeExpenditure;
  number timeDivision; //timeExpenditure divided by desired number of
      sessions per week

  //Identity
  List of number members; //The members of the team

  //Compatibility Scores
  number personalityScore;
  number knowledgeScore;
  number preferenceScore;
  number totalScore;
}
```

The preliminary step generates the list of candidates depending on the type of team (lecturer-assisted team or student-assisted team), the team formation parameters,

and whether the *origin* is a team or a student. The origin is the student or team that a compatibility score with all other students is calculated with. Before the score can be calculated, the requesting origin needs to be compiled into a `Team`.

Example 5.0.1. Student A wants to form a custom team. Since the system suggests candidates to A, A is the origin of the compatibility score request.

If the team consists of more than one member, all the numeric attributes are the arithmetic mean of the members' corresponding attributes. The avoidance list is the union of all members' avoidance lists. Therefore, "team↔team" or "team↔individual" scores can be calculated identically.

5.1. Preliminary Step "Assisting Students"

Students can either opt to create or join a custom team or go along with a lecture-based team. In the first case, there are several possible scenarios.

- A student A wishes to create a new custom team. The origin is the student A, and the candidate list consists of all students.
- A student A wants to join an existing custom team. The origin is the team, and the candidate list consists of the applying student A.
- An existing team wants to find more members. The origin is the team, and the candidate list consists of all students.

In the second case, a lecture-based team, team merging is also possible.

- A student A wishes to join a study group. The origin is the student A, the candidate list consists of all generic teams and all students.
- A generic team wishes to expand. The origin is the generic team, the candidate list consists of all generic teams and all students.
- A student or team A receives an invitation from a student or team B. The origin is A, the candidate list consists of B.

5.2. Preliminary Step "Assisting Lecturers"

When assisting lecturers, teams are not "recycled" and the tool operates on a pool of students only. However, there are two approaches: Best fit and best average team formation, with the former reducing the candidate list based on predefined criteria. Thus, there are two scenarios:

- Best fit is enabled: The origin is any student of the candidate list, and the candidate list consists of the admitted students
- Best fit is disabled: The origin is any student of the candidate list, the list consists of all applying students

An explanation about how the origin is determined is deliberately left out in this section. The section *Best Average Team - Clustering vs Constraint Solving* will elaborate on the specifics of team formation when assisting lecturers.

5.3. Calculating the Compatibility Score

Categories can be scored using a supplementary (similar values get high scores) or complementary approach (the more deviation within values, the higher the score). This can be achieved by representing the available data as vectors. For each pair of `Team`, for each category an euclidean distance is calculated: The higher the distance, the more complementary and the less supplementary the pair is in regard to that category. The euclidean distance is easy to use, yet suited as a similarity measure for this specific application: The attributes use a linear scale and only the absolute distance is relevant for the score. The following code shows how it can be calculated between a pair of arrays (representing a category, e.g. a 5-dimensional vector if the personality category is processed) x and y.

```
double euclideanDistance(double[] x, double[] y, double[] weightings)
{
  double distance = 0;
  for(int i = 0; i<x.length; i++)
  {
    distance += ((x[i] - y[i])^2)*weightings[i]; //can not be negative as
        weights are always positive
  }
  distance = sqrt(distance);
  return distance;
}
```

With the help of the euclidean distance, personality and for most scenarios, knowledge scores can be calculated. Another simple approach will be used for scoring avoidance list matches within the preference category: If there is a match, the subscore will be 0, if there is none, it will be $1 * weighting_{avoidance}$. When *Complementary Limited Deviation* is active, euclidean distance can not be used. In this case, a gaussian function is applied to the absolute distance between two knowledge scores:

$$f(x,a,b,d) = exp\left(\frac{x-(a\cdot d)^2}{b}\right)$$

The parameter $d \in [0,1]$ describes the center of the gaussian curve, which is equivalent to the ideal distance in knowledge. Since the knowledge scale ranges from 0 to 5, the maximum distance and coefficient of d is a=5. The parameter $x \in [0,5]$ describes the distance in a knowledge area between two students. The parameter a is calculated by multiplying the desired proximal distance with the maximum distance 5. The best value for parameter d needs yet to be determined and requires further investigation. The gaussian function is also used for determining the temporal preference score within the preference category. Since no maximum distance is known, normalization would prove difficult if euclidean distance were used. The maximum value for x is thus unbounded. For time division, the parameters a=0,

d=0 and b=12 seem reasonable, as it provides significantly less score when the total time expenditure differs by more than 3 hours. For time expenditure, which expresses the desired time expenditure per session, a=0, d=0 and b=4 is used. This configuration provides fairly high scores when time division, or expenditure per session, differs by less than one hour, however, the best value for parameter b needs yet to be determined.

```
double weightedGaussian(double[] x, double[] y, double a, double d,
    double b, double[] weightings)
{
  double score = 0;
  for(int i = 0; i<x.length; i++)
  {
    double distance = abs(x[i]-y[i]);
    score += exp(-((distance - (a*d))^2) / b) * weightings[i];
  }
  return score;
}
```

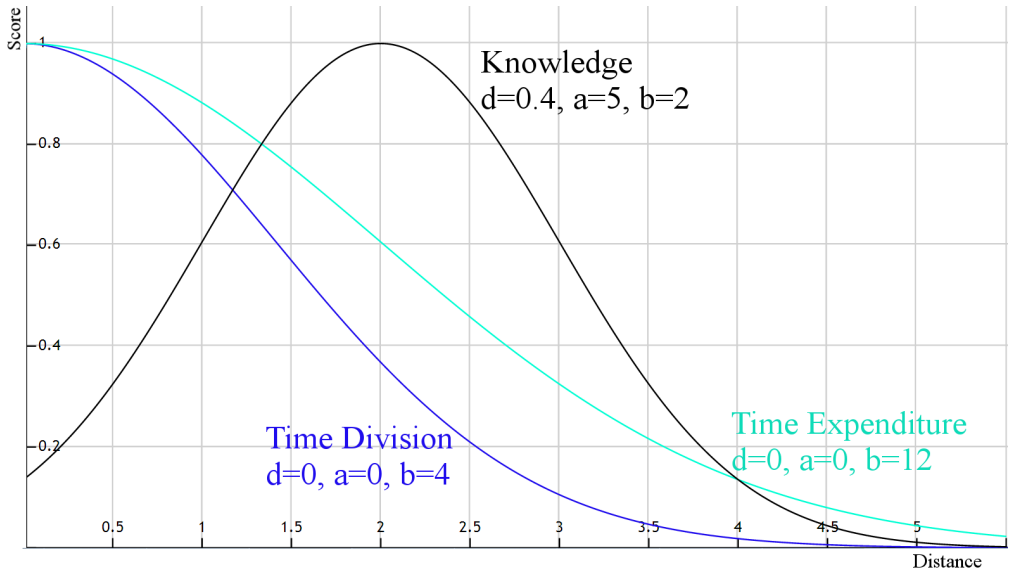


Figure 5.1.: Scoring behaviour using different parameters.

We can now finally calculate the compatibility scores between an origin and all other peers using the aforementioned auxiliary functions.

```
function CalculateScores
{
  for each peer:
  {
    <Calculate the personality score using euclideanDistance>
    if[Limited Deviation Complementary is active]
    {
      <Calculate the knowledge score using weightedGaussian>
    }
    else
    {
      <Calculate the knowledge score using euclideanDistance>
    }
    <Calculate the time expenditure score using weightedGaussian>
    <Calculate the time division score using weightedGaussian>
    <Add expenditure and division score to preference score>
  }
<Normalise Peer Scores>
  for each peer:
  {
    if[origin does not have the peer in avoidance list]
    {
      <Add avoidance list weight to preference score>
    }
    if[Supplementary approach is required for a category]
    {
      <Invert that category's score>
    }
    <Multiply every category score with it's category weight>
    <Add every category score to the total score>
  }
  <Sort all candidates descending according to their total score>
  <Output candidate suggestion list>
}
```

All scores are normalised: For each category, the highest possible score is calculated and each category score is divided by that highest score. After that, all scores range between 0 and 1. This ensures that they are comparable to each other. Dividing by the highest score possible is necessary in contrast to the highest available score so that a reasonable score can also be calculated when the peer list consists of only a small number of entities (e.g. only one student).

5.4. Demonstration - Sample Cases

Given the following exemplary students, the drafted algorithm is tested whether it delivers intuitive results. The Big Five scores in this demonstration can range from 10 to 50. A high openness score means that the student is very open to new ideas, a low extraversion score means that they are introverted etc. The knowledge scores may range, as introduced in the chapter *Establishing Data Models: Student and Team Attributes*, from 0 to 5. A score of 0 means that the student has absolutely no knowledge about a certain topic at all, while 5 means a deep understanding of the topic: Student 0 is an expert on topic A and topic C, but does not know anything

about topic B. Student 2 has average knowledge about all three topics.

	Student 0	Student 1	Student 2	Student 3
Openness	35	36	22	35
Conscientiousness	48	47	12	12
Extraversion	24	30	24	50
Agreeableness	48	46	35	15
Neuroticism	25	27	50	46
Topic A	5	0	3	5
Topic B	0	5	3	4
Topic C	5	0	3	3
Time Expenditure	6	6	8	8
Time Division	3	6	4	1

In all cases, Student 0 is the origin and the other three students are potential candidates. The gaussian function uses the following parameters: If Complementary Limited Deviation is active, the proximal distance is set to even-distributed 40%. This leads to a parameter b=2, as the proximal distance has to be multiplied with the maximum distance: $0.4 \cdot 5 = 2$. The parameter d was arbitrarily set to 0.4. The following compatibility scores are calculated by the drafted compatiblity score algorithm, where *Personality* is abbreviated by "Per.", *Knowledge* is abbreviated by "Know." and *Preference* is abbreviated by "Pref.". In the "Student" column, the total scores are listed. The highest compatibility score is colored in green, the runner-up score is yellow and the lowest score is colored in red.

Mode		Weightings			Student		
Personality	Knowledge	Per.	Know.	Pref.	1	2	3
Not Weighted	Complementary	0.0	1.0	0.0	0.938	0.541	0.516
Not Weighted	Supplementary	0.0	1.0	0.0	0.061	0.458	0.483
Not Weighted	Limited Dev	0.0	1.0	0.0	0.051	0.730	0.419
Complementary	Not Weighted	1.0	0.0	0.0	0.075	0.531	0.661
Supplementary	Not Weighted	1.0	0.0	0.0	0.924	0.468	0.338
Not Weighted	Not Weighted	0.0	0.0	1.0	0.642	0.741	0.577
Complementary	Complementary	0.3	0.3	0.4	0.561	0.618	0.584
Supplementary	Supplementary	0.3	0.3	0.4	0.552	0.574	0.477
Complementary	Supplementary	0.3	0.3	0.4	0.298	0.593	0.574
Complementary	Limited Dev	0.3	0.3	0.4	0.295	0.675	0.555

As can be seen, the scores intuitively represent the desired parameters. Different parameters lead to entirely different suggestions, which can be seen when only the knowledge category is weighted: Each mode leads to a different top suggestion.

5.5. Best Average Team - Iterative Grouping vs. Constraint Solving

When assisting students, the tool only needs to compare a pool of candidates to the specific student that is requesting a suggestion. When assisting lecturers by suggesting whole teams given a pool of students, a combinatorial problem arises: If there are n students, there are n! possible team configurations. An exhaustive approach would have to try out every single team configuration in order to find the team setup that yields the best teams given the specified formation parameters. That is only possible for a small pool size; for greater pool sizes, calculation time would exceed reasonable limits. Two different approaches are presented in this draft that partitionate a pool of candidates into whole teams.
Iterative Grouping is a proposed algorithm that is very easy to implement. It starts by picking i random students $x_1 ... x_i$, with $i = NumberOfStudents/TeamSize_{min}$. They are removed from the pool of students. Compatibility Scores are then calculated between x_1 and all other students. The most compatible student is grouped with x_1 and removed from the pool. After that, x_2 is processed the same way and so forth until x_i is reached, which is processed twice. The algorithm then continues to process x_{i-1}, this time going backwards. The algorithm stops when the pool of students is empty and all students are assigned to a team. The problem with this approach is that it produces a local optimum but not a global one. Also, x_1's team is slightly "better" than x_i's team, because the pool of candidates is greatest when the algorithm starts. This problem is relieved a little by reversing the processing order when the x_i and x_1 are reached, e.g. in the first run, x_i's first selection is within a pool of size s, and the second selection within a pool size of s-1, whereas x_1 may first pick from the complete pool, but the second pick is from within a pool of $2i-1$ lesser size.

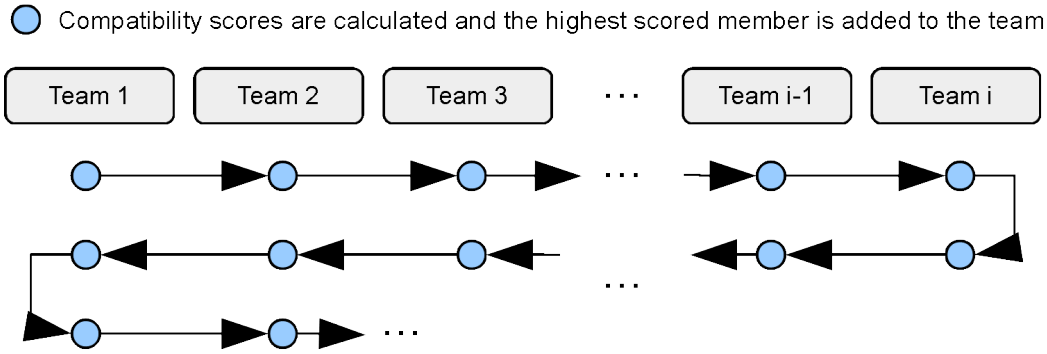


Figure 5.2.: The iterative grouping procedure that adds students until the desired team size is reached.

This algorithm is very fast, even for a large student pool. Multiple usage on the same pool of students with the same parameters yields different results every time because the initial students are picked at random. The local optimum problem in contrast to a global optimum seems acceptable - team formation, after all, is a very

"soft" area: A team probably does not perform noticeably better because it is a few percent more, for example, homogenous than another team: It is to be expected that errors made when self-assessing play a much larger role.
In case a global optimum is desired, an external constraint solver could be used. For this, the available student data, team formation parameters and distance measures used by the Compatibility Score algorithms could be parsed into a MiniZinc[NSB+07] file. MiniZinc is a comprehensive constraint language also used to express combinatorial optimization problems supported by many constraint solvers. Every time students are united, their respective compatibility score is added to a variable *overall compatibility score*. The constraint solver can then try to maximize the overall compatibility score by determining the ideal team configuration and thus the global optimum. However, the problem of assigning students to teams in a manner that maximizes the overall compatibility score is a variant of the multiple knapsack problem [MT90] which is *NP-complete*. If $P \neq NP$ holds, it would be extremely time-consuming to maximize the overall compatibility scores and thus find a global optimum.
Using a constraint optimization-based approach indeed yields a significant potential. However, constraint programming is a complex topic that exceeds the scope of this work. It is thus left to future work to elaborate on time consumption, additional required constraints and predicates and how the available data and distance measures need to be automatically modeled. The question also remains whether the global optimum is really required or if *Iterative Grouping* is sufficient. The advantage of sticking to the latter is that no external tool is required, and modifications can easily be implemented due to it's simplicity.

CHAPTER 6

Increasing The Environment's Attractiveness and Liveliness

So far, the drafted tool does suggest team candidates given user-defined parameters. However, it is important to motivate users to keep using the tool even after the charm of novelty has worn off. A platform is desired that is characterized by liveliness and active exchange - not only is it the goal to improve team compositions, but also to provide an environment where personal feedback can be given so that a student can review their behavior in the group, and possibly mature into a responsible, social human being.

6.1. Personal And Team Feedback

Personal feedback is important. It is a reward to those who successfully actively participate within a group and are social and empathic towards peers. Feedback can be an incentive to aim for that: Accumulating negative feedback will have an effect on the likeliness of being accepted into a team, whereas very positive feedback may allow team access even when someone else would be more "compatible" in respect to *Personality*, *Knowledge* and *Preferences* given the formation parameters. It is also an option to extend the *Best Fit* approach so that good feedback is taken into consideration. However, in this proposed team formation platform, negative feedback is omitted as it yields dangers of abuse: ill-affected students might provide or bullies might provide negative feedback just for various unjust reasons. Instead, a positive reinforcement approach is used: Students can only provide good feedback and highlight especially productive, social or creative team members.
In order for a feedback system to work, as in, provide incentive to not behave badly, it must be costly to receive "negative" feedback. As actual negative feedback was omitted, "negative feedback" equals to "having been in many teams without getting any positive feedback". It does not serve much purpose if creating a new account, thus wiping all team history, is easy to do. Accounts have to be uniquely linked to a student - this could be done by requesting the student number as a user account

identification. Changing the enrollment number or student email address provided by faculty takes reasonably enough effort. An identifying email address (often provided by the university at enrollment) is sufficient as well and should be used if possible, as it is a less personal attribute than the student number, which might be abused by other users.
Students should not only be able to give personal feedback, but also feedback about the whole team by giving 1-5 stars as answers to some questions, as well as a field for additional remarks. The questions should be asked in a way so that rewarding many stars is a positive thing. Such a feedback is valuable because every team member can see how their peers felt within the team. It is also interesting for lecturers, who can adapt their formation parameters if there were problems due to a "misconfigured" team setup. The following are sample questions that consider productivity as well as personal well-being within a team:

- Were you able to contribute to the team's success?
- Did you feel included in the team?
- Was it easy for your team to work together?
- Do you feel like you gained something from working in this team?
- Did you feel at ease working with this team?

Every time a team is disbanded or terminates, a student leaves a team or is expelled, or after a certain time period has passed, all involved students are asked questions like this. This allows to evaluate how the students were feeling as time passes and avoids potentially biased single ratings. Providing feedback should always be on a voluntary basis. Paying attention to the students' feedback and talking about it is important. When students feel like their feedback is taken seriously, future team formations as well as the tool's acceptance will benefit.
Public feedback is only indirectly be calculated using the team feedback of the teams an individual has participated in. The idea is that students that contribute to a team and try to empathize with their teammates will more often pave the way to a successful team, which will receive positive feedback from all of it's members. This reflects on each member, but a student with very good team skills will more often lead to a good team, and thus end up being in teams that are rated well more frequently.

6.2. Inspirations From Games

Most games nowadays are doing a great job at keeping the players motivated: badges and achievements for special, unusual or effortful accomplishments and ranks or leveling mechanics that represent the amount of total effort or success. It is fun to hunt for all achievements, compete with friends or simply show off by leveling up and getting a high rank even in adult age: Thom et al. removed gamification from an enterprise social network and observed that overall participation via contribution decreased [TMD12]. Thus, user levels could be introduced: A user

starts at Level 1, and they level up when they have earned enough points, e.g. experience points. Experience points could be awarded for joining or creating a team, taking part in team votes, and providing and receiving feedback. The amount of experience points awarded could be relative to the "goodness" of a received feedback, so that receiving a good feedback gives much more experience points than any other action. Experience points or special badges could also be rewarded for fulfilling achievements such as "was mentioned as a good team member x times", "has given feedback to x students" or "having taken part in x team votings", or to express special tendencies such as "preferring personality-wise homogenous teams". High level students could possibly be rewarded by the lecturer, for example, at the end of the semester.
It is also thinkable to introduce "negative badges", e.g. for quitting teams often. Negative badges could be "removed" by spending a certain amount of experience points, making it possible to compensate "bad behaviour" by other positive actions.

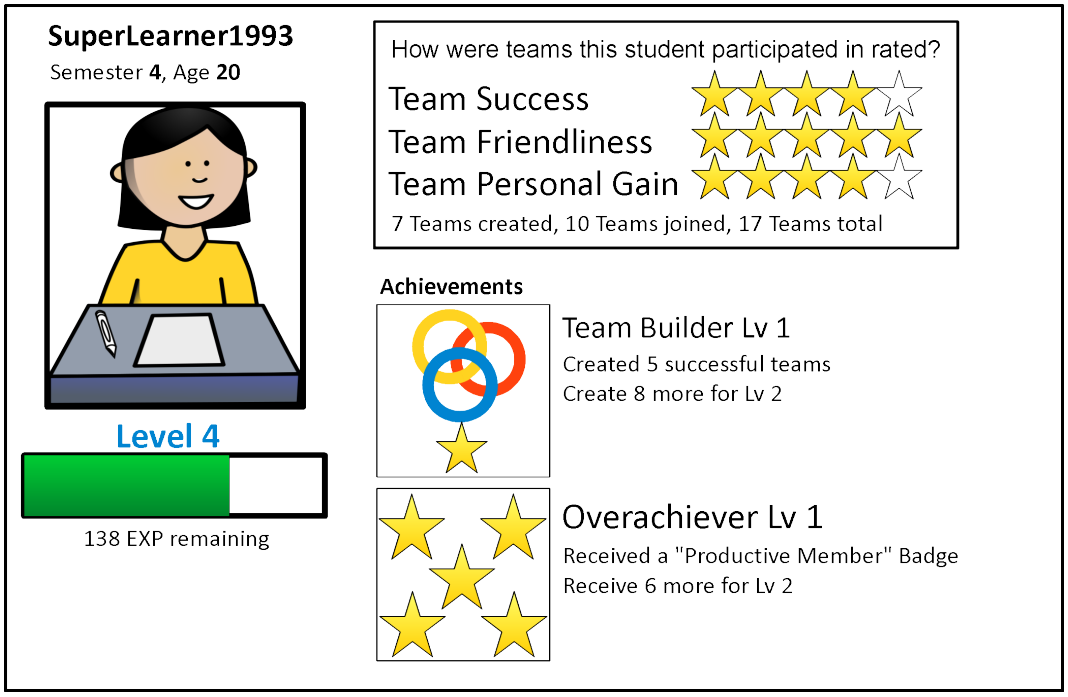


Figure 6.1.: A sample profile page draft showing gamification aspects such as badges and levels. Avatar picture taken from openclipart.org.

CHAPTER 7

Extending Versatility Of The Team Formation Platform

Some extensions to the tool are discussed in this chapter in order to make the tool useable for a wide variety of users and requirements. As it stands, the tool can only handle a single course. With more courses, the lecture-based group feature would not work as intended. Gamification aspects have been introduced, but there is no place to show them. Thus, some additions have to be made to increase the suitability of the tool as a social team formation platform. Also, the conclusion of the most beneficial team formation parameters might be interesting to researchers, so some features for this specific target audience are also elaborated in this chapter. Last but not least, the increased popularity of Massive Open Online Courses introduces some special challenges that need some extensions to the platform.

7.1. Social Media Features

Gamification aspects have been introduced, however, there is no place to show them off unless a student appears as a suggestion, in which case their profile can be viewed. Also, "competition" between friends on the actual formation platform is not yet possible: the team formation algorithm does not implement the concept of friendship. The tool should thus be extended to make it into a more personal platform, a place where students can present themselves so that they can express what kind of person they are besides the numeric values that each formation category stores. Every student has a profile page where they can list their interests, what they are working on and what they like. Although not considered in the actual team formation, befriending other students on the platform should be possible. Similar to Facebook's "wall", it is thinkable to have a page where activity of friends is presented. A simple message functionality for easier team communication and exchange between friends could be integrated to round it off.

7.2. Communities

Without any extension of the described concept of semi-automatic group formation, it would not be reasonable to employ one instance of the tool for more than one course: The lecture-based group feature relies on the prerequisite that every student is attending the same course and one lecturer sets the formation parameters. Also, if several courses used the same tool environment, students from different courses would have to filter team invitations and suggestions, removing all those students or groups which they are not interested in. The clarity of available teams and potential team members would suffer soon. Therefore, communities are introduced: A community represents a course or general field of interest. Groups are created within a community, which are isolated from each other.

Example 7.2.1. A student A joins community "Analysis", whereas student B joins community "Student Software Projects". When A is forming groups, they can not invite B, and B is not suggested to A, unless B joins community X as well.

This allows students to join only their courses' communities and other communities they are explicitly interested in. Multiple courses can use one platform, with multiple lecturers controlling their own community and setting their own formation parameters within that community.

7.3. Logging Capability For Research

Due to the versatility of the team formation tool on account of highly adjustable formation parameters, the tool could prove useful to further research regarding the best team formation parameters depending on the purpose of the team. To support researchers, the tool could log formation parameters and team members and their data along with the date, type and result of team actions. If increased anonymity is desired, real student names could automatically be replaced by generated aliases. With the help of these log files, it might be easier to find correlations between formation parameters and team performance or social harmony within a team. It is also thinkable to automatize the process of evaluating the log files: In everyday use at a large university, with many courses with a lot of students, a lot of data is bound to accumulate, which could then be utilized to improve default formation parameters.

7.4. Massive Open Online Course

With MOOCs (Massive Open Online Courses) on the rise, it is to be expected that teams will be formed that rely solely on digital channels such as email or (voice) chat in order to handle their communication. In order to make the team formation tool as flexible as possible, there could be a distinction whether the teams are operating *online* or *offline*. While the latter will not prevent the team members from using digital features, an important assumption can be made about a team flagged as an offline team: The team members will most likely share the same time zone,

so real-time media such as an online conference will always be an option. Online teams, on the other hand, might be comprised of individuals from all around the world. In order to allow an estimation of how feasible communication will be, applicants for an online team should be prompted to enter their time zones and preferred communication channels, as to allow lecturers or team leaders to consider this barrier when choosing team applicants. When operating in automatic mode, the team formation tool could lower the overall matching score if time zones differ too much.

CHAPTER 8

Prototype

A prototype was implemented that is able to assist lecturers in forming teams by generating complete team setups. Most of the features described in this work were omitted due to time-conditioned reasons. However, the prototype still implements the most important features of the team formation platform; it is able to calculate a compatibility score and use these scores to set up whole teams. The prototype makes use of a server-client architecture and consists of two pieces of software. The graphical client is written in Java and thus runs on most operating systems. All actions done in the client are sent to the server. The client allows users to register and manage their accounts. They may provide their name, age, semester count, gender and nationality and can take a Big Five test which is instantly evaluated; showing the users their Big Five scores right after they have completed the test. The prototype distinguishes two types of accounts: student accounts and lecturer accounts. Lecturer accounts may set up team announcements by providing the course name, a description and the required knowledge areas along with their weight, which represents the importance of the corresponding knowledge area. After the team announcement has been submitted, it shows up in the team announcement listing which is available to student accounts. Upon applying for a team, students have to self-assess their knowledge within the specified knowledge areas. Students also have to provide their desired time expenditure in total and desired time expenditure per session. At any time, a lecturer account may look at the applications for their team announcements. Lecturers can inspect a student's profile and the time expenditure and knowledge values they have submitted. Lecturers are also presented with the number of total applications.
When lecturers want to process the applications into a team setup, they may adjust the category weightings and the formation mode per category, e.g. whether the personality score should be calculated using a complementary approach (heterogeneous personalities receive higher scores) or supplementary (homogenous personalities receive higher scores). Lectures also need to specify the team size that should be aimed for. It should be noted that in some cases, not all teams can have

the same number of team members. For a desired team size of 5 and 16 applicants, there will be one team that has 6 members. After the team setup has been configured, the lecturer can request a team setup suggestion. If needed, the parameters can be re-adjusted and a new suggestion can be requested until the lecturer is satisfied with the setup. If that is the case, the suggested setup is saved and the team announcement is removed.
The server is written in C# using the Mono framework, which allows deployment on most available operating systems in contrary to the default .NET framework, which only runs on Windows. It makes use of asynchronous networking, thus multiple clients can connect to the server at once. The server receives the client's actions, processes them if the client has authenticated itself successfully and returns a status of the transaction to the client. In case of a profile update, the returned status just tells the client that the update was successful. In case of a team setup request, the server returns the suggested team setup. The server is also capable of returning error codes if something went wrong, e.g. if the client is not authenticated or the protocol used is unknown to the server. The server persistently saves users, team announcements, student applications and team setups to the hard disk each time an update occurs. In the worst case, if the server crashes while it writes the data to the disk, only the last update is lost.
The team formation algorithms implemented in the server are reasonably fast. A test conducted on an Intel Core i5 3317U, which is as of 2014 a mid range mobile CPU, yielded that it only takes about one second to compile 500 students into teams of 6. It should be noted that during this test, the server outputted all operations to the console. It was observed that C# console applications are slowed down when a lot of console output is produced, so it is likely that the server's team formation is a lot faster when the server runs in a quiet mode.
The source code and binaries can be found under the following address:

`http://141.84.94.83:56781/alex/ba_marco_hoffmann`

Please note that this link may only be reachable within the LMU/LRZ virtual private network.

CHAPTER 9

Outlook and Future Work

The tool drafted in this work is capable of forming teams based on predefined formation parameters, composing teams using personality, knowledge and preference-related data. It also includes some basic functionality for peer exchange and social networking, and tries to motivate users to keep using it by implementing gamification aspects. However, there are still a lot of remaining questions. For one, the ideal default formation parameters given a desired team purpose need to be elaborated, especially the most appropriate category weighting. Testing actual teams formed by this tool were not looked into yet - it needs to be verified whether the used distance measures are appropriate and yield the desired results, although the scores calculated by the tool do intuitively look correct. Also, the potential from using a constraint-based solution needs to be evaluated in respect of time consumption, benefit of finding a global optimum and increased versability by adding customizable constraints that can be used for very custom team formation. The implemented prototype does provide a show case of the student-assisting algorithm, but the whole platform including all described features remains to be implemented. To make semi-automatic team formation available to many students and simplify its use by lecturers, it is reasonable to embed the team formation outlined in this thesis into a web application. This work broadly discusses many aspects of team formation and the surrounding environment and provides a first step towards a platform that works together with its' users in order to form more efficient, more creative and more harmonic teams.

APPENDIX A

Prototype Manual

A.1. Starting The Server

Before the GUI client can be used, a server needs to be started. It is a console application that takes up to four command-line arguments.

```
TFTProtoServ.exe [port] [ran_stud] [lec_name] [lec_pw]
```

It is important to note that the order of the arguments is important and no preceding argument may be omitted. That means that the following startup variants are allowed:

```
TFTProtoServ.exe

TFTProtoServ.exe [port]

TFTProtoServ.exe [port] [ran_stud]

TFTProtoServ.exe [port] [ran_stud] [lec_name] [lec_pw]
```

The first argument [port] overrides the default port if specified. The default port is 20405. If one wishes to use the subsequent command-line arguments while leaving the port as default, 0 can be entered. The second argument [ran_stud] (random students) causes the server to generate a number of random sample students that will apply to team announcements automatically. 0 generates zero random students and can thus be entered if subsequent command-line arguments need to be used. The third argument [lec_name] (lecturer name) registers a new lecturer account. Lecturer accounts can (in contrary to student accounts) create team announcements and form teams. If the third argument is specified, the fourth argument [lec_pw] (the account's password) must be specified, too.
After start-up, the server outputs the address and port clients will have to connect to. The server can handle up to 500 simultaneous connections. During operation, the server creates a number of files:

- `users.db` stores all lecturers, students and automatically generated students.
- `anc.db` stores all team announcements.
- `app.db` stores all submitted applications.
- `teams.db` stores all created teams.

If one wishes to wipe all users, team announcements, applications or created teams, simply delete the corresponding file. The server also creates backup files using the file extension `.bak`. These files are auxiliary files that prevent data loss caused by a potential crash during the database writing stage and can be ignored by the operator.

A.2. Operating The Client

The client has a graphical user interface that welcomes the user with four tabs: "My Profile", "Team Listing", "Create Team" and "Form Teams". To students, the first two are available, for lecturers, all of them are available. On the first tab, "My Profile", a connection needs to be set up before anything. A user name and password can be entered along with the server adress and port. After all fields have been filled in, press the "Login/Register" button. The client will try to log in using the specified user name and password. If the user name does not exist in the server, a new user with that user name and password will be created and automatically be logged in. After a successful login, the user can update their personal information, add other usernames to their blacklist and take a Big Five personality test. The implemented Big Five test is a 50-item questionnaire which procedures are explained within the software. After every update, the user needs to hit the "Save Changes" button. The updated information is then permanently stored in the server.
The second tab, "Team Listing", provides an overview of current open team announcements. It informs the user about the course, the announcement's description, the number of applications that have been submitted already as well as a status, which reads "Open" if it is still possible to apply or "Requested" if an application has already been submitted. Before any teams show up on the list, the "Update" button needs to be pressed. The user can apply by selecting a team application in the list and clicking "Apply", after which they will have to self-assess themselves about the specified knowledge areas. They also need to provide information about their time expenditure in total and per team session. For lecturers, the "Remove Listing" button is available which deleted the highlighted team announcement permanently along with all student applications that were submitted for this announcement.
The third tab, "Create Team", provides an interface for lecturers to create new team announcements. A course name and description must be entered here. Knowledge Areas can be added by entering the name of the knowledge area and selecting a desired weight for this area. A higher number means that this knowledge area is more important and will have a stronger influence on the compatibility score and thus the team suggestion. After a knowledge area has been specified and the

desired weight is adjusted, press the "Add"-button to add it to the list of knowledge areas. Knowledge areas can be removed; highlight a knowledge area in the list and press the "Remove" button. After the team announcement has been set up, press the "Create" button. The team announcement will be stored in the server. Randomly generated students will apply automatically. The team announcement will now appear in the "Team Listing" tab.
The fourth tab, "Form Teams", is the place to request team setups. First, select a team announcement from the "Team Announcement" list and hit the "Inspect" button. The second list will be filled with those students who submitted an application. Selecting a student and pressing "Details" will open a new window that shows the profile and self-assessment scores of the student. Pressing "Reject Application" will delete the student's application permanently. This can not be undone; the student will have to apply again. After the weights and formation parameters have been configured to the lecturer's liking, press the "Suggest Teams" button. A list of teams will appear in the "Teams" list. Selecting a team will show the members of the teams. Since the team formation algorithm is not deterministic, hitting "Suggest Teams" again may yield a different team setup. Pressing the "Announce Teams" button will inform the server that the current team setup is to be saved permanently. This will also delete the team announcement.

Bibliography

[Bek05] BEKELE, Rahel: *Computer-Assisted Learner Group Formation Based on Personality Traits*, Universität Hamburg, Diss., 2005

[BH97] BRADLEY, John H. ; HEBERT, Frederic J.: The effect of personality type on team performance. In: *Journal of Management Development* 16 (1997), S. 337–353

[CM85] COSTA, Paul T. ; MCCRAE, Robert R.: *The NEO personality inventory: Manual, form S and form R*. Psychological Assessment Resources, 1985

[CMTW06] CHAPMAN, Kenneth J. ; MEUTER, Matthew ; TOY, Dan ; WRIGHT, Lauren: Can't We Pick our Own Groups? The Influence of Group Selection Method on Group Dynamics and Outcomes. In: *Journal of Management Education* 30 (2006), S. 557–569

[FWG03] FITZGERALD, James T. ; WHITE, Casey B. ; GRUPPEN, Larry D.: A longitudinal study of self-assessment accuracy. In: *Medical Education* 37 (2003), S. 645–649

[HC07] *Kapitel* Studying the Impact of Personality and Group Formation on Learner Performance. In: HÓRREO, Víctor S. ; CARRO, Rosa M.: *Group: Design, Implementation, and Use*. Springer Berlin Heidelberg, 2007, S. 287–294

[JS99] *Kapitel* 4. In: JOHN, Oliver P. ; SRIVASTAVA, Sanjay: *The Big Five Trait Taxonomy: History, Measurement, and Theoretical Perspectives*. Guildford Press, 1999, S. 102–138

[MN05] MANNIX, Elizabeth ; NEALE, Margaret A.: What Differences Make a Difference? - The Promise and Reality of Diverse Teams in Organizations. In: *Psychological Science in the Public Interest* 6 (2005), S. 31–55

[MT90] MARTELLO, Silvano ; TOTH, Paolo: *Knapsack Problems*. John Wiley & Sons, 1990

[NSB+07] NETHERCOTE, Nicholas ; STUCKEY, Peter J. ; BECKET, Ralph ; BRAND, Sebastian ; DUCK, Gegrory J. ; TACK, Guido: MiniZinc: Towards a

Standard CP Modelling Language. In: *Principles and Practice of Constraint Programming* 4741 (2007), S. 529–543

[NWC99] NEUMAN, George A. ; WAGNER, Stephen H. ; CHRISTIANSEN, Neil D.: The Relationship between Work-Team Personality Composition and the Job Performance of Teams. In: *Group Organization Management* 24 (1999), S. 28–45

[OFBE04] OAKLEY, Barbara ; FELDER, Richard M. ; BRENT, Rebecca ; ELHAJJ, Imad: Turning Student Groups into Effective Teams. In: *Journal of Student Centered Learning* 2 (2004), S. 9–34

[PGBB11] POHL, Alexander ; GEHLEN-BAUM, Vera ; BRY, François: Introducing Backstage - a digital backchannel for larger class lectures. In: *Emerald Journal: Interactive Technology and Smart Education* 8 (2011)

[RFRT01] ROBINS, Richard W. ; FRALEY, R. C. ; ROBERTS, Brent W. ; TRZESNIEWSKI, Kali H.: A Longitudinal Study of Personality Change in Young Adulthood. In: *Journal of Personality* 69 (2001)

[Sco60] SCOFIELD, Robert W.: Task Productivity Of Groups Of Friends And Non-Friends. In: *Psychological Reports* 6 (1960), S. 459–460

[SRS14] SPOELSTRA, Howard ; ROSMALEN, Peter van ; SLOEP, Peter: Toward Project-based Learning and Team Formation in Open Learning Environments. In: *j-jucs* 20 (2014), S. 57–76

[SRV+13] SPOELSTRA, Howard ; ROSMALEN, Peter van ; VRIE, Evert van d. ; OBREZA, Matija ; SLOEP, Peter: A Team Formation and Project-based Learning Support Service for Social Learning Networks. In: *Journal of Universal Computer Science* 19 (2013), S. 1474–1495

[TMD12] THOM, Jennifer ; MILLEN, David ; DIMICCO, Joan: Removing gamification from an enterprise SNS. In: *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work* (2012), S. 1067–1070

[Vyg78] VYGOTSKY, Lev S. ; COLE, Michael (Hrsg.): *Mind in Society: Development of Higher Psychological Processes*. Harvard University, 1978

[Wes97] WEST, Michael A.: *Developing Creativity in Organizations: Personal and Professional Development*. Wiley-Blackwell, 1997

[WJKC98] WATSON, Warren E. ; JOHNSON, Lynn ; KUMAR, Kamalesh ; CRITELLI, Joe: Process gain and process loss. In: *International Journal of Intercultural Relations* 22 (1998), S. 409–430

[WKM93] WATSON, Warren E. ; KUMAR, Kamalesh ; MICHAELSEN, Larry K.: Cultural Diversity's Impact on Interaction Process and Performance: Comparing Homogeneous and Diverse Task Groups. In: *The Academy of Management Journal* 36 (1993)