CARLETON UNIVERSITY

(Carleton University Project Partner Identifier)

Algorithm Design Document

Team Insomnia

Dabeluchi Ndubisi
Charlie Li
Pierre Seguin

Submitted to:
Dr. Christine Laurendeau
COMP 3004 Object-Oriented Software Engineering
School of Computer Science
Carleton University

Contents

Tables

Figures

Introduction

This design document is written to describe, in details, the modus operandi of the cuPID matching algorithm. The procedures described here will allow the algorithm to create optimal teams for the project in question. We have decided to characterize the optimality of the algorithm on a per individual basis as opposed to a per team basis. It should be noted that this has no effect on the efficiency of the algorithm, but helps characterize the correctness of the algorithm. This algorithm examines and computes each StudentUser's Project Partner Project (*PPP*) in detail and formulates its findings into a team match report. The details of the algorithm that this document include:

1. Characterization of each personal and teammate qualifications from a StudentUser's PPP
2. Specification of the details of and provision of an analysis on why the qualifications chosen will complement the algorithm
3. Definition of the set of unique, deterministic algorithmic rules to achieve a balanced result from input PPPs
4. Description of the details in which the algorithm computes the final result in respect to each project configurations using the rules and illustrating edge cases

Qualifications

Table 1 - List of qualifications

Tracebility	Ranges / Options	Qualification
		Work Ethic and Personality Qualifications
		Pick 5 out of 8 traits per StudentUser
Q-01	N/A	Dependable
Q-02	N/A	Organized
Q-03	N/A	Proactive
Q-04	N/A	Efficient
Q-05	N/A	Sense of humor
Q-06	N/A	Impulsive
Q-07	N/A	Flexible
Q-08	N/A	Hardworking
		Technical Qualifications
Q-09	0 to 12	CGPA
Q-10	0 to 10	Object oriented programming skills
Q-11	0 to 10	User interface and user experience skills
Q-12	0 to 10	Scripting skills (any language)
Q-13	0 to 10	Database design skills
Q-14	0 to 10	System modeling, patterns and architecture design skills
Q-15	0 to 10	Familiarity with data structures and algorithms
Q-16	0 to 10	Computer Security experience
Q-17	0 to 10	Software documentation experience
Q-18	0 to 10	Computer Networking experience
Q-19	0 to 10	Version Control experience
Q-20	0 to 10	Web development skills

Qualification Details

Work Ethic Qualifications

The essence of the work ethic qualification section is to identify the specific work traits that each StudentUser can bring to a project. Generally, Based on psychological research it has been established that in order for a particular group task to prosper, the group of people responsible for the task need to have similar work ethic traits. Consequently, the cuPID algorithm follows suit. Care has been taken to ensure that StudentUsers with similar work ethic will end up together in a team.

In this section, 8 traits (qualifications) are provided to each StudentUser, and they are meant to select 5 out of the 8 traits that best qualify them. These 5 traits selected by the

StudentUser will be the collection of his/her work ethic traits that he/she has chosen to identify himself/herself with. The explanations of each work ethic qualifications are given below:

(Q-1) Dependable

The dependable qualification is a contributing factor to determine the work ethic of a StudentUser. This qualification helps to resolve students who have identified themselves to be able to commit to obligations. StudentUsers who are dependable are likely to fulfil commitment they have made to the project. The objective of this trait is to classify members of each group based on their ability to pull their pull through their commitments they make for the project

(Q-2) Organized

The organized qualification is a contributing factor to determine the work ethic of a StudentUser. This qualification helps to resolve students who have identified themselves to be organized over group work. Students who have the same organized qualification are likely to have smooth interactions over the group meetings. The objective of this trait is to classify members of each group based on their ability to be ready when called upon and be on time with the workflow schedule.

(Q-3) Proactive

The proactive qualification is a contributing factor to determine the work ethic of a StudentUser. This qualification helps identify StudentUsers who are always on top of a matter involved with the project. StudentUsers with this trait tend to take on actions that were not necessarily delegated to them in order to benefit the project.

(Q-4) Efficient

The efficient qualification is a contributing factor to determine the work ethic of a StudentUser. This qualification helps to resolve students who have identified themselves to be generally handle a particular task without the expense of either time or resources. StudentUsers who are efficient can generally finish work quickly without the need to meet for long times. The objective of this trait is to classify members of each group based on their efficiencies during a project.

(Q-5) Sense of humor

The sense of humor qualification is a contributing factor to determine the work ethic of a StudentUser. This qualification helps resolve students who have identified themselves to have a sense of amusement and can generally change disappointments into amusing moments in order to uplift the team. The objective of this trait is to classify members of each group based on their sense of humor during a project.

(Q-6) Impulsive

The Impulsive qualification is a contributing factor to determine the work ethic of a StudentUser. This qualification helps resolve StudentUsers who have identified themselves to make decisions without forethought. The objective of this trait is to classify members of each group based on their impulse during a project.

(Q-7) Flexible

The Flexible qualification is a contributing factor to determine the work ethic of a StudentUser. This qualification helps resolve StudentUsers who have identified themselves to be able to change easily in order to adapt to situations that may arise during the course of a project. The objective of this trait is to classify members of each group based on their flexibility during a project

(Q-8) Hardworking

The hardworking qualification is a contributing factor to determine the work ethic of a StudentUser. This qualification helps to resolve StudentUsers who have identified themselves to have a knack for dedicating hard work into a project in order to make it prosper. The objective of this trait is to classify members of each group based on how hard they are able to work during a project.

Technical Qualifications

The essence of the Technical qualifications section is to identify the technical skills that each StudentUser can bring to a particular team. Generally, it is good to put StudentUsers with diverse skill sets into a particular project. This will ensure that each person will have a unique skill that he/she provides to the team. Also, we make a valid assumption that StudentUsers technical skills will be able to learn new skills in order to help the team.

The technical qualifications section is set up such that each qualification ranges from 0 to 10, hence providing a tangible value to the competence of a particular StudentUser with respect to a particular technical skill. The CGPA of the StudentUser is also needed and it ranges from 0 to 12.0.

Finally, a simple coding question will be assigned to the StudentUser in order to ascertain his/her actual technical level. This coding question will be treated as (Q*) because it is not fully representative as a qualification, but it is more of a confirmatory exercise for the technical skills provide by the StudentUser. The explanations of each technical qualifications are given below:

(Q-9) CGPA (Cumulative Grade Point Average)

The CGPA qualification helps to give a general performance indicator of the StudentUser's technical skills. The valid ranges of this qualification is from 0 to 12.0.

(Q-10) Object Oriented Programming Skills

The Object Oriented Programming Skills qualification helps indicate the StudentUser's competence with the concept of Object Oriented Programming. The valid ranges of this qualification is from 0 to 10.

(Q-11) User Interface and User Experience Skills

The User Interface and User Experience qualification helps indicate the StudentUser's competence with the concept of creating good User Interfaces and User Experience for client based Softwares. The valid range of this qualification is from 0 to 10.

(Q-12) Scripting Skills (any language)

The Scripting skills qualification helps indicate the StudentUser's competence with the concept of creating scripts that can help automate activities that will be related to creating a project. The valid range of this qualification is from 0 to 10.

(Q-13) Database Design Skills

The Database design qualification helps indicate the StudentUser's competence with the concept of creating scalable and efficient data models for storing persistent data related to a project. The valid range of this qualification is from 0 to 10.

(Q-14) System Modelling Patterns and Architecture Design Skills

The System modelling patterns and architecture design qualification helps indicate the StudentUser's competence with the concept of building scalable and extensible system models and architectures for a particular project. The valid range of this qualification is from 0 to 10.

(Q-15) Familiarity With Data Structures and Algorithms

The Familiarity with Data Structures and Algorithms qualification helps indicate the StudentUser's competence with the concept of solving problems in order to maximize efficiency and minimize space used up by data structures. The valid range of this qualification is from 0 to 10.

(Q-16) Computer Security Experience

The computer Security qualification helps indicate the StudentUser's awareness of security flaws of computer systems today and contribute their experiences to build a safer system by following the best practices approach. The valid range of this qualification is from 0 to 10.

(Q-17) Software Documentation Experience

The Software Documentation Experience qualification helps indicate the StudentUser's competence in composing professional grade documentation outlines and the ability to produce system models and analysis. The valid range of this qualification is from 0 to 10.

(Q-18) Computer Networking Experience

The Computer Networking qualification helps indicate the StudentUser's familiarity with network programming, network transport protocols, and network algorithms. The valid range of this qualification is from 0 to 10.

(Q-19) Version Control Experience

The Version Control Experience qualification helps indicate the StudentUser's familiarity with version control software like git, which will prove necessary in typical group coding projects. The valid range of this qualification is from 0 to 10.

(Q-20) Web Development Skills

The Web Development Skills qualification helps indicate the StudentUser's competence in web languages, technologies, and development. The users should understand how to build a web application from end to end (blank page to hosting environment configurations). The valid range of this qualification is from 0 to 10.

Algorithm Rules

Goal

The aim of this algorithm is to create the **most compatible**, cohesive team from StudentUsers who are registered with respect to the project that the algorithm is invoked on. A compatible team is characterised by a group of StudentUsers who bear a mix of complementary personality traits, similar technical skill-set, and display similar work ethics.

Rules

This algorithm uses two major qualifications sections (*Technical* and *Work Ethic and Personality*) to define a compatible matching between the users who are registered in a project. When a StudentUser registered a Project Partner Profile, they must complete the registration form by entering all their personal qualifications and teammate qualifications. Personal qualifications are defined to be a self-evaluation of personal technical skills. Teammate qualifications justify the desired technical skills of teammate who register under the same project.

Work Ethic and Personality

The StudentUser must specify (choose) five out of the eight work ethic and personality traits that best defines themselves. This category is used by the algorithm to find teammates that selected similar traits as themselves. It is assumed that students who select the same personality and work ethic traits, will be most compatible with each other. The system representation of the student's choices will be realized in bit string format. Each work ethic and personality trait that the user selected will map to a specific index of a bit in one byte. The use and application of this bit string is further described the in **Algorithm Details** section.

The StudentUser will not specify the work ethic and personality traits requested from their project partners because the matching algorithm will take careful considerations to match people with similar traits together. The rule for similarity is defined as two project partners to have at least **three** out of five qualities that they select, match during team creation.

Technical Traits

The StudentUser will provide the cuPID system with an honest self-assessment their own personal technical skills. There are twelve categories of technical skill in which the user will select the range of competence in which the user believes they deserve. Each qualification will contribute a score to the technical aspect of the Project Partner Profile. These will be used by the system to pool together all personal and teammate qualification separately. An aggregate of the personal qualifications will be generated from the competence levels of each. The system will modify this personal aggregate slightly (see next section *Technical Normalization*) to finalize the technical score of this StudentUser.

On a separate note, the StudentUser's teammate desired qualifications will also be used to generate a separate aggregate score to define the desired partner's technical score level.

The personal technical score will serve as a competency identifier for the student. This score will be inputted into the algorithm to determine the fit of a StudentUser when auditing them for a team. These identifiers will be placed to group together students with common skill

level. It is the belief that a group of students who belong to the same competency level can fulfil the *theme of this algorithm* and will work better with each other.

Technical Normalization

Since total discretion is given to the StudentUser to evaluate their own personal competence and their desired teammate competencies, it is possible for the users to overshoot their competencies. This part of the rules is to define a constraint on the final process of generating a technical score and will be used to normalize the self-claimed qualifications in a StudentUser's project partner profile. The system will use a simple test (coding) to verify the abilities of the user. A full mark on the initiation test will confirm their claims for their competencies or else a deduction (the normalization) will be applied to their personal and teammate technical score. It is believed that a user who is capable of acing the simple test would hold worthy of their self-proclaimed qualifications, otherwise, the system will normalize their score based on the outcome. The technical score normalization is hidden from the user and the user will not receive feedback on the outcome of their test.

Algorithm Details

The cuPID matching algorithm is described to be **deterministic**, and **greedy** over its iterations of process. The characteristics of the procedure is bound to return the same result with the same input set and will make the most compatible choice on each step.This design decision was settled after comparing multiple algorithm implementation to fit the model matching model by using: global optimal team based, optimal individual based, global balanced approach, and local optimal team based. The cuPID algorithm is designed with the *local individual optimality based approach* where the algorithm only has a local view of students within a similar technical proficiency and strives to create teams with similar abilities. In summary, students are selected and grouped based on their technical scores and work ethic personality traits for maximal compatibility.

The following section will elaborate further and describe the procedures for the cuPID team matching algorithm in detail. The primary constraint to algorithm launch is that there must exist at least twice the number of students registered in the project as team size configuration predefined by the AdministratorUser. This ensures at least two teams are created from the project matching algorithm. Hence, the following preconditions must be met in order for the algorithm to begin execution on a particular project:

1. The configurations required by the algorithm must have been set for the given project in question.
2. The number of StudentUsers currently registered in the project in question must be at least twice as much as the team size specified by the AdministratorUser.

When the AdministratorUser starts the algorithm for Project A, the following system processes follow:

1. Every StudentUser's final technical scores are computed, retrieved and loaded into tuples.

a. The first element of the tuple represents the StudentUser's personal technical qualification score.
b. The second element of the tuple represents the desired teammate's technical score.

Each StudentUser tuple is ordered by the first element, the personal technical score, in the collection of registered users for Project A. That is, the collection of student users is ordered from the StudentUser with the highest technical score to the lowest. The StudentUser collection will map each possible key, the technical score to the nearest whole number (100, 90, 80, ect…), to store a collection of StudentUsers whose technical scores, when floored, corresponds to the key.

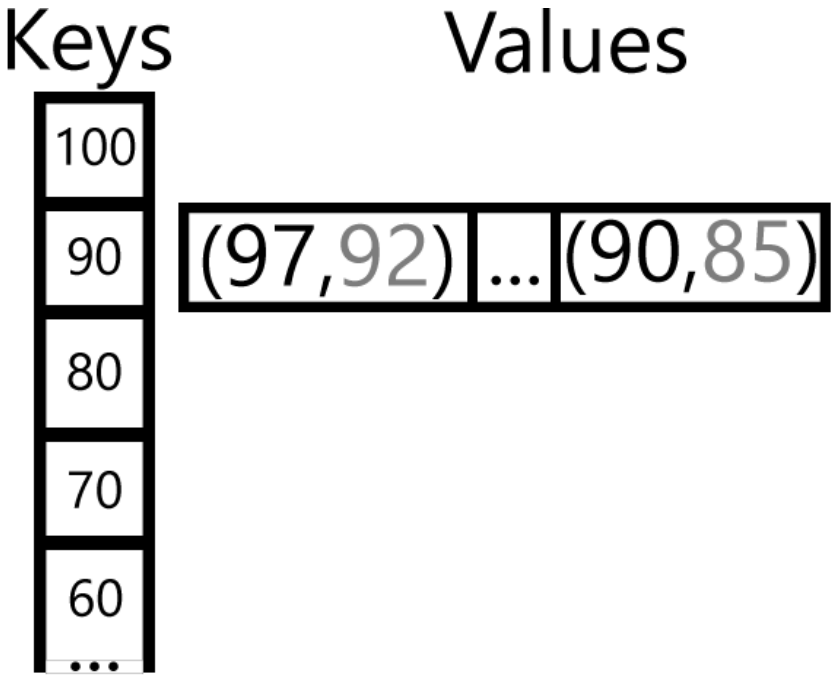

Figure 1 - Model of student collection where each rounded off technical score maps onto a list of StudentUsers. Grey values indicate desired teammate technical score.

2. Take the StudentUser with the highest (personal) technical score out of the student user collection starting from the highest possible key.
    a. The algorithm will determine the technical score bracket they belong to and pop the student with the highest score from the first element of their tuple.
    b. This user's teammate technical qualification will decide the minimum acceptable score of his first group member to be matched with.
3. Suppose the student we have just popped out is Student A. The algorithm will proceed to find and match a student to accommodate Student A's desired teammate technical score.
    a. The search for a complimentary StudentUser B starts from the reverse side (lowest technical score in the bracket). The aim of this is to satisfy Student A's

teammate requirement in the greedy way, thereby not giving a StudentUser much more than he/she asked for.

b. If a complimentary StudentUser B was not found in terms of similar work ethic and acceptable technical skills, then in such case the algorithm will massage the technical requirement of the StudentUser A in order to find a StudentUser within the same bracket that complements StudentUser A in terms of work ethic. (*It should be noted that work ethic must maintain highest priority*)

c. If a complementary StudentUser B cannot be found for the StudentUser A within the same key bracket, then the algorithm moves into a key bracket lower to find someone who matches the requirement of the StudentUser.

4. As of now, we will continue with the assumption that the algorithm has successfully found a complementary StudentUser B with respect to Student A's teammate technical requirement. It's time to test their compatibility in work ethics and personality traits. Two students are determined to be compatible if three out of five work ethic and personality traits match. Below is a visualization of this behaviour:

Table 2 - Matching student work ethic and personality traits through an AND operations

Entity	**Operation**	**Bitstring / Result**
Student A		10011101 (Representation of work Ethic as bit string)
Student B		01101101 (Representation of work Ethic as bit string)
	AND	0000**11**0**1**
Team (A,B)	OR	**11111**0**1**

The algorithm has found a matching between Student A and Student B through a bitwise *AND* operation as at least **three** out of the five 1's have matched. Both students form a team, but now they aggregate their traits with a bitwise *OR*. This will result in the work ethic and personality trait of the team.

The Idea is that as more people are gathered to form a team, the union of their work ethics become the characteristic feature of the team, thereby defining the work ethic requirement for the next potential team match.

5. The process above resulted in an aggregation of the new team's work ethic and personality trait. Now that a complementary StudentUser B has been found for StudentUser A, the algorithm then normalizes the difference in how StudentUser B satisfied StudentUser A's teammate technical requirement. The Figure 2 below shows how this is achieved. After normalizing, a new team (of 2) has been created and the personal technical score of the team along with the teammate technical score of the team has now been produced. The algorithm now uses the aggregated work ethic along with the new technical score of the team (consisting of StudentUser A and B) to find a StudentUser C that will be entered into the new team. As the personal technical score of the new team fluctuates due to addition of new members, the algorithm tries to find the next user from the respective technical bucket that matches the team's technical score.

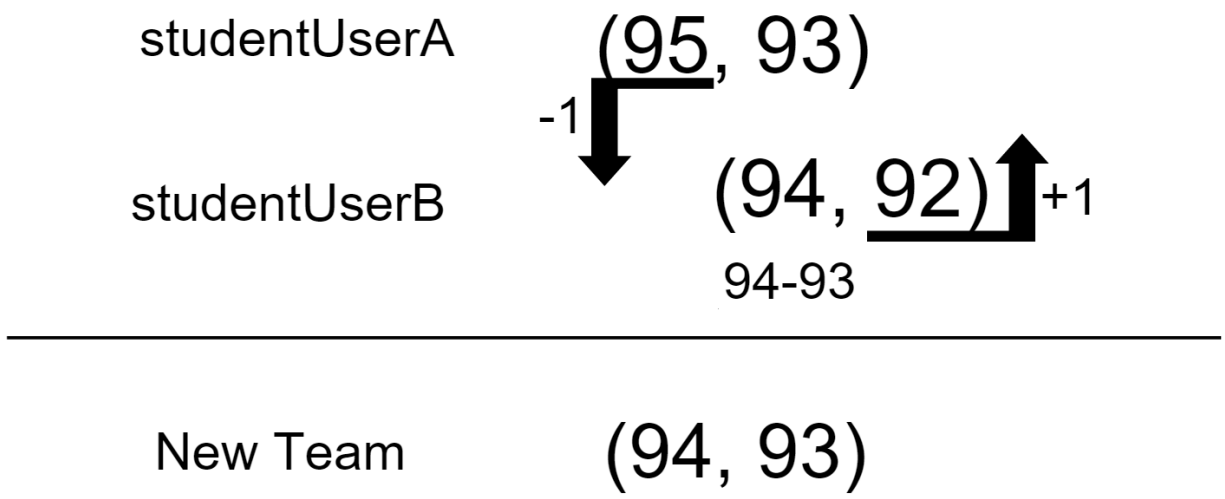

Figure 2 -Technical score normalization for newly matched team members.

6. The algorithm repeats steps **3, 4,** and **5** until a team whose number matches that of the configuration of the project is achieved. Once this is done to create one team successfully, the algorithm then begins from step **2** to create another team. This procedure is continued until all teams have been created for the project.

Algorithm Details (Known Edge cases and Work Arounds)

In this section, we cover the know edge cases that may arise during the execution of the algorithm and how they are respectively handled:

1. The first edge case is the issue of the number of students registered in the teams not being completely divisible by the team size. (i.e number of registered students % team size > 0). In this case, we define the **potential** (**P)** of a group of teams to be the difference between the size of the largest team subtracted from the size of the smallest team. At the expense of ambiguity, we provide an example:
    - Consider a scenario whereby the number of students registered is 51 and the team size configuration is set to 5. We know that we will have 10 teams of 5 StudentUsers and 1 team of 1 StudentUser. Hence the potential of the group of teams in this case is 4.

    Once the potential of the group of teams has been established, the algorithm computes a configuration where by: $0 \leq P \leq 1$. It should be noted that this computation is done before the team matching sequence is executed. Using the example provided above, the

algorithm then realizes that the best number of teams that satisfies the condition: $0 \leq P \leq 1$ is 4 teams of 4 StudentUsers and 7 teams of 5 StudentUsers.

2. The next edge case is the situation whereby a StudentUser's teammate technical score is greater than the personal technical score. In this case, it is to be noted that no special behavior is executed by the algorithm, but what is to be noted is the definition of the resulting behaviour. An example is provided to shed more light on this abstract:
    - Consider a scenario whereby StudentUser A has technical qualifications tuple (85, 92), and the best possible match found for such a user (based on work ethic and teammate requirement) by the algorithm was StudentUser B with technical qualifications tuple: (87, 87).

    As you can see in the example above, the teammate specification of the StudentUser A was not fully met because he/she was asking for a bit too much. The algorithm strives to satisfy a StudentUser's teammate request without losing sight of fairness. But we shall note something interesting as these two users are aggregated to form a team (we assume that StudentUser A and B satisfy the work ethic constraint of the algorithm). The technical qualifications tuple of the new team formed by the aggregation of StudentUsers A and B becomes (90, 84). As you can see, the personal technical score of the team increased due to the negative difference and consequently, the teammate technical score decreased. This a means of the algorithm adding priority to a given team's next match in order to satisfy the requirements of all the members that make up a team. (In this case, the next team member must be selected from the 90's bucket)

Observations and Reasons for Decisions

In this section, we outline the different attempts we came up with and their flaws.

- This first attempt of achieving such an algorithm was posed at using the concept of dynamic programming to create the **most optimal** teams for each projects. This design was a good choice at first thought but it proved to be computationally intensive (NP-complete), and hence, such an algorithm would have greatly used up time and other resources in order to create teams for the given project. The decision had to be made to discard this idea and go for an idea that guaranteed speed and also achieved optimality of some sort
- Our next attempt was to translate the problem to a graph problem, whereby StudentUsers became vertices and the edges became connections based on similar work ethic. In this case, an optimal team was to be formed by picking the subgraphs with the highest number of aggregate edge weight (which represented technical scores). This idea was also computationally intensive in that it used up a lot of space in order to provide the functionality of creating teams, and hence did not prove a good idea when weighed against scalability.

- Finally, we arrived at the algorithm stated above, whereby locally optimum teams are computed for each StudentUser which translates to an optimal team in aggregate. This idea was generous in space consumption and also proved to require less time in order to compute teams for a given project, and hence a decision was made to go with this one.

Revisions to Document

In order to maximize the teams satisfaction, a decision was made to calculate team's technical and required teammate score by averaging the technical and teammate score of the comprising team member. This is as opposed to the subtraction step done in step 5 of the algorithm.