From Talents to Team

Matching supply and demand with algorithms

Master thesis submitted to Delft University of Technology

in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in **Complex Systems Engineering & Management**

Faculty of Technology, Policy and Management

by

Maurits van der Goes

Student number: 1326252

To be defended in public on June 25th, 2020

Graduation committee

Chairperson & First Supervisor	: Prof.dr.ir. M.F.W.H.A. Janssen, Section ICT
Second Supervisor	: Dr.ir. Z. Roosenboom-Kwee, Section ETI
External Supervisor	: Ir. R. Boeije, Part-up

Summary

The globalizing economy with its new goods and services, knowledge spread, and competition for talent is an increasing complexity for organizations, which requires organizations to adapt more quickly. Organizations are essential to society, as people are more productive in groups. For their continuity, it is important that organizations continuously keep adapting to their environment. The new economy requires agile organizations that can quickly (co-)produce customized responses to the demands of the market. At the present moment, the hierarchical organizational structures face limits of their usefulness and are being replaced by a lean form of organization. In their place, these decentralized organizations are growing to overcome the limitations of these hierarchical structures. The coronavirus pandemic enforced a working from home policy that strongly decreased the access to work. This development stimulated organizations to move to decentralized organizing with a marketplace for team formation.

The introduction of self-managing teams enables this desired dynamic and leads to an agile organization. Virtualization has further enriched and diversified these teams. These self-managing teams are useful for constructive conflict, diversity, innovation, performance gains, and synergy. The coordination of work changes with the introduction of self-managing teams. These teams in here in a strong individual choice of people, which is both a benefit and a risk for people and organizations. A digital marketplace is a supporting institution that can provide overview, insights, and access to several opportunities by combining supply and demand of teams and talents across organizational borders. Connecting workers to relevant teams is also the main performance criterion. Still, bounded rationality can result in non-optimal individual choices on the marketplace and a one-size-fits-all approach does not match the preferences of the heterogeneous workers. A filter within the marketplace is needed to decrease the obstacles of decentral coordination of activities, however, there is limited research available on this subject.

This research follows the theory of design science research with the three cycles. The design objective is to support users in self-managing team formation with a recommender system and custom filtering algorithms. This is achieved with twelve design artifacts: personalization, criteria and objectives, recommender system, data engineering, ETL script, logical data model, custom filtering algorithms, data science, validation queries, hybrid filtering algorithm, DevOps, and advice on UX design and usage. The designed system was built with the requirements of the program of business demands. The constraints for the recommender system with custom algorithms are personalized recommendations for each worker and the incorporation of platform rules. The objectives are regular updates, scalable, open-source, and easy to set up.

The recommender system is set up separately from the marketplace for performance and scalability reasons. All data is stored in the graph database Neo4j with the plugin GraphAware Reco. This plugin is a recommendation framework that runs the algorithms that generate recommendations. There are five main filtering techniques: collaborative, content-based, demographic, knowledge-based, and utility-based. Collaborative filtering is a method that identifies similar workers or teams to suggest relevant teams. This network approach makes it domain independent. However, there is no known research on applying collaborative filtering to self-managing team formation. Demographic filtering uses chosen demographic

attributes such as language, location, and preferences to filter the available teams. Knowledge-based filtering applies knowledge engineering to the reason which items meet the requirements of a user. Content-based filtering uses content attributes, which are in this research unrestricted and accordingly very diverse. Utility-based filtering uses computation to determine the utility of an item for a user. The collaborative, demographic, and knowledge-based filtering techniques are chosen because these three are most suitable for the logical data model of this marketplace. These three filtering techniques are applied in the development of four filtering algorithms: a demographic filtering algorithm, a knowledge-based filtering algorithm, a user-based collaborative filtering algorithm, and an item-based collaborative filtering algorithm. Additionally, the fifth algorithm is a random filtering algorithm that functions as a control algorithm.

The recommender system and its algorithms were validated in three ways: quantitative validation, user interviews, and qualitative validation. The quantitative validation proved that the algorithms have a high user and item coverage while taking care of privacy by respecting the platform rules. 21 workers were interviewed to validate trust, user-preference, and utility. Latent trust was not enforced by combining familiar and new teams. However, it was surprising that workers tended to like teams better when those are presented as recommended. Alleviating them of their choice dilemma, even with only the suggestion of personalization, may seem counter-intuitive, but a confirmation that users are in need of support with self-managing team formation. Further, seven presentations and discussions reviewed confidence, robustness, and scalability. The multiple validations showed that collaborative filtering can indeed be used to personalize the team discovery. Most workers liked the recommendations and expected to join the team.

As a result of this research, a hybrid algorithm is implemented in the marketplace. The most important element is a user-based collaborative filtering algorithm. This is combined with a random algorithm to add serendipitous suggestions and overcome the cold start. The positive results of the validation led to the deployment of the recommender system into the marketplace. To realize the deployment of the system two challenges were resolved. The recommender system was updated with the feedback from the validation. New elements were developed to feed the system live data, retrieve the recommendations, and present the recommendations in the marketplace. All developed software is online available with an open-source license. The eventual deployment followed a step-by-step process to track down and fix all errors before the recommender system was opened to the public in November 2016. The successful deployment further proves the relevance of this research for the workers, marketplace, and academics.

This research has all three possible additions of design science research to the knowledge base: extensions to original theory, new meta-artifacts, and experiences gained. This research extends the literature on a recommender system and filtering algorithms for self-managing team formation. To the best of my knowledge was there no literature available on this subject. The required modules of a recommendation framework are studied and known information filtering algorithms adapted to the characteristics of self-managing teams. During the research, multiple design artifacts were developed, all open source or public domain. These artifacts were deployed as the initial recommender system of Part-up. This research demonstrates that recommender systems with custom algorithms can be used to ease the use of a marketplace for self-managing team formation. The successful implementation eases the manager of the central division of

tasks and team formation. At the same time, it empowers workers to self-organize and choose activities of their interest via a marketplace. This new way of working is expected to deliver a productivity increase for both managers and workers. The limitations of this research are the governance of algorithms, switching supply and demand, and detailed input. Governance and the ethical implications were not neglected, but while designing the system there was too little attention to the governance of algorithms. A feature that switches supply-and-demand can be quite easily developed with new data retrieval and graphic user interface components, relying on the same hybrid database, recommendation framework, and algorithms. Human interaction and judgment are difficult because not all interactions are tracked by the marketplace This can be better mitigated by analyzing and predicting users' behavior based on the fraction of interactions that were tracked.

Future research is advised on a case study with soft skills, artificial neural networks, hybrid algorithm, live feedback, and presentations of the recommendations. These directions can strengthen the technology or the adoption of such a marketplace with a recommender system. A marketplace with a smart filter will change the coordination of work. Workers will rely on the overview, easy communication, and vigorous organizational functionalities it provides. A new way of working that makes the workers more effective and happier, while also contributing to the goals of the organization. A transition that seems impossible without a supportive institution of a marketplace with a recommender system. Ready to overcome the complexity of the globalizing economy and stimulate a modern way of organizing.

Keywords: *algorithm; collaborative filtering; data engineering; data science; design science; marketplace; matching; personalization; recommendations; recommender system; self-managing team;*

Preface

For years I never believed nor may be hoped that this day would ever come. As this research marks the end of my academic adventure for now, leaving this university is accompanied by both a feeling of relief and unease. As I lack the guts to drop out, I figured that the only suitable escape was finishing it. In search of motivation, my curiosity made me drift even further away. It is my own Catch-22. A combination of actions that I only blame myself for. Regardless of everything, I love the knowledge-rich climate of universities with its passionate people.

While I was sometimes geographically or mentally far away from the Jaffalaan, there were two drivers to always return. My mother and father, who kept stimulating and supporting me in taking every small step. And the unique opportunity of this extensive research on a relevant subject. Part-up provided a symbiosis of intellectual challenge, social wealth, and time traveling. For this research, it was an honor to be supervised by this diverse committee. Marijn Janssen, Zenlin Roosenboom-Kwee, Ralph Boeije, Bram Klievink, and Scott Cunningham were essential in structuring my puzzled mind. Where asking Scott as a supervisor is without a doubt the best study decision I ever took.

Maybe this thesis reads like an ordinary research report. That is just the secondary objective and a requirement to close the book in Delft at last. When you discover the darkest corners of your life during your studies, a projected degree is not what drives you. I hope one day, I will fully value the education that I received, by contributing back to dynamic communities with new forms of knowledge sharing.

My primary objective for finishing this thesis was to show the value of an unpolished university journey. During this research finally, the proof originated that my deviating choices made sense. Never wait for knowledge to come to you: Ask questions and feed your curiosity. I was pointed back to tech by JC. Learned proper programming in Aalborg. Explored graph networks in the Utrecht Data School. Breathed the innovative culture of Silicon Valley. Steered accountability at DSC. And I was part of diverse teams throughout Europe. A selection of the experiences beyond my studies that were indispensable for this research. The truth is, this study trained my analytical mindset and gave me the excuse to combine these experiences. At last, this research became a steppingstone for my career. It was a decade that defined me.

Today I am very grateful for all of the family and friends that were there along the way. The precious moments that we shared. Their appreciation that I often underestimated. And the knowledge in a wide range of subjects that we gained. But also ashamed towards them I pushed away because I chose the easy-way-out.

Tomorrow is what it is all about. The warm-up is finally over. Hello world.

Maurits van der Goes

June 2020

Table of contents

Technology glossary

Algorithm	A sequence of automatic step-by-step rules.
API	Application Programming Interface. A convenient solution to providing a safe access point to create a connection between two software tools.
Attribute	In the context of graph theory, this is a variable that holds information. An attribute is always part of an entity or a relationship.
Collaborative filtering	Recommendation technique that uses the similarity of users' interest to suggest items.
Cypher	The query language of Neo4j. Similar to SQL (structured query language).
Demographic filtering	Recommendation technique where personal attributes of a user are matched with items.
Docker	Automated deployment and monitoring of software containers.
Entity	In the context of graph theory, this is a node in a graph database. It can be labeled or unlabeled and is also known as the vertex.
ETL	Extraction, Transformation, and Load is the collection of processes to move data from one data source to another.
Graph database	A database that stores its data in a network structure with entities and relationships. The linkages make it easy to extract patterns from the database.
Knowledge-based filtering	Recommendation technique where functional knowledge is used to suggest items that match a user's needs.
Machine learning	System of algorithms that discovers patterns from data without being specifically programmed for this. These algorithms are considered self-learning.,
Marketplace	A place where supply and demand meet. In this context, a marketplace matches the demand of workers with the supply of teams within and across organizational borders.
Memory-based	An approach where an algorithm is run every time a recommendation is needed.
MeteorJS	A JavaScript framework to generate a single page application.
MongoDB	A 'not only SQL' (NoSQL) database that stores information without schema as documents. It is also a popular database because it is open-source and available on multiple operating systems.
Model-based	An approach where an algorithm is run regularly to build a model from which the recommendations are read.

Neo4j	The leading graph database technology that is well-known for its reliable, native graph processing and storage. Written in Java. Available under an open-source or commercial license and for operating systems.
Platform	An environment that assembles people to obtain benefits from each other.
Recommender system	A type of information filtering system that predicts the relevancy of an item for a user by running an algorithm. The most relevant items are presented as recommendations.
Relationship	In the context of graph theory, this is a connection between two entities. A relationship can be labeled or unlabeled and directed (arc) or undirected (edge).
Self-managing team	A team that is responsible for its own organization and actions.
Software container	Virtualization of a Linux operating system to run a software application like a website, database, or API. Named after the shipping containers.
Team	A group of people that collaborate for joint actions.

1 | Introduction

1.1 | Organizations

Organizations face new complexity. Trade agreements, emerging markets, and new technologies create a new global economy (Townsend et al., 1998; Porter, 2000). This changing environment demands organizations to be increasingly more adaptive argues Aughton (1996). Globalization also requires that organizations deliver a new set of complex goods and services (Hidalgo & Hausmann, 2009). Simple, bulk products add little value. Even mass customized goods offer little sustainable advantage for companies and regions since these skills and products can easily be copied by others. Knowledge is spreading and widely available (Pinch et al., 2003). Private organizations are forced to specialize their activities and invest in knowledge capital. For their continuity, organizations need to be able to adapt more quickly. The individual worker becomes the most valuable asset of organizations.

Organizations are essential to society. They are a bundle of activities performed by people (Nelson & Winter, 1997). These collections are more productive than the individual. The success of organizations led to an increase in both the number and size of organizations. The symbiosis between organizations and society entails two-way traffic. In order to be a successful organization, its activities and structure should be matched successfully with society. There are many perspectives on organizations, but for now, it is assumed that organizations are a dynamic composition of activities and people with a collective purpose and external relations.

Many organizations currently have a hierarchical structure for the coordination of work, identified by Adam Smith as a product of the industrial revolution (Peart & Levy, 2009). The introduction of the assembly line required subdivision of tasks (Mintzberg, 1979), which in turn spurred governance. The common denominators are siloed departments, standardization of the work process, fixed job descriptions, and static hierarchy. The opacity and static nature of these organizational structures limit their ability to adapt to the changing environment. The bureaucratic structure is too rigid for the current turbulent ecosystem (Aughton, 1996). It must be replaced by highly adaptive structures. Traditionally the project-oriented team, an organic form of organizational design, was developed to create customized responses to market needs. Most of the US organizations chose a team-based structure and were pleased with it (Ranney & Deck, 1995). The hallmark of these organizational forms is an assemblage of individuals with specialized skills, deep knowledge of the problem, and an ability to broker solutions in concert with customers (Hobday, 2000). Organizations that are better able to adapt to external forces can grow and are perpetuated. The hierarchical structure has been effective for years but is now facing its limitations. A new organizational structure is needed to overcome these limitations.

Organizations are always trying to improve their efficiency by applying new organizational structures. In this process, they review their own and other structures, before switching to new approaches. A new option must fit the current activities and aspects of society. Organizational structures are the cumulative result of best practices for the current complexity they are facing. Presently, a new, often temporary form of organization,

well adapted to the needs of the 21st century is emerging (Bechky, 2006). Organizations are replacing the conventional hierarchy with a network of teams. Flat, non-hierarchical organizations with decentralized coordination of work are now quickly becoming the norm (Deloitte, 2016). Cutting out layers of management and a more flexible workforce are being introduced in traditionally hierarchical organizations. Lee & Edmondson (2017) describe self-managing organizations as systematic efforts to progressively decentralize leadership within an organization. The self-managing team is a component of this new approach. The introduction of self-managing teams is the result of two drivers: first, the desire for an adaptive organization requires an agile and lean structure with shared responsibilities, and second, the current educated and self-determined workers no longer fit the hierarchical structure (Aughton, 1996). This is a vigorous change for organizations. This shift in the coordination of work requires a new division of responsibilities. Workers move out of fixed functions and into temporary teams (Figure 1.1). Zenger and Hesterly (1997) visualize the infusion of markets in organizations by the shift from hierarchical governance to highly autonomous, market-governed internal units. The central coordination of work is replaced by decentralized coordination. Parallel to this shift a second change is visible: Networked collaboration between organizations. In a network environment sharing resources is actively promoted by leaders (Powell, 2003). Network organizing within and between organizations is seen as the modern key to success.

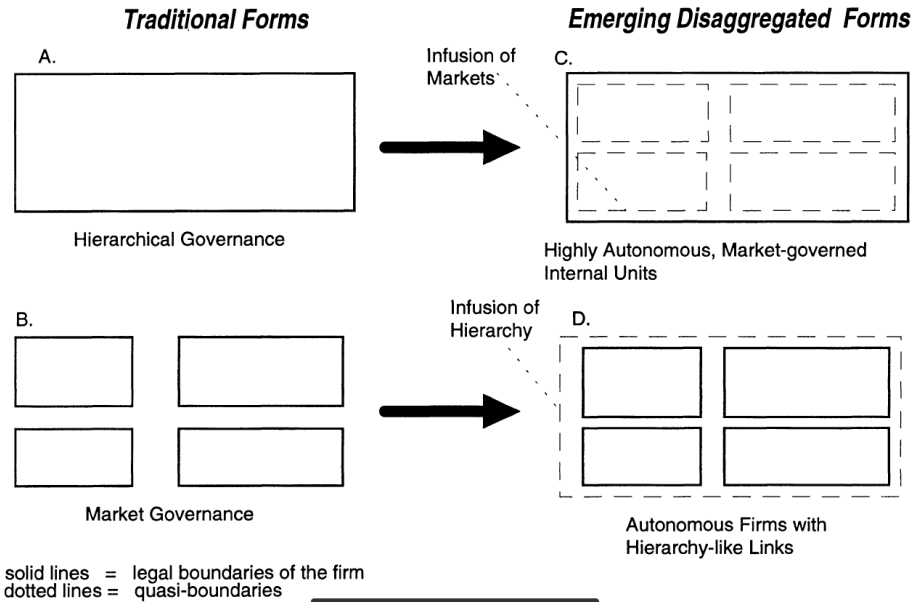

Figure 1.1: Shift toward decentralized organizations (Zenger & Hesterly, 1997)

1.2 | Self-managing teams

The introduction of self-managing teams enables this desired dynamic and leads to an agile organization. Self-managing teams are an evolutionary form of a network organization (Jarvenpaa & Leidner, 1998). By utilizing information technology, it is possible to surpass time, space, and culture because of the loose boundaries. Research demonstrates that when workers feel responsible for the process it leads to increased commitment, productivity, and quality (Aughton, 1996). Joost de Blok came up with Buurtzorg and

Eckart Wintzen introduced cell-theory, two successful examples of self-managing teams. Cummings (2004) also noticed the beneficial aspects of increased heterogeneity and higher demographic diversity. This decentralized structure challenges individual workers to choose for themselves what to work on and who to work with. The static hierarchical structures in place today have not taught them the value of change and flexibility, making individual workers reluctant to change. Problematically, when the number of options to choose from increases, people do not necessarily benefit from it. This is the tyranny of freedom according to Barry Schwartz (2000). As self-managing teams strongly rely on the organizational choices of individuals, this is both a benefit and a risk for the workers and organizations. Self-formation is the first autonomous act of organization by a self-managing team. For this reason, the teams in this report are labeled as self-managing teams rather than self-forming teams. In order to become more adaptive as an organization, the organization must help its workers to adapt to the new organizational structure.

Mid-March 2020 the coronavirus pandemic drastically changed work life. The Dutch government requested their citizens to work from home for social distancing (Rijksoverheid, 2020). Replacing in-person contact by videoconferencing, text messages, and calls. As a result, information distribution throughout organizations decreases, and working from home employees turned out to miss the overview of available work activities, as observed by a leader of Part-up (personal communication, 8 April 2020). At the same time, managers were struggling to divide the work remotely (Gripenstraw, 2020). A preferred management style of *'management by walking around'* was no longer possible. This delays decision-making and reduces access to work. To overcome this, organizations stimulated decentralized organizing. This resulted in more effort necessary for a successful match between an employee and an activity. Accordingly, organizations search for solutions that can support their workers with this culture change (Walsh, 2020). A leader of Part-up shared that before the corona outbreak, Part-up was already discussing with companies the opportunity to improve the access to work (personal communication, 4 June 2020). The subsequent corona crisis created both a necessity and momentum to speed up this implementation. Part-up's marketplace can reduce a manual employee-activity match from 45 minutes to a marketplace match in 1.5 minutes. This transition has two advantages: time savings, freedom of choice. Both contribute to a productivity increase.

1.3 | Marketplace

A platform is an environment that assembles people to obtain benefits from each other (Church, 2017). Fenwick et al. (2019) call the growth of digital platforms as significant, turning platforms into a routinized aspect of people's daily life. An emergence supported by the availability of digital devices like the computer and smartphone. This interconnectedness stimulates the network effects of the platforms, which are essential for the success of a platform. This research focuses on a core element of a digital platform: the marketplace for connecting users.

A digital marketplace is a supporting institution for workers in adaptive organizations. It accelerates this organizational change by providing overview, insights, and access to self-managing teams. A successful marketplace is about matching the demand of workers with the supply of teams within and across organizational borders. This marketplace can support freedom of choice for three reasons. First, it provides an

overview of the available opportunities and skills. Second, it gives insight into the interactions between people. And finally, it allows for control of people's own activities. The key to the success of the marketplace is the interplay between people and technology. However, choosing a self-managing team to join from this vast number of available teams creates a challenge for the workers.

In the present research, team formation is approached as a decision-making problem for an individual. The decision-making process aims for a satisfactory solution. It is affected by the bounded rationality of humans. This is the psychological assumption by Herbert Simon (1991) that in decision-making humans are limited rationally. They are unable to quickly process all the information that is required to make a balanced decision. This is caused by three limiting factors: information volume, cognitive capacity, and time. An individual is incapable to know all the alternatives for a decision. Due to limited cognitive capacity and time this individual is unable to compare the couple of alternatives that he knows. The limiting factors make the search process of a worker sequential. An individual stops at the first item found that meets his criteria and does not maximize his search effort. As a result, an individual makes a satisfactory decision instead of the optimal decision. Or the choice dilemma even paralyzes the individual (Hindle, 2008). The extent of this paralysis is not an objective of this present research. The observation that bounded rationality threatens the effective use of the platform is enough reason to further study its mitigations.

A marketplace is only supportive of workers if it matches supply and demand. Other marketplaces like Airbnb, Amazon, Netflix, and Spotify have faced the same problem (Koren et al., 2009). These platforms have applied an information filter to reduce or even prevent this problem. A system predicts the relevancy of an item for a user by running an algorithm. The most relevant items are presented as recommendations, while the other items are filtered out. A machine learning expert at Netflix expressed personalization as maximizing enjoyment and minimizing search time (personal communication, 8 June 2018). The type of information filtering system to automate this process is called a recommender system (Bobadilla et al., 2013). However, there is limited research available on the usage of recommender systems for filtering in a marketplace for team formation. This limits the successful introduction of self-managing teams and slows down the desired transformation to adaptive organizations, which is a requirement in the changing global economy. That is why the goal of this research is to study how in a digital marketplace the supply of self-managing teams can be filtered to the demands of workers. Vice versa matching is also an option. For example, a team creator invites recommended workers to a team. Although, this research chooses the viewpoint of the worker to design a clear solution. In the final part of the research and this report, the vice versa option is again addressed with a recommendation on how to efficiently use the artifacts of the designed system for this stakeholder.

1.4 | Structure

This research report is structured in eight chapters plus appendices. The definition phase is covered by the first three chapters. Chapter 2 analyses the characteristics of self-managing teams and recommender systems. Chapter 3 discusses the design approach of the research. The development phase is split into three chapters. Chapter 4 presents the design of the recommender system, while chapter 5 presents the design of the algorithms. In chapter 6, the results of the design evaluation are discussed. The development phase is addressed

in chapter 7. Finally, the report is concluded in chapter 8 with a discussion about the conclusions and recommendations for future research.

2 | Problem exploration

The purpose of this chapter is to further analyze the matching problem in this complex socio-technical system. First, the background, stakeholders, and characteristics of a self-managing team are discussed. It is important to understand these teams before improvements for the formation process are suggested. In the second part, the approach of information filtering including challenges and filtering is discussed. The third part describes the environment and requirements of the system. The theoretical foundation of this research and design objective is a recommender system with custom filtering algorithms to ease the usage of the digital marketplace. The theory of self-managing teams supports the research on recommender systems.

2.1 | Self-managing teams

2.1.1 | Historical perspective

Organizational structures have developed throughout the years by constantly adapting to changes in complexity. From the tribal organization forms of hunter-gatherers to the harsh authoritarianism of the middle ages and the authority based industrial era, we are now moving into more communitarian and holistic structures. These organizational forms were always fit for their purpose: both the value systems of the individuals and the tools at hand for task distribution, overview, and communication.

Kondratiev argued first that the development of the world economy can be divided into cycles. These business cycles are driven by innovative technology affecting all industries. The industrial revolution itself was for example driven by the invention of the steam engine (Allen, 2009). Schumpeter (1939) further developed the concept and Linstone (2002) acknowledges these cycles. Currently, information technology is the overarching technology (figure 2.1).

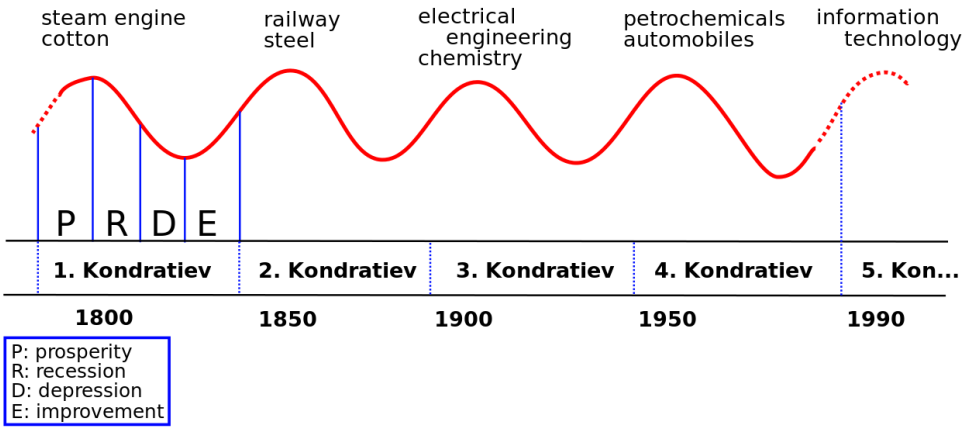

Figure 2.1: Kondratiev waves (Wikimedia Commons, 2009)

2.1.2 | Virtualization

Information technology creates a new dimension for teams: virtualization. This is one of the modern organizing trends identified by De Bruijn et al. (2014). Castro and McQuinn (2015) noticed that already half

of the world's services are digitalized. New technology makes collaboration between dispersed coworkers possible. Modern teams are location independent, integrate online and offline communication, and overcome technical failures. During the COVID-19 pandemic, the Amsterdam Internet Exchange (2020) experienced a 17% increase in traffic volume compared to February 2020. Walther (1997) notes in his research that the competence of social information exchange does not differ between face-to-face communication and computer-mediated communication. This report does not speak about virtual teams, but self-managing teams. I agree with Venkatraman and Henderson (1998) that a virtual organization is not a distinct structure. Rather it is a strategic characteristic that is complementary to functional, divisional, matrix or network organizations. Also, it is difficult to determine when a team is primarily virtual or nonvirtual. This is always intertwined.

2.1.4 | Characteristics

These modern, self-managing teams have five advantageous characteristics (Gibson & Cohen, 2003): constructive conflict, diversity, innovation, performance gains, and synergy. However, self-managing teams are only effective if there are a shared understanding, integration, mutual trust, and limited team size. An aligned effort increases the motivation and satisfaction of the team members. For profiling team members, Bell (2007) makes a distinction between surface-level and deep-level composition variables as predictors of team performance. The former are demographic characteristics of a user, the latter are psychological characteristics like a user's personality, values, and attitude. Over time, while collaborating the diversity of deep-level variables strengthens, and diversity of surface-level variables weakens (Harrison et al., 2003). Algorithms including both types of variables were developed, either based on demographic variables or soft skill variables from a personality test.

Workers must trust each other: *"Trust is the glue of the global workspace."* note O'Hara-Devereaux and Johansen (1994, p. 243). However, Langfred (2004) concluded that too much trust leads to decreased monitoring of team members. A very important factor for trust is communication. This functions best in a supportive and conversational climate.

Team size is influencing this conversational climate, as extra team members do not necessarily mean a better team performance, because of possible communication mishaps and dysfunctional conflict. This especially applies to the new virtual domain. There is a performance loss if a team grows from one individual to two or three, a phenomenon known as the Ringelmann effect (Ingham et al., 1974). This is the discrepancy between potential and actual productivity. When adding a fourth, fifth, and sixth member additional decrements occur. If one individual performs at 100%, then the well-known research by Max Ringelmann shows that this decreases to an average of 93% for a group of two, 85% for a group of three, 77% for a group of four and down to 49% for a group of eight (Moede, 1927). The performance loss is caused by a lack of motivation, complex coordination, or a combination of both. First, members will decrease their effort when they feel less responsible for the output (Rond, 2012). This can be either intentionally or unintentionally. However, in both cases, it affects the individual action and the motivation of the other team members. In teams that are too large, it is very difficult to relate your effort to collective performance. This missing feedback causes a decrease in responsibility. Second, coordination is affected by the necessary number of conversations.

Hackman (1983) discovered that within a team of five members it takes only ten conversations for every person to touch base. If the group size grows to thirteen the number of necessary conversations for all persons combined increases to 78. Research by Darleen DeRosa and Richard Lepsinger (2010) confirms this by noticing that self-managing teams with thirteen members or more score the worst. When forming a team, it is important to take the team size into account, as performance decreases with too many members. It is preferred to form teams of eight members or smaller and not larger than thirteen members.

2.2 | Information filtering

2.2.1 | Introduction to information filtering

This complexity of self-managing teams underlines the necessity of supporting the workers in their choice dilemma. Information filtering is a common method to address this challenge. Workers are steered towards the item that fits their profile best by removing irrelevant information (Tang et al., 2013). A worker perceives the filtered overview of relevant teams as recommendations. A recommender system automates the process of information filtering. It is expected that the new technology of the marketplace for team formation is adopted more easily with these personalized recommendations (Levy et al., 1993). Each recommendation must be presented in the proper context and with detailed information according to Sinha and Swearingen (2001). To determine the usefulness of a recommended item a worker prefers a more extensive description. Usually, this is at first hidden in the user interface, but it can be easily shown in a good design. A clear interface is especially important for a successful recommendation. A recommender system can incorporate these described characteristics of a self-managing team into the filtering algorithms. This is important in the process to advise relevant teams to the workers. An algorithm is a sequence of automatic step-by-step rules (Cormen et al., 2009). Algorithms are part of people's daily lives. For example, Google's search engine runs the PageRank algorithm and the Elo-rating is indispensable for the chess world, but also for Tinder. Generating personalized recommendations is the main constraint for the proposed solution.

2.2.2 | Challenges

Multiple researchers (Resnick & Varian, 1997; Su & Khoshgoftaar, 2009; Melville & Sindhwani, 2011) identify four main challenges for generating personalized recommendations: data sparsity, gray sheep, shilling attacks, and synonymy. Each of these four challenges is briefly discussed and incorporated while designing the algorithms in chapter 4. Data sparsity occurs when workers neglect to rate or interact with teams. The worker-team matrix is sparse (Melville et al., 2002) and results in a low amount of comparable ratings that are required for a proper statistical analysis. This challenge often happens when the system has a very high team-to-worker ratio or with the entry of a new worker: the cold start problem. The lack of activity in teams means that the system cannot produce any recommendations (Ogul & Ekmekciler, 2012). After all, there is no data to perform the statistical analysis. The team-to-worker ratio is 1:4 in the dataset used, thus not an issue. The cold start problem is a challenge because a new user profile can hold very limited or diverse information. This also applies to gray sheep in the user base. A gray sheep is a worker with ratings on teams that are not

consistent with any other group of workers. The diversity activities that the marketplace wants to support, means that an algorithm must figure this out. Shilling attacks are fake profiles to influence the predicted ratings (Gunes, 2014), often to stimulate the popularity of a team. Ratings are limited used in the platform; the focus is on user activity. Accordingly, shilling attacks are a minor challenge. Synonymy covers language issues. The tags *'coding'* and *'programming'* mean the same to humans, but not to an algorithm. According to Su and Khoshgoftaar (2009), this degree in variability is larger than commonly suspected. Again, the freedom in specifying and the decision to not categorize users and teams, implies that synonymy was a challenge while designing the algorithms. Although, with the attention to diversity these challenges were not insurmountable while designing the solution.

2.2.3 | Governance

The benefits of automated filtering are accompanied by risks and ethical implications. Governance is an important design aspect of technology. As the filtering algorithm is hiding options for a user, a part of this human decision is influenced by the algorithm. It is a unique universe of information that is created by a system, which changes our perception of information and the creation of ideas (Pariser, 2011). This is considered algorithmic bias. Technology is not neutral. An algorithmic bias expert of Spotify claimed that human decisions do affect machine learning outcomes, and a lead scientist of Google agreed with the importance of addressing bias in algorithm development (personal communication, 8 June 2018). In journalism, the filter bubble is a threat that is often referred to. Perhaps there is a team bubble too, that gives the impression that the best team is composed while relevant alternatives are left out by a system. During the corona crisis, this is likely to be even more important. The working from home policy limits a user in comparing the overview of recommended teams to the overview based on office interactions. The dependency on a fair algorithm increased.

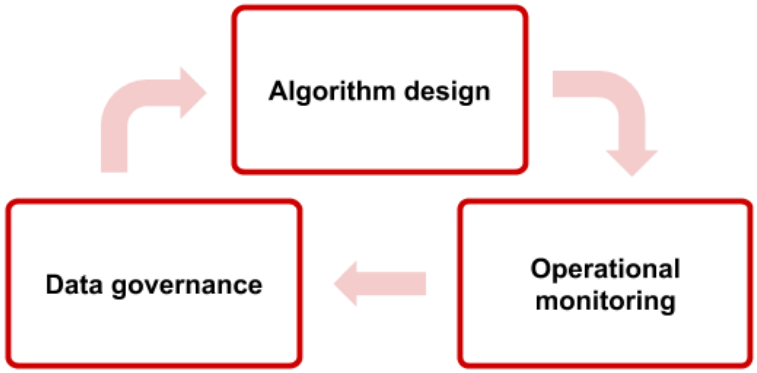

Figure 2.2: Governance of algorithms

The governance of algorithms is a continuous process that consists of three domains (figure 2.2): data governance, algorithm design, and operational monitoring. Data governance is required because algorithmic bias can also be non-human and unintentional when there is a bias in the data source (Haijan et al., 2016). To detect algorithmic bias in data, Cramer et al. (2019) developed a prototype checklist with eight types. While designing algorithms, data scientists need to be aware of the bias that they add to an algorithm. To enforce a

four-eye principle, Arets emphasizes the importance of algorithmic transparency (Utrecht University, 2019). Information filtering can analyze user behavior. Still, the privacy of each user should be ensured in the algorithm design. Just like other privacy restrictions of the marketplace. Operational monitoring cannot be neglected because even if an algorithm is carefully designed, it can still show unexpected behavior (Bozdag, 2013). For example, to report changes in the data sources or users that actively try to influence the algorithm. The governance of algorithms in these three domains is the responsibility of the marketplace owner. Overall, when using recommendations, users should also be aware of any bias in these recommendations.

2.3 | Environment

Supporting the growth of these self-managing teams within organizations is exactly what three alumni of the Faculty Technology, Policy and Management started. In 2014 they founded Part-up, a startup with the mission to change organizational structures by empowering users in choosing their own activities and teams. With this startup, they have developed an online marketplace for the formation of these teams and collaboration on tasks. Individuals can start or join teams of their own or other organizations.

2.3.1 | Stakeholders

There are five stakeholders for a recommender system with custom filtering algorithms, presented with their goals in table 2.1. The leadership team of an organization ranges from the board of directors to managers. Workers of an organization are the people who contribute to the organization on a contract or freelance basis. Together they organize themselves in an effective structure. There is no difference in workers, even if a worker started a team. The marketplace owner, in this research the leadership of Part-up, determines the mission, purpose, features, and development roadmap of the platform. Personalization provides a service to users if performs as expected and does not negatively affect other marketplace features. Subsequently, governance is the responsibility of the marketplace owner, although performed by the data scientists who hold the necessary knowledge. The developers of the marketplace build the features as planned on the roadmap. They are responsible for ensuring the stability of the platform. A recommender system with complex algorithms can affect the performance and resource costs of the marketplace. The logical data model is maintained by the developers on advice of the data scientists. The data scientists design new algorithms, improve the performance of current algorithms, and monitor the governance of operational algorithms. Although there are five stakeholder types, a person can belong to multiple types. For example, a developer that is also a data scientist and organizes his work via the marketplace.

Stakeholder	Goal
Leadership team	Efficient organizing of employees and external workers by

	dividing activities.
Workers	Work on activities that fit their skills, interests, and availability.
Marketplace owner (Part-up)	Providing a stable and transparent marketplace service to let organizations effectively and trustworthy organize work themselves.
Developers	Develop and maintain a stable marketplace.
Data scientists	Design, improve, and monitor algorithms that match workers and activities.

Table 2.1: Stakeholders

2.3.2 | Requirements

In each team are multiple actors involved, with different interests. Next to the constraint of personalized recommendations, there is one other constraint of the proposed solution that is specifically important of Part-up. Their marketplace has public and private teams. A worker can only access a private team if he is also a member of the corresponding network. That is why it is a constraint that the recommender system follows the platform rules when generating recommendations. These two constraints are accompanied by four objectives (figure 2.23). An objective is a non-critical goal for the solution. There are four objectives formulated: regular updates of the recommendations, scalability, open-source software and easy to set up.

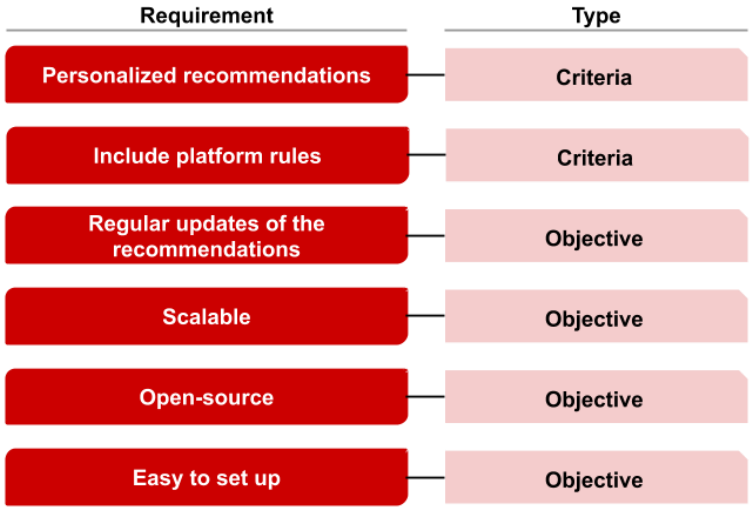

Figure 2.3: Program of business demands

The first objective is regular updates of the recommendations. Designing a solution that can be integrated differs in some respects to a research setup. The activity of the marketplace is changing the information in the database constantly. This means that the dataset to generate recommendations is also dynamic. If the generated recommendations do not get updated, then the recommender system becomes outdated and irrelevant. That is why the system must regularly update its recommendations. The second objective is the scalability of both the recommender system and its algorithm. When the database volume grows, the information filtering process should not be noticeably affected. This performance is expressed in the time it takes to generate the recommendations. There are multiple ways to ensure high performance in all

circumstances and it starts with writing efficient code (Sarwar et al., 2002). Other more complex methods like dimensionality reduction or singular value decomposition are not applied. These methods can affect the quality of the recommendations negatively in favor of the performance. This trade-off must be researched separately. The third objective is the usage of open-source software. Open-source software has a large financial advantage, as it often lacks a license fee (Lakhani & von Hippel, 2003). Especially for fellow developers of open-source software like Part-up, which published all their code under an open-source license. Another advantage is that this research lowers the barriers for others to replicate the solution and develop a recommender system themselves. This also aligns with the vision of the former Undersecretary for Education, Culture and Science Sander Dekker (2013) that academic research should always be freely available under the Open Access principle. According to him, this contributes to the spread of knowledge and the promotion of innovation. The fourth and final objective is an easy setup of the recommender system. The Faculty of Technology, Policy and Management, Part-up, and me have very limited knowledge of designing and developing a recommender system. I rely on online knowledge, external experts, shared use cases, and available technology. For these reasons it is preferred that the recommender system is easy to set up.

2.4 | Conclusions

This literature review delivers three crucial components of the proposed solution: purpose, constraints, and objectives. The purpose of this socio-technical system is supporting workers in discovering and joining self-managing teams within and across organizational borders. The function of the possible solution is to generate personalized recommendations for relevant teams. A structure with this function is a recommender system with custom algorithms. The design requirements for this recommender system consist of two constraints and four objectives. The constraints are personalized recommendations for each worker and the implementation of the platform rules. The objectives are regular updates of the recommendations, scalable architecture and algorithms, open-source software, and easy to set up. The purpose, function, constraints, and objectives determine the goal and scope of the research and thereby function as the program of business demands. This guideline connects the business environment with its strategic objectives to the architecture that is developed (Nijhuis, 2006). The distinction between the constraints and objectives implies a trade-off in preferences and is a clear indication of the requirements. According to Janssen (2009), this guideline can also be used to assess the quality in retrospect. The program of business demands is the starting point for the research that is described in detail in the next chapter.

3 | Design approach

This chapter presents the approach of designing the proposed solution: a recommender system with algorithms for filtering self-managing teams There are multiple platforms that support their users with a recommender system, but the problem at hand is a bit different than a regular recommendation challenge. The definition of a team as described in the previous chapters implies that a recommendation can only be consumed by a limited number of users. A movie, for example, is with its characteristics not limiting the number of users who can watch it. Parts of the knowledge that other platforms share about the architecture and algorithms can be integrated into this present research for designing this new solution. However, the problem exploration exposed that there is limited academic research available on such a specific system in the domain of teams. That is why the research objective is to address this knowledge gap on recommender systems within a team marketplace, which must lead to the design of a reliable and suitable recommender system for joining self-managing teams. As this solution is a support system, the workers who use the system are further defined as users in this report. This chapter further addresses the research questions, design artifacts, dataset, and concise the validation of the research to structure the design phase.

3.1 | Design objective

The observations as presented in the first two chapters result in the design objective and four sub-questions. This research has the design objective to support users in self-managing team formation with a recommender system and custom filtering algorithms. This design objective is achieved with four sub-questions, each addressing a different subsystem within this research:

1. What is the open-source architecture design of a recommender system for this marketplace?
2. Which information filtering algorithm is evaluated best from theory and user interviews?
3. How is bias in algorithm design addressed by the governance of algorithms?
4. What steps are needed to integrate the artifacts of the recommender system and evaluated algorithm in the marketplace to provide new opportunities?

3.2 | Research method

This research follows the theory of design science research, with the design science research cycles by Hevner (2007). This framework builds on the information science research framework (Hevner et al. 2004) by adding three cycles: relevance cycle, design cycle, and rigor cycle (figure 3.1). The relevance cycle collects the application context from the Part-up marketplace. This consists of the requirements and the technical systems of the marketplace. The rigor cycle provides the necessary knowledge base. Hevner (2007) argues to ground design science research on various ideas: theories, opportunities from the relevance cycle, existing artifacts, and analogies. Exactly the mixture that this research combines, with the theory on information filtering and team characteristics and artifacts of recommender systems. The design cycle of this research

creates the recommender system with its custom filtering algorithms and algorithms in quick iterations and evaluates it once extensively with qualitative validation, quantitative validation, and user interviews. Afterward, the recommender system is field-tested on technical feasibility in the deployment phase. The inputs are the requirements from the relevance cycle and the rigor cycle.

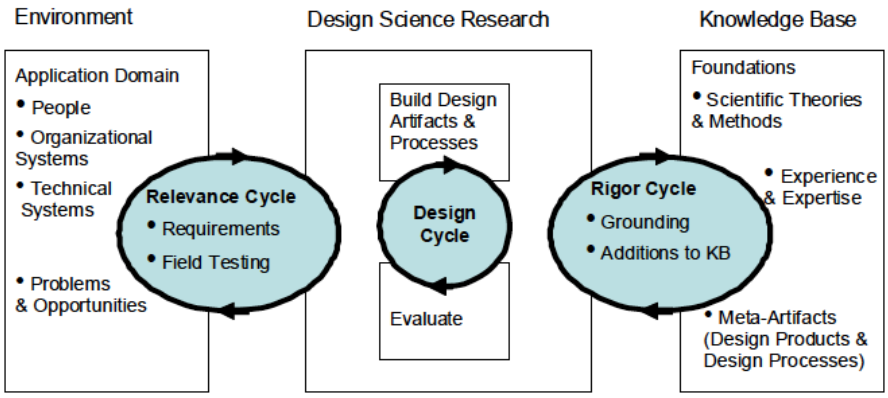

Figure 3.1: Design Science Research Cycles

Within this design science research, multiple methods for the creation of artifacts were applied. Three variants for the recommender system were designed and developed with software engineering. One was selected with a best-of-class chart and designed in detail with the architecture design meta-framework (Janssen, 2009). The algorithms are adapted algorithms from scientific methods and meta-artifacts, further developed with software engineering. Both the recommender system and the algorithms were evaluated with qualitative validation, user interviews, and presentations with expert feedback. Governance advice is the result of scientific theories and discussions with experts. The evaluated recommender system and the chosen algorithm are deployed following the DevOps methodology (Bass et al., 2015).

3.3 | Design artifacts

The design cycle builds artifacts to solve organizational problems (Hevner et al, 2004). Offerman et al. (2010) identified after literature research of design science publications a typology with eight IT design artifacts. Following this typology of Offerman et al. (2010) twelve artifacts are identified, listed in figure 3.2.

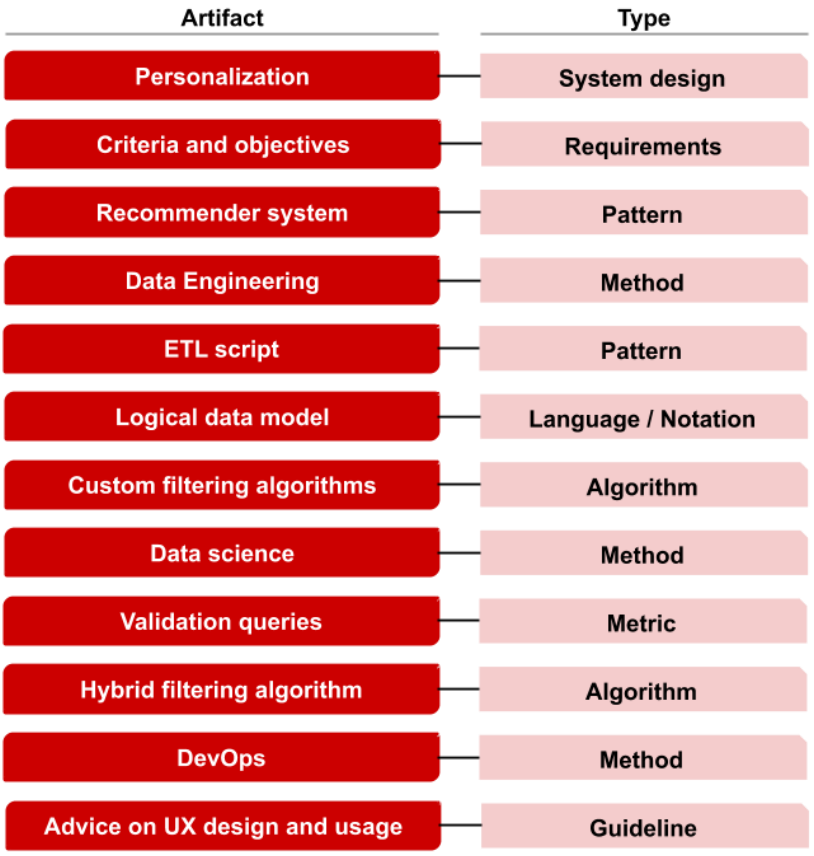

Figure 3.2: Design artifacts and types.

Personalization is a system design artifact. It describes the structure and process of an IT-related system. The criteria and objectives as formulated in chapter 2 are requirement artifacts. In the system design, the requirements define the design space and describe the development goals. The recommender system is a pattern artifact, a system design element that is generalized. The recommender system consists of the recommendation engine, post-processor, blacklist, filter, logger, and config modules. Data engineering, a type of software engineering, is the method artifact practiced in the design cycle to develop the recommender system. The ELT script to extract, transform, and load the necessary data from the marketplace database to the recommendation database is a pattern artifact. The logical data model is a language / notation artifact that shows the interrelation between data entities using a graphical notation. It supports the development of the system. The four custom filtering algorithms and the hybrid filtering algorithms obviously have the algorithm artifact type. This artifact is a sequence of activities executed by a computer. Data science is the method artifact to develop the algorithms. The validation queries are examples of metric artifacts. These mathematical metrics evaluate the system design from a business perspective. DevOps, development and operations, is the method artifact used in the deployment of the initial recommender system. The advice on the UX design and user usage of the recommendations within the marketplace is a guideline artifact.

3.4 | Available dataset

This research requires a dataset to design and validate the recommender system with its custom filtering algorithms. As the facilitator of the research, Part-up is providing a dataset of their own marketplace. On 31 July 2015 Part-up launched its marketplace. Their database contains many different attributes that express the activities of the users in their different teams. For security reasons, user passwords are excluded. In the Part-up ecosystem, a user is called an upper, a team is a part-up and a network is a tribe. For independency and readability reasons, the present report will use common terms like user, team, and network. For this research a copy of the database from 27 January 2016 is used, the day that the evaluation started. As the platform was growing it had a maximum of entities in its six months of existence. At that moment the marketplace had 2.743 users, 656 teams, and 184 networks with an active status. This distinction does not imply any activity of the entity (user, team, or network) but confirms that this entity has not been deactivated. Two other aspects of this dataset must be noted. First, the logical data model of the marketplace was already chosen. This restricted changes or additions of attributes. Second, not everybody is digitally savvy or comfortable in the virtual domain, and that implies not all professional interactions take place inside the marketplace. Still, the volume and diversity of this dataset provide intriguing opportunities for this research.

3.5 | Evaluation

Often recommender systems are validated only on accuracy (Said & Bellogín, 2014). This is the system's ability to generate recommended teams based on a training set that is the same as chosen teams in the test set. The dataset as provided by Part-up covers a period of just six months and the marketplace has no recommender system yet. For these reasons, an interviewed expert reasoned that data validation was not reliable and I agree with his argumentation (personal communication, 2015). That is why a different approach was chosen. There are three types of experiments to review a recommender system: offline, online, and user study. For an offline experiment, exported data is used in a local environment to measure the performance of the recommender system. An online experiment is performed within the operational system. Finally, in a user study questions about the recommender system are asked to a small set of users. Online experiments cannot be performed, as there is currently not a recommender system integrated into the Part-up marketplace. This offline database with the provided dataset is running locally on a laptop. That is why in this research only the offline and user interviews experiments are used to validate the recommender system.

4 | Design of the recommender system

This fourth chapter discusses the design of the recommender system architecture, operationalized from the program of business demands. As mentioned, the research architecture is set up separately from the marketplace. The integration with the marketplace in the deployment phase will be discussed in chapter 7. Also, for the research setup, there are three path dependencies in the ICT-architecture to be considered. These are the JavaScript framework MeteorJS, database MongoDB, and the chosen logical data model.

The recommender system is designed following design science requirements, the criteria, and objectives from the program of business demands as formulated in chapter 2. The criteria are personalized recommendations and include platform rules. The objectives are regular updates of the recommendations, scalable, open-source, and easy to set up.

4.1 | Marketplace

The platform roughly consists of two parts: a marketplace for team formation and team pages for collaboration. The user activity flow in the marketplace is presented in figure 4.1. In the marketplace, each user has the option to create a team or search for any accessible team. When creating a team, a user provides basic information about the team, like title, description, privacy type, end date, tags, and activities. The team overview presents all accessible teams for a user. This research adds the overview of recommended teams to the marketplace. This is a personalized view generated by information filtering algorithms. A user can decide to join a team as a supporter or search for other teams. If the user picks a task, he or she becomes a partner in a team. It is also possible to keep following a team as a supporter and search for other teams. Partners collaborate on tasks, while maybe also searching for other teams or collaborating in other teams. This collaborative environment is not part of the marketplace aspect of the platform. But the user behavior can be valuable data for an algorithm to personalize the team overview for a user.

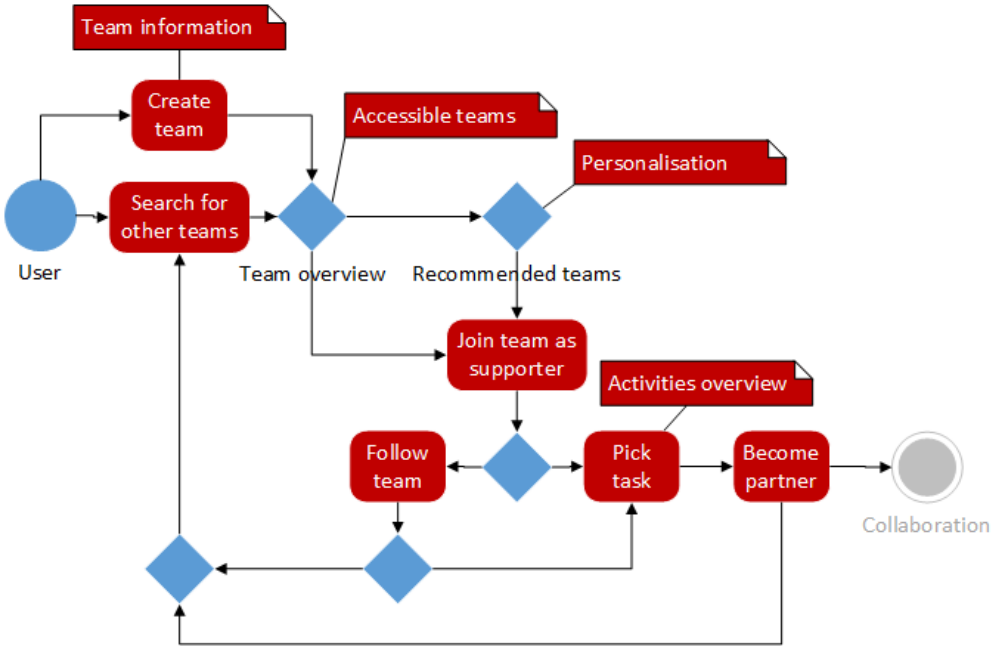

Figure 4.1: User activity flow in the marketplace (UML diagram).

The Part-up marketplace uses the JavaScript web framework MeteorJS for their website. It is the core on which the website is designed and developed (Lehmann et al, 2015). All the data of this marketplace is stored in a MongoDB database, as this is the usual companion of the MeteorJS framework. MongoDB is a document-oriented NoSQL database, which implies that the database does not have a fixed table-based structure. The database is divided into collections of documents. For Part-up these are for example users, networks, teams, contributions, comments, and ratings. The lack of table-structure means that there is not a predefined list of attributes. However, that does not mean that there is no logical data model present in the database. This logical data model arises naturally with every value that is stored. Every night the MongoDB database generates a data dump with all the data except sensitive information like passwords. This can be easily extracted to a local hosted MongoDB database. Another option to access the data is via the MeteorJS framework. By integrating code in the marketplace or listening to the events and recreate the database. For security reasons, it is not possible to access the marketplace MongoDB database directly.

4.2 | Technology selection

A recommender system is a combination of an information searching and information filter. The core activity of this system is processing data. That is why the choice for a specific database type is essential in designing an appropriate research architecture. There are three options identified for the research architecture:

- Document NoSQL database: MongoDB with MeteorJS. Alternatives for MongoDB are CouchDB and RavenDB.
- Relational database: MySQL with custom Java code. Alternatives for MySQL are Oracle and Microsoft SQL Server.

- Graph NoSQL database: Neo4j with GraphAware Reco. GraphAware provides a recommender framework plugin for Neo4j. Alternatives for Neo4j are Dex, Objectivity/DB, OrientDB, and Titan.

These three prescriptive technical structures are compared using a best-of-class chart. The advantages of this method are that it is easy to explain and that each alternative is evaluated with respect to the results of each metric (Dym et al., 2014). It is also a subjective method with a nontransparent process behind it. For choosing the research architecture this is an acceptable disadvantage. This research is about the first application of a recommender system with custom filtering algorithms in a marketplace for team formation, and not about designing the most optimal system and algorithms.

	MongoDB with MeteorJS (Document NoSQL DB[1])	**MySQL with custom Java code (Relational DB[2])**	**Neo4j with GraphAware Reco (Graph NoSQL DB)**
C: Personalized recommendations	2	2	1
C: Include platform rules	1	3	2
O: Regular updates of the recommendations	2	3	1
O: Scalable	3	2	1
O: Open source	1	1	3
O: Easy to set up	2	3	1

Table 4.1: Best-of-class chart (Neo4j, 2016a)

(C: criteria; O: objective)

The option of Neo4j with GraphAware Reco scores best for all constraints and objectives except for including platform rules and open source. Confirmation of the constraints is obliged and most important. Generating personalized recommendations is expected to be most easy with Neo4j because GraphAware Reco is a proven plugin with this functionality. For the other two options, such a recommender framework must be custom developed. Including the platform rules are most easy for the MongoDB-option as it can reuse the JavaScript code within MeteorJS that was written for this purpose. GraphAware Reco provides via its blacklist and filter tools to enforce the platform rules onto Neo4j. Realizing this constraint with the MySQL-option requires again writing new code.

Fulfilling all objectives is desired, but not compulsory. GraphAware Reco enables regular updates of the recommendations that are stored in Neo4j. For the MongoDB-option the cron job scheduler via MeteorJS can be used. Again, for the MySQL-option, this feature must be developed from scratch. The Neo4j-option is best scalable because both Neo4j and GraphAware Reco provide several options and support to realize this. The MySQL-option is also quite scalable by adding plugins. The MongoDB-option is the least scalable because the recommender function would be included in the MeteorJS framework. This restricts independently scaling,

[1] http://neo4j.com/developer/graph-db-vs-nosql/
[2] http://neo4j.com/developer/graph-db-vs-rdbms/

a freedom that the other two options do have. All three options are open source, but Neo4j and GraphAware limit this to only non-commercial use. For this research Neo4j (2016a) provided an educational license (personal communication, 7 December 2015). Part-up received a startup license by Neo4j for using the technology in its marketplace. They can also apply for the Fair-Trade Licensing option, as all code of Part-up is open-source available under a GNU AGPLv3 license. Affero General Public License (AGPL) is a license that protects the public modification, sharing, and usage of the software. All derived software products may only be shared under the same GNU AGPLv3 license.

A similar agreement was made with a manager of GraphAware (personal communication, 26 April 2016). It was confirmed that Part-up can use their products indefinitely without costs in return for sharing their own code and implementation experiences. With all the available plugins, knowledge, and support the Neo4j-option is expected to be most easy to set up. For this objective, it is both an advantage and a challenge that the MongoDB-option is intertwined with the marketplace. It gives a head start but also complicates the development because it must be integrated with respect for the other modules. As mentioned, the MySQL-option requires many lines of new code. This means that it is not easy to set up.

Based on the best-of-class chart and this accompanying analysis, the best research architecture is a graph NoSQL database: Neo4j with GraphAware Reco. This option conforms to the constraints and scores best on the objectives. The core of this chosen research architecture is Neo4j as the recommendation database. Neo4j is available as an embeddable Java server and its transactions are reliable because it is full ACID: atomicity, consistency, isolation, and durability. The GraphAware framework will function as the recommendation framework running the algorithms. A script will be written in Java to extract, transform, and load (ETL) the data from MongoDB to Neo4j. The architecture is structured in table 4.2 following the architecture design meta-framework by Janssen (2009) and visualized in figure 4.2. The meta-framework highlights the focus of the architecture. In the next paragraphs, this architecture is further described.

Business architecture	Generate personalized recommendations for interesting self-managing teams to support users in discovering and joining teams within and across organizational borders.
Business process architecture	Analyze the profiles and activities of the users and the features of the teams. Compute and rank the teams on relevance to each user using an algorithm. Present the most relevant teams to the users. The first and second processes need to regularly run to ensure up to date recommendations.
Information architecture	Extracting the necessary information from the marketplace database to the recommendation database using an ETL script that applies the logical data model. The recommendation framework uses this information to analyze, compute, rank, and store the recommended teams for each user.
Application architecture	● Extract data from the marketplace database ● Transform data to the defined logical data model ● Store data in the recommendation database ● Extract data from the recommendation database ● Generate recommendations using an algorithm
Technical architecture	Marketplace database (MongoDB), ETL script (Java), recommendation database (Neo4j) and recommendation framework (GraphAware plugin)

Table 4.2: Layers from the architecture design meta-framework (Janssen, 2009)

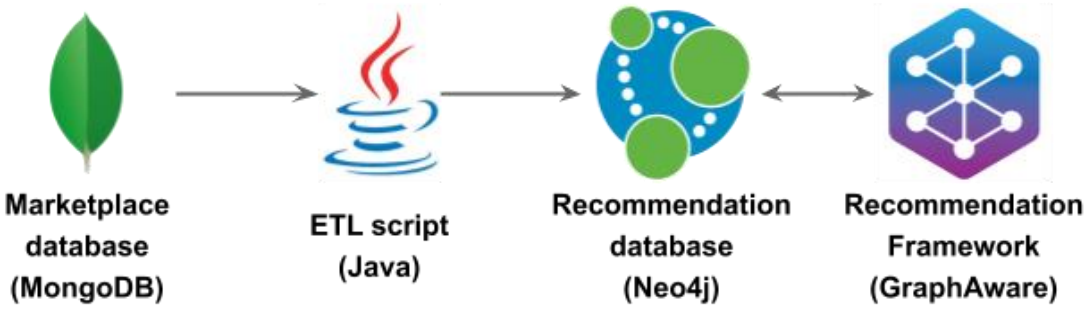

Figure 4.2: Chosen design of the research architecture

The subsystems of a socio-technical system can be split into static and dynamic subsystems (table 4.3). The static subsystem structures the behavior of a process (dynamic subsystem).

Static subsystem (Structure)	Dynamic subsystem (Process)
Marketplace database (MongoDB)	Generate the daily data dump
ETL script (Java)	Extract, transform and load data
Recommendation database (Neo4j)	Store data
Recommendation framework (GraphAware) & algorithms	Run & check algorithms
Algorithms (Java)	Generate/update recommendations

Table 4.3: Subsystems of the recommender system

4.3 | Data structure

To fill the Neo4j with the current data, a custom ETL script written in Java is used. Java was chosen as a core programming language of this research for three reasons. First, it has all the libraries to connect with both the MongoDB and Neo4j databases. Second, it is the same language as Neo4j is written in. Third, I am proficient in Java. Collection by collection and document by document the script reads the data in the MongoDB database. This is converted based on the chosen logical data model into a query for Neo4j, which will be explained in the following paragraph. Neo4j uses the expressive graph database query language Cypher. Alternatives for Cypher are the more complex query languages SPARQL and Gremlin. Cypher is chosen because it is a declarative language with the expressive power of a property graph. It is also an advantage that Cypher is familiar with SQL. By executing the query, the commands create, read, or update the entities, relationships, or attributes. The ETL script is written solely by the author of this report. The code of this ETL script is available on GitHub[3] under a GNU AGPLv3 open-source license. GitHub is a digital platform for code sharing and collaboration.

The Neo4j graph database exists of two types of properties: entities and relationships. Both the entities and relationships can be categorized with none, one or multiple labels, and hold attributes. These are key-value pieces. The graph structure is intuitive. In a graph database, the logical data model is the same as the physical

[3] https://github.com/part-up/mongo2neo

data model (Lopez, 2015). The data is supplied by the Part-up marketplace. All attributes of the users and teams are extracted together with the relevant events. The logical data model holds all the attributes that need to be extracted from the platform. It is the design of the graph database. Attributes that are expected to be part of this logical data model are shown in figure 4.3. The choice for this structure with these attributes is based on the literature review, constraints, objectives, and previous property graphs designs. There are more attributes available in MongoDB. These are not necessary for the designed algorithms. In later stages, this can be expanded as Neo4j is schema-free. The logical data model is the digital basis of the recommender system. Both from the capabilities and performance point of view it is very important that this artifact is well designed.

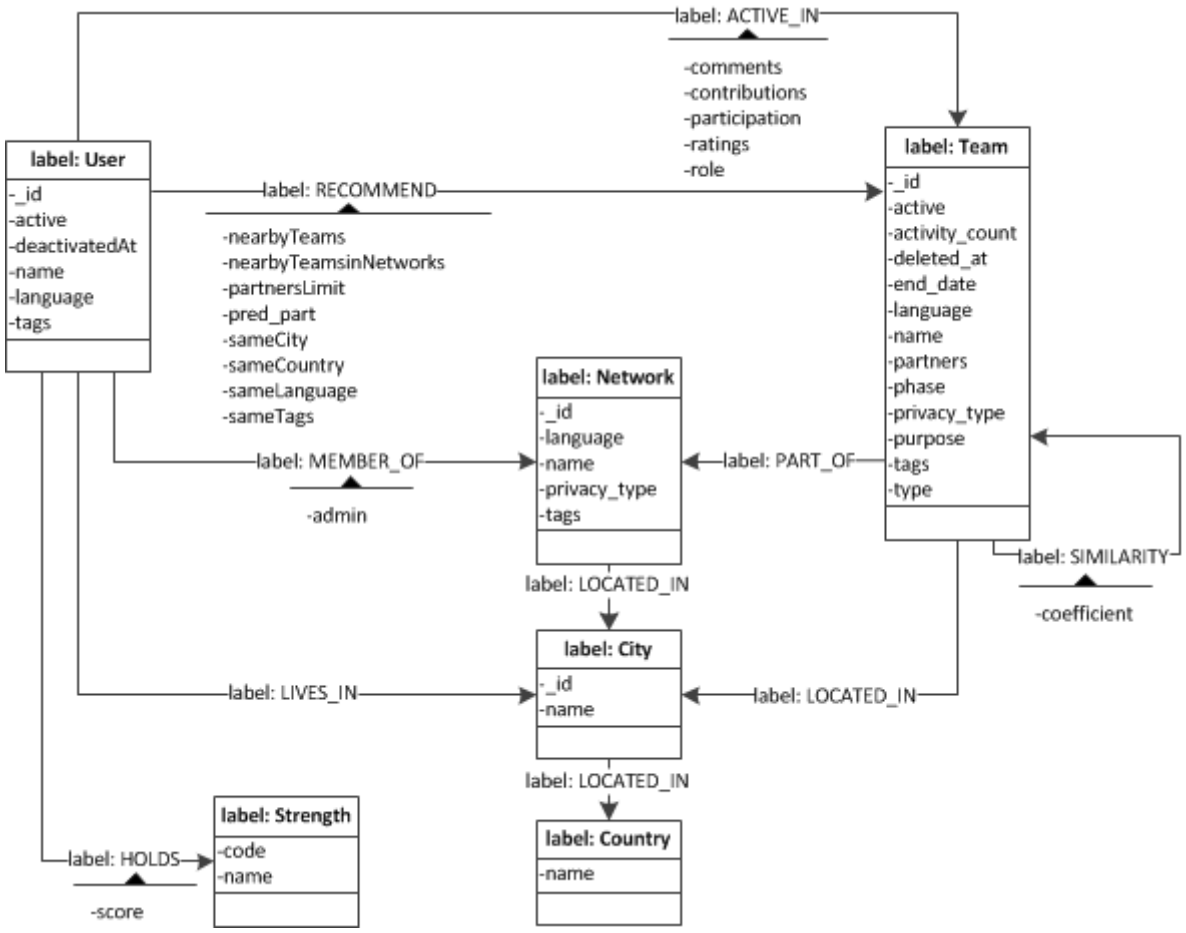

Figure 4.3: Logical data model

Neo4j also automatically provides every entity and relationship with the *'<id>'*-attribute. With this attribute, every entity or relationship can be uniquely identified. This value is not stored in the MongoDB database. That is why this attribute is not used, nor included in figure 4.2. The attributes of the entities are on purpose named the same as in the MongoDB database. Hence the Neo4j convention of CamelCase is not respected for the attributes. For the labels, the Neo4j convention is used. The *'role'*-attribute of the *'ACTIVE_IN'*-relationship is categorized with weights on an ordinal scale as shown in table 4.4. The scale in my general interpretation of a user's commitment to a team. To have a clear distinction between the recommendations of the different algorithms the *'RECOMMEND'*-label is altered during the research. The

label of each recommendation relationship is the concatenation of *'RECO_'* and the code of the algorithm. For example, *'RECO_RAND'*. The extensive data dictionary is included in appendix B.

Role	Value
Creator	2,0
Partner	1,5
Supporter	1,0

Table 4.4: Role of a user in a team.

4.4 | Recommendation framework

The GraphAware plugin extends Neo4j with a framework for data analysis and processing. Part of this framework is the product GraphAware Reco. This is a recommendation framework for Neo4j. The built-in structure for the different algorithm modules and monitoring the process of generating recommendations make it a suitable product for this kind of analysis (GraphAware, 2015). It is even more favorable because it has a high performance and a flexible design. This solution is also scalable because of the feature to pre-compute recommendations that are too complex to compute in real-time. To conclude, GraphAware Reco has two advantages and disadvantages. The advantages are that GraphAware Reco is a developed product and that a manager of GraphAware offered their support to this research (personal communication, 10 September 2015). The disadvantages are that the algorithms must fit the structure of the product and the limited documentation and examples. The efforts to overcome the limitations of the structure can be expected to be smaller than the efforts to develop the necessary functions from scratch. The small knowledge pool can be solved by actively using the offered support of GraphAware. This line of reasoning resulted in the confidence of choosing GraphAware Reco as a recommendation framework.

4.5 | Conclusions

Following multiple methods, the recommender system for this marketplace is designed from existing meta-artifacts and new artifacts. Considering the requirements and technology from the environment, an architecture was selected with a best-of-class chart. The architecture design meta-framework specifies an architecture to generate personalized recommendations for relevant self-managing teams, to support users in team formation. Essential subsystems are the marketplace database, ETL script, recommendation database, recommendation framework, and algorithms. The data is extracted with a Java script from the marketplace database MongoDB and transformed following the designed logical data model before stored in the recommendations database Neo4j. GraphAware Reco was chosen as a recommendation framework, because it is scalable, flexible, a good fit with Neo4j, and well supported. Together it creates the initial version of a recommender system for self-managing team formation.

5 | Design of the recommender algorithm

The core of the recommender system is the algorithm. An algorithm is a collection of rules that are applied sequentially (Cormen et al., 2009). The algorithm for this recommender system must create a filter by awarding a score to several teams. This can also be a combination of algorithms, but for evaluation purposes, each algorithm is designed separately. A recommender system can be driven by several filtering techniques (Melville et al., 2002): collaborative, content-based, demographic, utility-based, and knowledge-based recommendations. Burke (2002) analyzed the five techniques on their background, input, process, advantages, and disadvantages. Collaborative filtering is according to Burke (2002) the most mature technique. For this reason, two variants of this method are chosen: user-based and item-based collaborative filtering. Content-based and demographic filtering are similar techniques. Just like knowledge-based and utility-based are analogous. From each duo the technique that fits the logical data model the best is chosen. Hence, the five algorithms that are developed for the recommender system are:

- Random filtering (RAND) algorithm
- Demographic filtering (DF) algorithm
- Knowledge-based filtering (KBF) algorithm
- User-based collaborative filtering (UBCF) algorithm
- Item-based collaborative filtering (IBCF) algorithm

This chapter explains the design choices of each algorithm in detail. Providing this transparency into the functioning of the algorithm prevents it from becoming a black box (Herlocker et al., 2000). The rationale of each algorithm should be explainable to both experts and users, otherwise, it is according to Holton (2001) difficult to perceive a system that is supported by technology without distrust. Further, Xiang (2011) supports this by noting that providing the context of a recommendation is one of the most important components of every recommender system. This transparency for the experts, product owner, and users contribute to the governance of the architecture. Janssen (2009) identified governance as an indispensable element of the meta-framework for architecture design. For experts and the product owner, this required transparency is ensured by four components. First, by describing the design choices and working of each algorithm. Second, by including multiple Cypher queries are for reproducibility and knowledge-sharing in the report. Cypher is the query language of Neo4j and is used in the algorithms. Third, by applying the ten guidelines for future-proof code by Joost Visser (2015) during the development phase. These guidelines guard the readability of the code. Fourth, by publishing the algorithms on the code-sharing platform on GitHub. The transparency for the users is addressed in the development phase.

5.1 | General algorithm structure

The recommendation framework GraphAware Reco consists of six modules: recommendation engine, config, logger, blacklist, filter, and post-processor. The config, logger, blacklist, and filter modules are always

the same for each algorithm. The config module contains essential information like the maximum number of recommendations. The two loggers present information on the activities by the recommendation framework in the terminal. The RecommendationLogger shows the entity for which the recommendation is generated. The StatisticsLogger shows the composition of the recommendation score. The blacklist and filter modules are the context. This context is determined if the recommendations can be served to a user based on predefined rules within the Part-up marketplace. Each of the five modules that together form the context of Part-up is shortly described in the following paragraphs.

The blacklist module removes results based on user interaction. There is only one blacklist element: ExistingRelationships. This element checks if the user is already a partner or creator of a team because these teams should not be suggested. Supporting of a team does not affect the recommendations list. Even while there already is a relationship, the expectation is that it still makes sense to recommend the team for two reasons. The goal of the recommender system is to suggest teams to become a partner in and not just a supporter of. And Swearingen and Sinha (2001) discovered that the confidence of users in the recommender system increases when some recommendations are familiar. This element is written in Cypher (query 5.1) and runs inside GraphAware Reco.

```
MATCH (u:User)-[r:ACTIVE_IN]->(t:Team)
WHERE id(u)={id} AND r.role>1.0
RETURN t as blacklist
```

Cypher query 5.1: ExitingRelationships element (Blacklist for current relationships).

The filter module applies the marketplace rules to the initial list with recommended teams. There are four filters. The marketplace has a couple of platform rules to offer private teams next to the public teams. The closed teams cannot be suggested to all users. The FilterOutPrivate element inspects if a team can be joined by a user. As shown in table 3.3, there are five privacy options for a team: open, closed, part of an open network, part of an invitational network, and part of a closed network. The privacy restrictions for teams that are part of a network are covered by the FilterOutNetwork element. Cypher query 5.2 only filters out the private teams without a network where the user is not already active in as a supporter. As mentioned, pending invitations for a team are neglected by this filter.

```
MATCH (u:User),
      (t:Team)
WHERE id(u)={id}
      AND t.privacy_type=2
      AND NOT (u)-[:ACTIVE_IN {role:1.0}]->(t)
RETURN t as blacklist
```

Cypher query 5.2: FilterOutPrivate element (filters out private teams).

Teams in a network can only be suggested if this network is open (privacy type 3), or the user is a member of the invitational, or closed network (privacy type 4 and 5). The FilterOutNetwork element verifies this relationship. This is done inside GraphAware Reco with the Cypher query 5.3.

```
MATCH (u:User),
      (n:Network)<-[:PART_OF]-(t:Team)
WHERE id(u)={id}
      AND ( t.privacy_type=4 OR t.privacy_type=5 )
      AND NOT ( (u)-->(n)
      OR (u)-[:ACTIVE_IN {role:1}]->(t) )
RETURN t as blacklist
```

Cypher query 5.3: FilterOutNetwork element
(Filters out private teams of an invitational network or a closed network).

To exclude old teams the FilterOutEnded element was included. The FilterOutDeactivated element was developed to exclude deactivated teams. Each team holds the Boolean attribute *'active'* which can be set to false for deactivation. This is not an automatic process. Thus, teams can have a passed end date but still categorized as active. Both are written in Java. Table 5.1 shows that with these two filters only 287 teams can be recommended.

Active	End date before 27 January 2016	End date after 27 January 2016	Total
True	367	287	654
False	125	35	160
Total	492	322	814

Table 5.1: Division of active and/or ended teams.

In the post-processing module were five elements developed. These post-processor elements influence the generated list with recommendations more gently. Instead of excluding teams, the score of some teams is positively or negatively adjusted. The DF-, KBF- and UBCF-algorithms apply the five post-processor elements. The RAND-algorithm must be strictly random. The IBCF-algorithm cannot use these post-processor elements, because it calculates the initial score of each recommended team differently. All post-processor elements are written in Java. The code is included in each GitHub repository. Table 5.2 presents an overview of the five post-processor elements. The attribute is added to the *'RECOMMEND'*-relationship for the total score. The score can be negative or positive and multiple distributions are supported. Each element is discussed in alphabetical order in the following four paragraphs.

Element	Attribute	Max negative score	Max positive score	Distribution
PenalizeTeamSize	partnersLimit	-5	0	Pareto
RewardSameCity	sameCity	0	5	Degenerate
RewardSameCountry	sameCountry	0	1	Degenerate
RewardSameLanguage	sameLanguage	0	1	Degenerate
RewardSameTags	sameTags	0	50	Uniform distribution
	Total:	-5	57	

Table 5.2: Post-processing score

The PenalizeTeamSize element negatively influences the recommendation of large teams following the mentioned Ringelmann effect. For the team size, only the creator and partners count as team members. The supporters are ignored because they are not active in activities. This element awards a negative score based on a Pareto-distribution starting at five team members with the 80% point at thirteen team members. As presented in table 5.3, 612 (93%) teams have five team members or less and almost all teams have thirteen or fewer team members (645; 98%). The maximum negative score is five points.

1	2	3	4	5	6	7	8	9	10	12	13	15	16	18	19	Total
370	124	57	39	22	11	8	6	6	4	2	2	1	1	2	1	656

Table 5.3: Distribution of team sizes.

There are two geographical post-processing elements: RewardSameCity and RewardSameCountry. If a user and a team are located in the same city the total score is increased with five points. For the same country, one point is awarded, also if the user and team are located in the same city. It should be noted that only 376 (14%) users and 563 (86%) teams hold the geographical information of a city and a country. This strongly limits the applicability of these elements.

The RewardSameLanguage element increases the score with one point when a user and team have the same language. All users and all teams hold the *'language'*-attribute. The user language is chosen by the user. The team language is determined by analyzing the title and description via a Google service. As shown in Table 5.4, are English and Dutch by far the most popular languages in the marketplace. These are also the only two languages that are currently present within both the users and teams.

Language	AF	DE	EN	FI	IT	HR	NL	RU	ZH	Total
Users	1	1	1.222	2	1	0	1.514	1	1	2.743
Teams	0	0	346	0	0	1	309	0	0	656

Table 5.4: Languages of the users and teams.

The RewardSameTags searches for similar tags of a user and team. It was hard to design an effective element to compare the tags because a user is free to choose them. The marketplace does not contain a bag-of-words model. This is a method to steer a user voluntarily towards tags that were already submitted by other users. If the tag differs, the user is still free to submit and pick this new tag. This especially prevents multiple

tags that mean the same but have a couple of different letters. To restrict the development complexity a full string match and not a fuzzy match was chosen. This strongly reduces the chance of a match; however, this specific element was not identified as crucial for this research. For the same reason was synonymy not addressed in this element. This even further decreases the chance of a match between a user and a team based on tags. It should also be noted that half of the users do not even have tags added to their profile (table 5.5). For each tag that matches ten points are awarded. With a maximum score of 50 points. Compared to the other elements this is a very high reward, but as described the chance of an exact tag match is very small. When such a match occurs, this indicates a shared interest.

Tags	0	1	2	3	4	5	Total
Users	1.454	166	60	139	212	712	2.743
Teams	0	25	27	67	126	411	656

Table 5.5: Number of tags of the users and teams.

Algorithms can be executed with a memory-based or model-based approach. The former means that the algorithm is run when the recommendation is needed, while the latter means that the algorithm creates a model from which the recommendation is read. The advantages of memory-based collaborative filtering over its model-based counterpart are according to Breese et al. (1998) the simplicity of implementing this algorithm and the adaptivity of the recommendations as the algorithm runs on the latest data. Although these researchers also point out the disadvantage that running the algorithm for each recommendation individually takes time or large amounts of memory. This lack of model building in memory-based collaborative filtering limits the scalability. That is why this method is considered impractical for large datasets (Linden et al., 2003) and is also the main reason why in the research setup, all algorithms are run with a model-based approach.

5.2 | Random filtering algorithm

The random filtering (RAND) algorithm produces complete random recommendations. Therefore, this RAND algorithm can function as a control algorithm. For collecting the random recommended teams, a Java method of the GraphAware framework is used. The generated list is also not post-processed. Only the blacklist and the four filters are applied. The structure of this RAND algorithm with all its elements is shown in figure 5.1.

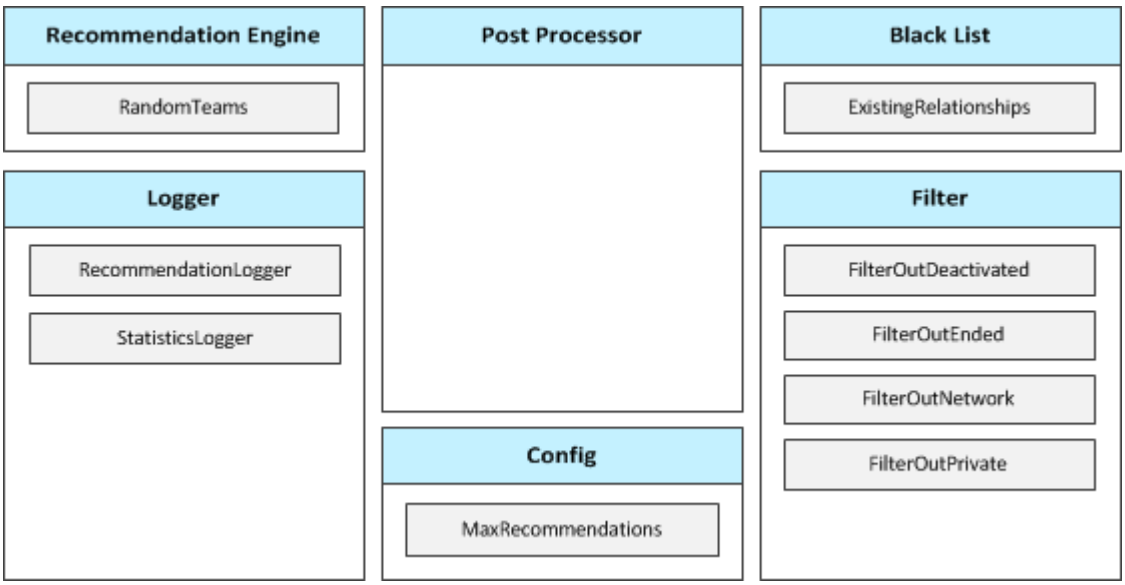

Figure 5.1: RAND algorithm modules in the GraphAware Reco framework.

5.3 | Demographic filtering algorithm

The technique of demographic filtering uses chosen demographic attributes such as language, location, and preferences to filter the available teams. Pazzani (1999) amongst others conclude that demographic information can be used to identify which items a user likes. The recommendation engine of the demographic filtering (DF) algorithm is the same as the RAND algorithm, but by using all described post-processor elements a filter on demographic information is realized. The structure of this DF-algorithm with all its elements is shown in figure 5.2.

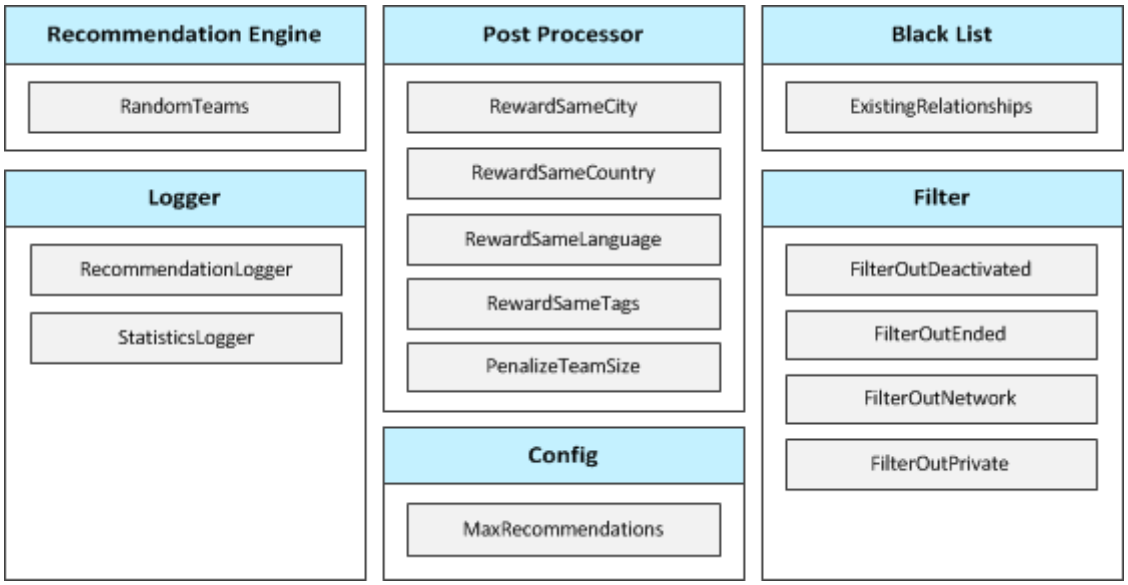

Figure 5.2: DF-algorithm modules in the GraphAware Reco framework.

5.4 | Knowledge-based filtering algorithm

To include soft skills in an information filtering algorithm, a knowledge-based filtering method is chosen. The technique of knowledge-based filtering applies knowledge engineering to the reason which items meet the requirements of a user (Trewin, 2000). This knowledge engineering process involves understanding the specific features of users and teams. Investing in understanding the features and requirements prevents a cold start, but also results in a static recommendation ability (Burke, 1999). This knowledge-based filtering (KBF) algorithm uses the results of a personality test with the soft skills of a user. Human resource specialist Meurs HRM developed a personality test for the Part-up marketplace. Based on ten questions a user receives two out of five available individual strengths: influential strength, managerial strength, mental strength, social strength, and personal strength. As Sinha and Swearingen (2001) argued, in exchange for better ratings, users are willing to fulfill these kinds of system requests for more information but only if the outcome is likely to improve. The structure of this KBF-algorithm with all its elements is shown in figure 5.3.

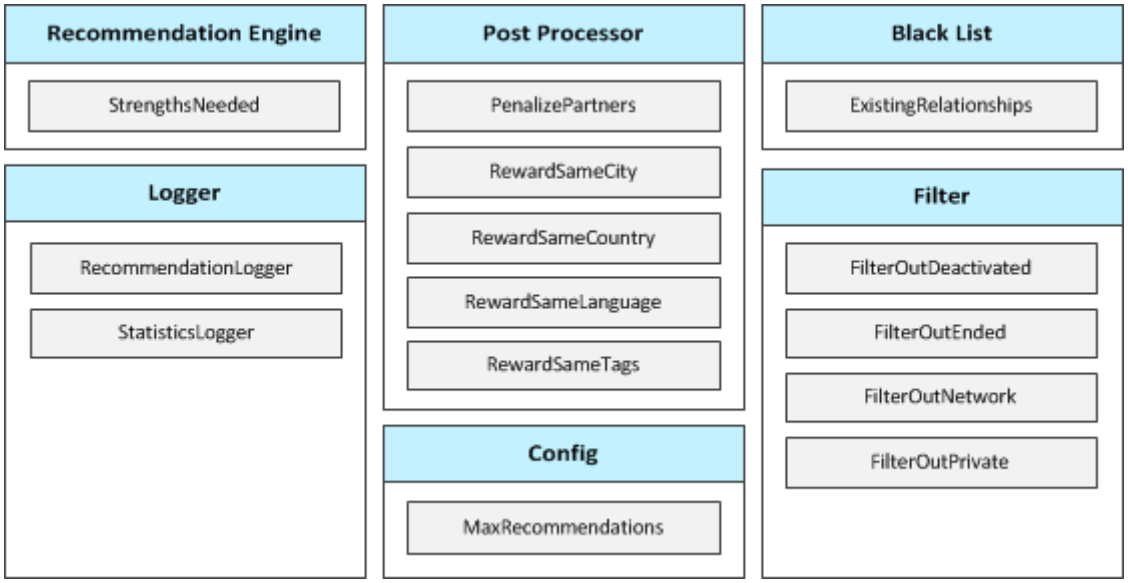

Figure 5.3: KBF-algorithm modules in the GraphAware Reco framework.

The core of this knowledge-based filtering algorithm is the StrengthsNeeded element (Cypher query 5.4). This element functions in two steps. First, the element retrieves the two strengths of the user for which the recommendations are generated. The algorithm completely depends on the identified strengths a user holds. If the user does not hold any strengths, because he did not complete the inquiry, then this algorithm will not function for this user. In the second step for each team, the distribution of the strengths of its members is analyzed. For this initial version of the algorithm, it is assumed that each team strives for an equal distribution of each strength. If one of the user's strengths is underrepresented in a team then the team is recommended. The score of the recommended team depends on the need for a strength. The maximum awarded score for each strength is ten points. That means that this element can recommend a team with a maximum of twenty points. This score can be further increased or decreased by the post-processing elements. Of course, the blacklist and filters are also applied to the generated list of recommended teams.

```
MATCH (u:User)-[:HOLDS]-(s:Strength)
WHERE id(u)={id}
WITH s.name AS userStrength
MATCH (reco:Team)-[r:ACTIVE_IN]-(u:User),
      (u)-[:HOLDS]-(s:Strength)
WITH userStrength,
      reduce(need=0,
            i IN COLLECT(s.name) |
            need +
                  CASE i WHEN userStrength
                        THEN 1
                        ELSE 0
                  END
      )/toFloat(COUNT(s.name)) AS needs,
      reco
WHERE needs<0.2
RETURN reco, (0.2-needs)*50 AS score
```

Cypher query 5.4: StrengthsNeeded element (Core of the knowledge-based filtering algorithm).

The code of this algorithm is published on GitHub under a GNU AGPLv3 open-source license[4].

5.5 | User-based collaborative filtering algorithm

The DF- and KBF-algorithms strongly depend on a couple of chosen attributes. This static structure makes the algorithm domain dependent. As a result, every time the interests of the users change the algorithm needs to be adjusted. In Palo Alto, the researchers at Xerox Parc already faced this challenge in the early nineties. They developed a system called Tapestry that can filter the continuous stream of electronic documents (Goldberg et al., 1992). The annotations of users on these documents are compared to other users to discover similar interests. For this reason, the new approach was named collaborative filtering (Bell & Koren, 2007). Independent of the domain, the purpose of collaborative filtering is to find user-item pairs based on the user's interests and interaction. This replaced straightforward correlation statistics and predictive modeling (Melville & Sindhwani, 2011). The technology was further developed by Amazon for shopping suggestions. Later similar technology was implemented and developed by Facebook, Google, Hulu, LinkedIn, Netflix, and Twitter to support their customers because from a scalability perspective a domain-independent algorithm is preferred. The lack of a domain dependency ensures that this algorithm automatically adjusts to the diverse interests of users, which solves the necessity for a new algorithm each time a new topic is introduced on the platform.

Multiple researchers describe collaborative filtering as one of the most successful filtering methods (Lemire & Maclachlan, 2005; Su & Khoshgoftaar, 2009). It delivers the most complete package with its ability to identify cross-genre niches, the lack of required domain knowledge, quality improvements, and enough implicit feedback. The main advantage of collaborative filtering is that it performs well while a user or team

[4] https://github.com/part-up/KBF-algorithm

holds limited content, but has relationships with other entities (Melville et al., 2002). Koren et al. (2009) agree by stating that technology is domain independent. This content-agnostic aspect means that algorithms can be applied to various subjects without adjustments. A second advantage is that it can deliver serendipitous recommendations. Melville et al. (2002) describe this as relevant recommendations of items (teams), although not matched on content supplied by the user via its profile. Multiple researchers conclude that these two reasons led to the successes of recommender systems in different domains (Goldberg et al., 1992; Resnick et al., 1994).

According to Koren et al. (2009), there are two main areas of collaborative filtering: neighborhood methods and latent factor models. The former focuses on the relationships between users and teams. User-based collaborative filtering is based on this principle. The latter infers a latent variable of a user's appreciation for a team by analyzing observable variables about their activity in that team. Item-based collaborative filtering is a successful implementation of latent factor models. The item-based collaborative filtering algorithm is described in section 5.6.

User-based collaborative filtering is by Wang et al. (2006) illustrated as the *"Word of Mouth"* phenomenon. This is the traditional version of collaborative filtering where team suggestions are generated based on users with similar interests (Breese et al., 1998). Figure 5.4 shows a simplified version of the user-based collaborative filtering approach visualized for users active in teams. To generate recommended teams the algorithm follows a four-step-process. First, for a user, the algorithm lists the teams in which the user is a member. Second, for each listed team, the algorithm collects which team members are also part of that team. Third, for these collected members, the algorithm gathers the other teams in which they are also a member. Finally, these discovered teams are ranked on frequency in the third step. All these users, teams, and relationships are usually merged in a table. However, as figure 5.4 shows, this method is very suitable for path querying in a graph database.

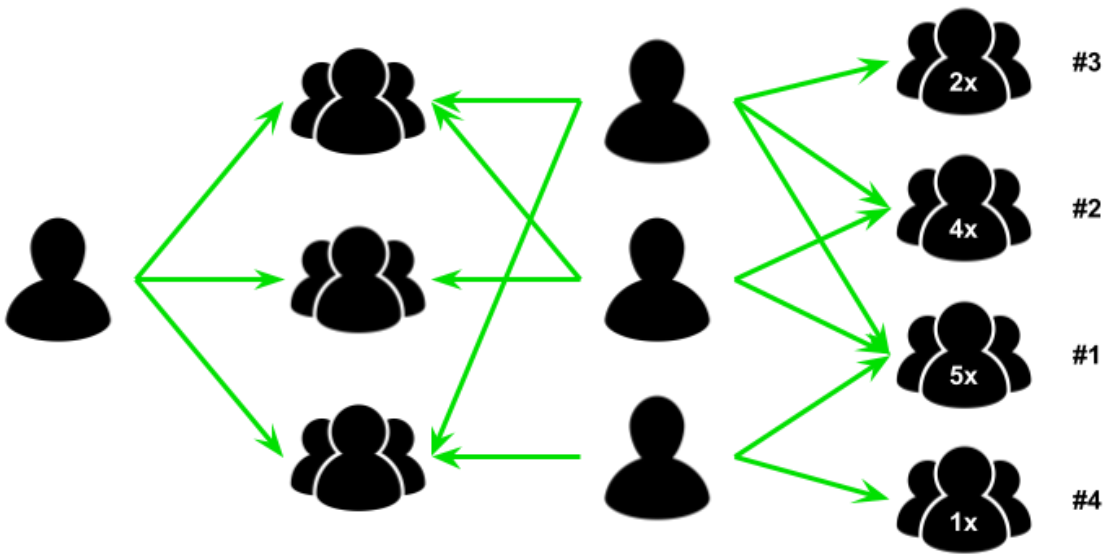

Figure 5.4: User-based collaborative filtering for teams.

The structure of this UBCF-algorithm with all its elements is shown in figure 5.5.

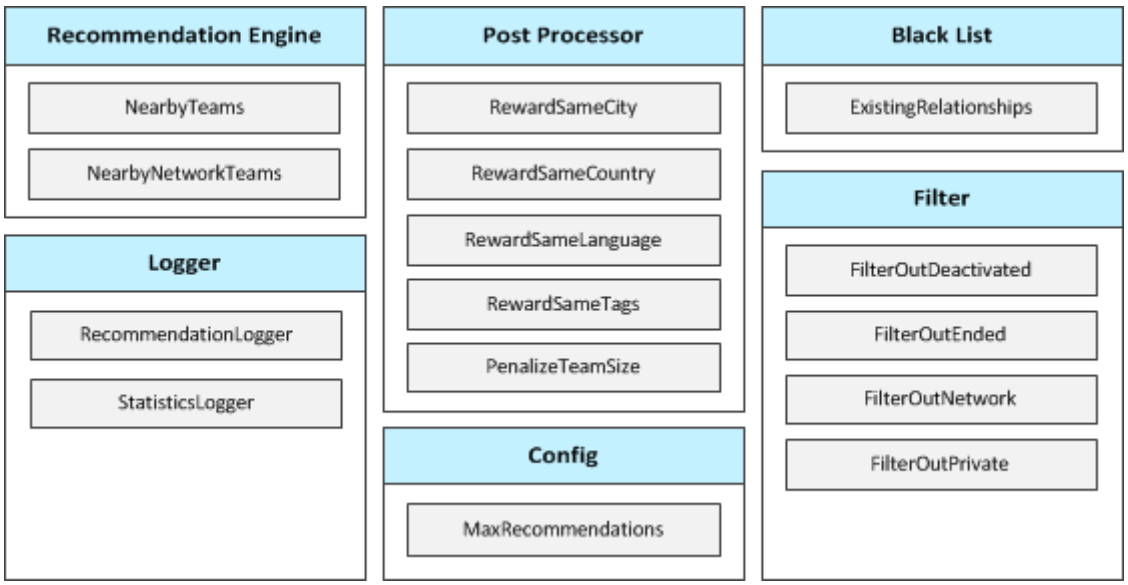

Figure 5.5: UBCF-algorithm modules in the GraphAware Reco framework.

The core of the collaborative filtering algorithm is the NearbyTeams element (Cypher query 5.5). The approach is as shown in figure 5.4. For each discovered path of the three relations, the average of the three roles is awarded to the team. The value can vary between 1 and 2. For example, if the path consists of a creator (2 points), partner (1,5 points) and supporter (1 point) in the two teams then the score is 1,5 points.

```
MATCH                          (u1:User)-[r1:ACTIVE_IN]->(t:Team)
      <-[r2:ACTIVE_IN]-(u2:User)-[r3:ACTIVE_IN]->(reco:Team)
WHERE id(u1)={id}
RETURN reco,
      COLLECT(t._id) as team_id,
      u2._id as partner_id,
      (r1.role + r2.role + r3.role)/3 AS score
```

Cypher query 5.5: NearbyTeams element (User-based collaborative filtering algorithm via teams).

The discovery process of relevant teams for a user by this NearbyTeams element is supported by the NearbyNetworkTeams element (Cypher query 5.6). This second element has a similar approach and analyses in which teams the fellow members of a network are active as a creator or partner. For each discovered path of three relations, a score of 0,5 points is awarded to the team, regardless of the role. This low score is chosen to value a discovered path by the NearbyNetworkTeams element always lower than a discovered path by the NearbyTeams element. As the similarity between users who are active in the same network is assumed to be lower than users who are active in the same team.

```
MATCH (u1:User)-[r1:MEMBER_OF]->(n:Network)
      <-[r2:MEMBER_OF]-(u2:User)-[r3:ACTIVE_IN]->(reco:Team)
WHERE id(u1)={id} AND r3.role>=1.5
RETURN reco,
      n._id as network_id,
      u2._id as member_id,
      0.5 AS score
```

Cypher query 5.6: NearbyNetworkTeams element (User-based collaborative filtering algorithm via networks).

The code of this algorithm is reviewed by a senior data scientist of GraphAware and a platform architect of Part-up. The code of this algorithm is published on GitHub[5] under a GNU AGPLv3 open-source license.

5.6 | Item-based collaborative filtering algorithm

The most complex algorithm being designed in this research is the item-based collaborative filtering algorithm. It was chosen to resolve the scalability challenge, which is a well-known issue of the user-based collaborative filtering algorithm (Sarwar et al., 2001). Linden et al. (2003) addressed especially this problem while designing the item-based collaborative filtering algorithm for Amazon. This collaborative filtering technique focuses on the rating distributions per item as usually, a digital marketplace holds more users than items. It scales independently of the number of users or items on a platform by building a similar-items table. As the user rating of an item is expected to be stable, this model-based algorithm can create a model of the database. Accordingly, the algorithm does not have to identify the neighborhood of similar users first. As a result, the recommender system can produce much faster results, note Sarwar et al. (2001). In this context of team formation an item references to a team.

The item-based collaborative filtering algorithm requires a valuation of a team by a user. For a recommender system, the relationship between a user and an item is defined by several attributes. The input data of the user on a team can be split into two types: explicit feedback and implicit feedback (Koren et al., 2009; Xiang, 2011). The explicit feedback is the most valuable one because this is the rating a user gives to a team. However, the availability of this information is low, as users give only ratings to fellow team members after they finished a collaborative task. This is usually at the end of a team's lifespan. Implicit feedback is, for example, browsing history, mouse movements, page visits, search patterns, or other interactions. This user preference can be modeled as a pseudo-rating with a formula. Ensuring that the outcome of the formula is quite static and robust were two additional design goals. Otherwise, every change in the database needs a wide recalculation of the participation-values.

For the Part-up marketplace, a participation formula (formula 5.1 & Cypher query 5.7) was developed that combines four attributes from the logical data model: ratings, role, contributions, and comments. The composition of the participation formula is mainly based on experiences from the Part-up environment gained

[5] https://github.com/part-up/UBCF-algorithm

during the relevance cycle, combined with literature from the knowledge base gained during the rigor cycle. The ratings a user receives on his activities in a team by another user is the most valuable information, as this is the only available explicit feedback on the quality of a user's activities in a team. When a user receives high ratings, he is actively participating in that team. Each rating is divided by twenty to transform it to a scale of one to five. Next to these available explicit ratings also an implicit rating is constructed based on three attributes that every user-team-relation contains: role, number of contributions and number of comments. The role a user holds in a team is expressed following the ordinal scale from table 4.4 (creator: 2 points, partner: 1.5 points, supporter: 1 point). The number of contributions the user has in the team is divided by the maximum number of contributions this user has in all its teams. The same calculation is chosen to normalize the number of comments. Note that the normalization is applied with a smoothing term to prevent the dividing by zero when a user has no contributions or comments. The normalized contributions are weighted double compared to the normalized comments because a contribution is a commitment to a team and scarcer than a comment. The implicit rating is the sum of the role-weight, normalized contributions and normalized comments and results in a value between one and five. The average of these explicit ratings and the implicit rating is the participation score of a user in a team. That means that when the number of explicit ratings increases, the influence of the implicit rating on the participation score directly decreases. A high participation score implies that the user is interested in the team. More design cycle iterations with additional evaluation and field testing of the formula are necessary to optimize these participation scores. With this formula, the similarity between teams can be calculated to perform item-based collaborative filtering.

$$R_u = \frac{\sum_{i=1}^{ratings} \frac{ratings(i)}{20} + role + \frac{contributions}{maxContributions} \cdot 2{,}0 + \frac{comments}{maxComments}}{ratings + 1}$$

Formula 5.1: Participation formula.

```
MATCH (u:User)-[r:ACTIVE_IN]->(t:Team)
WITH MAX(r.contributions) AS maxContributions,
      MAX(r.comments) AS maxComments, u
MATCH (u)-[r:ACTIVE_IN]->(t:Team)
WITH
r.role+(r.contributions/(toFloat(maxContributions)+0.00001)*2.0
)
      +(r.comments/(toFloat(maxComments)+0.00001)*1.0) AS part,
      r
SET r.participation=((REDUCE(avg=0, i IN r.ratings | avg +
(i/20)))+part)
      /(LENGTH(r.ratings)+1)
```

Cypher query 5.7: Participation formula.

With these participation scores, a user-team matrix is constructed. The first step is the analysis of the user-item matrix on similarity as presented in figure 5.6. The similarity between the two teams is determined by comparing the participation scores in those teams. For the core analysis, only the participation scores of the users who are active in both teams are used. Otherwise, it is not possible to compare the similarity.

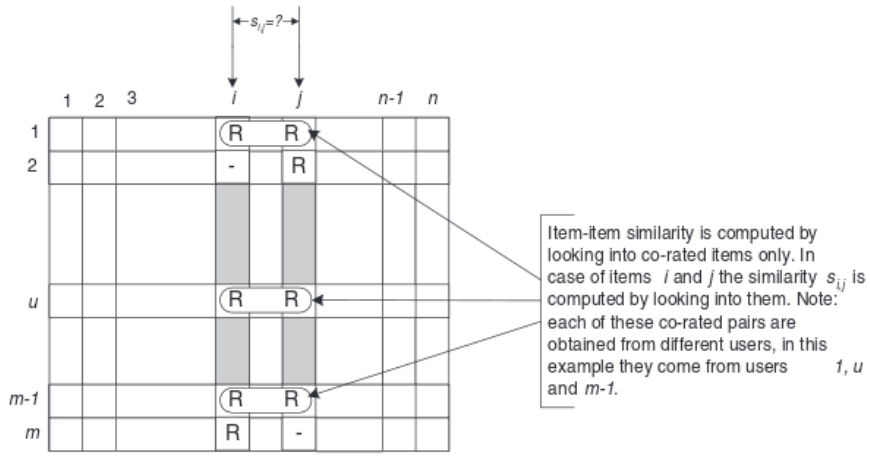

Figure 5.6: Item-based similarity (Sarwar et al., 2001).

Sarwar et al. (2001) advise three types of similarity methods to compare items: cosine-based similarity, Pearson correlation-based similarity or adjusted cosine similarity. The first option is cosine-based similarity (formula 5.2). This similarity formula is also known as vector-based similarity because this similarity is the cosine angle between the vectors. It is the simplest approach for a mathematical similarity comparison.

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

Formula 5.2: Cosine-based similarity (Sarwar et al., 2001).

The second option is the Pearson correlation-based similarity (formula 5.3). An improved version of the cosine-based similarity formula that normalizes the result with the average rating of all users on a team, not just the ratings of the users that are active in both teams. This normalization is advantageous in this research on team formation, as it corrects the valuation of younger teams with relatively low participation scores. This self-damping effect improves the accuracy of the algorithm. However, this extra calculation also affects performance.

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R}_i)(R_{u,j}-\bar{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_i)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_j)^2}}$$

Formula 5.3: Pearson correlation-based similarity (Sarwar et al., 2001).

The third option to determine the similarity is the adjusted cosine similarity (formula 5.4). This is an improved version of the cosine-based similarity. It corrects for a user's rating scheme by normalizing the rating on a team with the average of all ratings that the user has given. Some users have a rating scheme in which they consistently rate items lower or higher. This additional normalization is not advantageous for this research on team formation, as the primary participation value is an implicit rating. Accordingly, it does not need to be corrected for a rating scheme. If the participation score is not a good representation of the activities by all users in a team, then it must be corrected by the participation formula and not the similarity method. For example, by adjusting the chosen attributes or their weights.

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_u})(R_{u,j} - \bar{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R_u})^2}}$$

Formula 5.4: Adjusted cosine similarity (Sarwar et al., 2001).

Taking the arguments on the normalization of the participation scores into account the Pearson correlation-based similarity is chosen. This Pearson correlation-based similarity is applied to the requirement that there is a minimum of three users to determine the similarity. The reason for this threshold is quality and accuracy (Herlocker et al., 2002). Multiple users ensure that there are enough scores available to normalize and compare. There is no maximum in the number of similarity relationships that a team can hold with the other teams. Between two teams there can only be one similarity relationship. This means that every time the similarity scores are recalculated, the previous score is replaced. The chosen similarity formula including the additional rules is transformed into Cypher query 5.8 to create the *'SIMILARITY'*-relationships in the Neo4j database.

```
MATCH (t1:Team),
      (t2:Team)
WHERE t1<>t2
MATCH (t1)<-[r:ACTIVE_IN]-(u:User)
WITH toFloat(AVG(r.participation)) AS t1Mean, t1, t2
MATCH (t2)<-[r:ACTIVE_IN]-(u:User)
WITH toFloat(AVG(r.participation)) AS t2Mean, t1Mean, t1, t2
MATCH (t1)<-[r1:ACTIVE_IN]-(u:User)-[r2:ACTIVE_IN]->(t2)
WITH toFloat(SUM((r1.participation-t1Mean)*
      (r2.participation-t2Mean))) AS numerator,
      toFloat(SQRT(SUM((r1.participation-t1Mean)^2)) *
      SQRT(SUM((r2.participation-t2Mean)^2))) AS denominator,
      t1, t2, COUNT(r1) AS r1Count
WHERE denominator<>0 AND r1Count>2
MERGE (t1)-[q:SIMILARITY]-(t2)
SET q.coefficient=toFloat((numerator/denominator))
```

Cypher query 5.8: Pearson correlation-based similarity

The model learning stage is followed by the recommendation stage. This model is used to make predictions (Breese et al., 1998). The user gets teams suggested that are like teams the user has interacted with earlier. This prediction is either a weighted average or a regression according to Sarwar et al. (2001). For this research, the weighted average (formula 5.5; Cypher query 5.9) was used following the objective of easy to set up.

$$P_{u,i} = \frac{\Sigma_{all\ similar\ teams,N}(s_{i,N} * R_{u,N})}{\Sigma_{all\ similar\ teams,N}(|s_{i,N}|)}$$

Formula 5.5: Weighted average

```
MATCH (u:User)-[a:ACTIVE_IN]->(t:Team)
      -[s:SIMILARITY]-(reco:Team)
WHERE id(u)={id}
WITH AVG(a.participation*s.coefficient) AS score, reco
RETURN score, reco
```

Cypher query 5.9: Weighted average

A simplified overview of item-based collaborative filtering incorporating the described formulas is shown in figure 5.7. Some user-team pairs were left out and only one predicted participation score was included to ensure an uncluttered figure. The green arrows and corresponding values are the participation scores of user-team pairs. The blue arrows and values represent the Pearson correlation-based similarity of two teams. The weighted average formula uses this information to predict a user's participation score in a team. The red arrow with the value is this predicted participation score. A high predicted participation score is a high likelihood for this user to actively participate in a team. The teams with the highest predicted participation scores are recommended to the user.

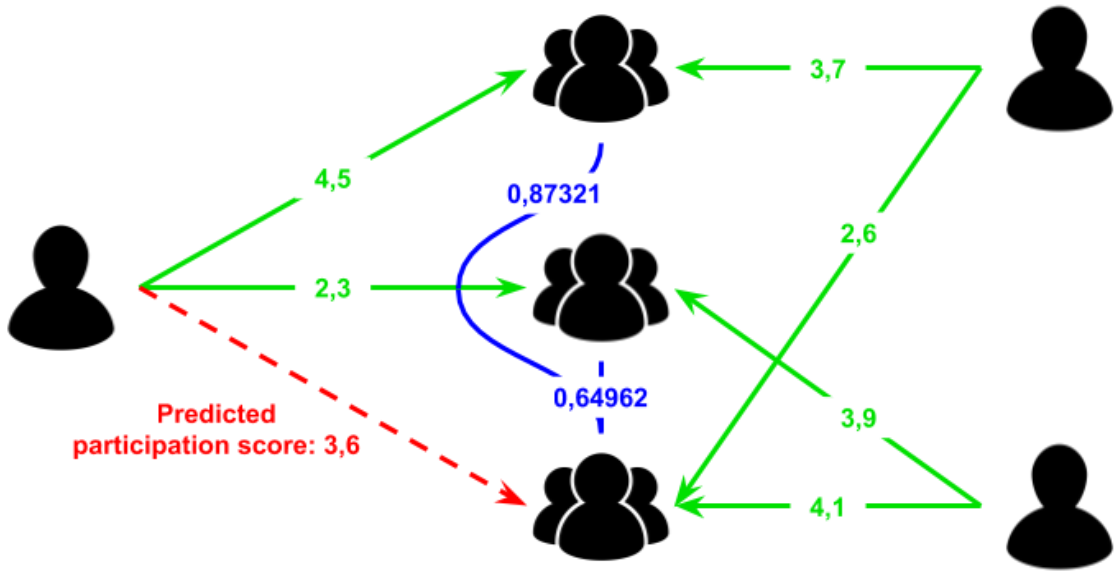

Figure 5.7: Item-based collaborative filtering (Some relationships are left out for overview reasons).

The item-based collaborative filtering approach has four disadvantages. First, when an entity changes the whole model needs to be updated. Shani and Gunawardana (2011) already mentioned adaptivity. Users expect that recent activity is also used for generating recommendations. Although, this does not necessarily require instant update of the model. This can be done with a daily routine, for example in the night when the servers have overcapacity due to low web traffic. In this offline research, the data does not change. Thus, updating the model is not a major disadvantage.

Second, calculating the similarities between teams is complex. For this similarity, it is necessary to analyze the activity of multiple users in multiple teams. This requires a framework with computational power that can run the chosen similarity formula for all teams. During this offline research, the similarity can be calculated by the ETL-script after all data is stored in Neo4j, as the data does not change in the research setup. In an online situation, this similarity-formula can be run within GraphAware Reco or by an external script that accesses Neo4j. This latter option is better from a scalability point of view.

Third, with data sparsity the system performance is weak. This challenge was mentioned earlier and applies to multiple algorithms, but an IBCF-algorithm specifically suffers from it. To run an IBCF-algorithm both the user and its teams must hold multiple relationships. Just one relationship is not enough to generate relevant recommendations. This requirement strikes out many users, but not all users. For users with many data points, this technique can be advantageous. Analyzing more data can result in better recommendations. For using it in a marketplace it is necessary to combine it with other algorithms with a lower data threshold into a hybrid algorithm.

Fourth and finally, item-based collaborative filtering with a model-based approach was patented by Amazon (Linden et al., 2001a) in the United States of America until 18 September 2018. This restricted the use and sale of a product containing this patent in the USA. However, this did not imply that it was also protected in Europe. Therefore, Amazon had also filed a copy to the European Patent Office (EPO). The European version of this patent (EP1121658) by the same authors (Linden et al., 2001b) was rejected by the EPO. The discrepancy between the EPO and Amazon is the result of two issues. First, the data preparation in the IBCF-algorithm is based on existing mathematical formulas. But the process of generating recommendations is not recognized as a mathematical method by the EPO. This EU organ characterizes this method as a marketing strategy and thus non-technical, while Amazon claims that it is an engineering strategy. Second, the division in online and offline tasks is seen by the EPO as an obvious manner to solve a technical problem. The decision of the EPO was under appeal by Amazon (T1056/11) until the patent expired in the USA in September 2019. The EPO ceased the case. Thus, the former patent is now free to use. As described are none of the four mentioned disadvantages of item-based collaborative filtering a showstopper. The structure of this IBCF-algorithm with all its elements is shown in figure 5.8.

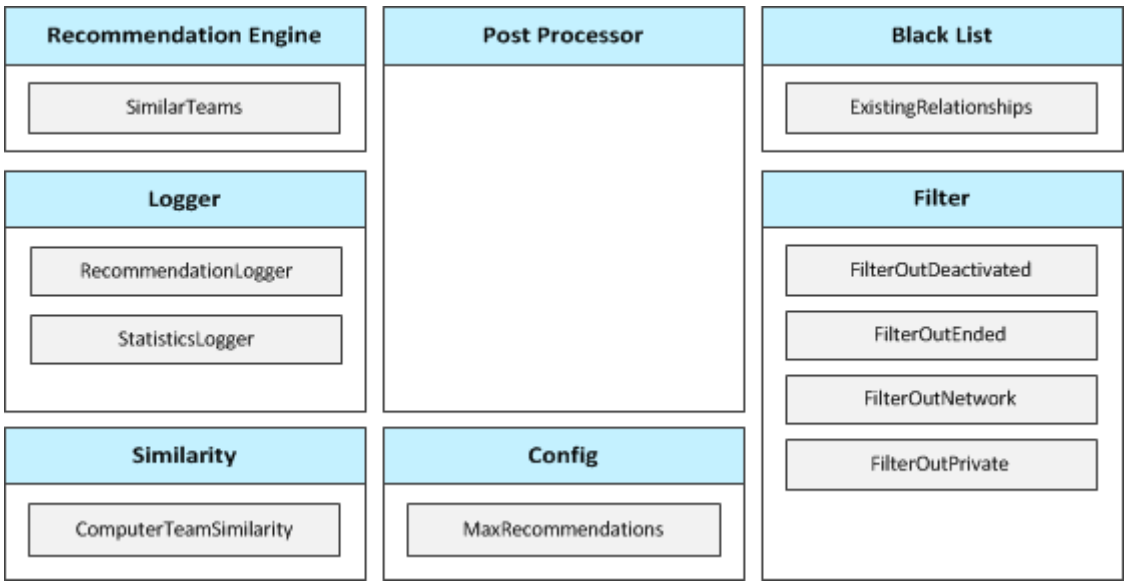

Figure 5.8: IBCF-algorithm modules in GraphAware Reco framework

The code of this algorithm is published on GitHub[6] under a GNU AGPLv3 open-source license.

5.7 | Conclusions

As described, there are multiple information filtering techniques to apply to this case of self-managing teams. It has been mentioned that there is little research in these fields. That is why these algorithms first must be tested with validation experiments. Although some algorithms are only slightly different, it can still lead to different results. Hence, it is important to compare multiple algorithms. Table 5.6 shows the advantages and disadvantages of the four algorithms. Random filtering is not covered by this comparison as it operates as a control algorithm.

[6] https://github.com/part-up/IBCF-algorithm

Algorithm	Advantage	Disadvantage
Random filtering	(control)	(control)
Demographic filtering	No cold start	Domain dependent Static profile Only demographic values included
Knowledge-based filtering	Based on soft skills (personality test) Cold start mitigation	Domain dependent Static profile Perfect composition per team unknown
User-based collaborative filtering	Domain independent Performs well with entities that have relationships with other entities, even with limited entity content Dynamic profile with team frequency Serendipitous recommendation	Cold start (data sparsity) Unstable rating
Item-based collaborative filtering	Domain independent Scales independently Detailed, dynamic profile with participation score Stable rating	Frequent update of the full model Complex similarity calculation Cold start (data sparsity) Formerly patented

Table 5.6: Algorithm comparison.

The demographic filtering and knowledge-based filtering algorithms have the advantage that their approach does not suffer from a cold start or mitigates this. The other two collaborative filtering algorithms require user activity before they can generate any recommendations. Both the demographic and knowledge-based filtering algorithms are domain-dependent, which means that they do not automatically adapt to changes in the interests of users. An aspect that is successfully addressed in the design of the collaborative filtering algorithms. These algorithms also have dynamic profiles based on either the team frequency for user-based filtering or the participation score for item-based filtering. User-based collaborative filtering performs well with teams that hold little information but have several active team members. As a result, this algorithm generates serendipitous recommendations. If the team is just created it suffers from the cold start problem, hence it is hard to recommend. Item-based collaborative filtering is expected to be a stable predictor of a user's interest in a team. The stability of this rating is essential for scaling a collaborative filtering algorithm for a large volume of users. Otherwise frequent updates of the full model are required. This requires calculating the team similarity, a complex and thus resource-intensive process. This comparison shows that all algorithms have different advantages and disadvantages. However, none of the algorithms is unsuitable or a clear winner based on this comparison. In the evaluation, the algorithms are further and extensively compared.

6 | Design evaluation

The evaluation of the five algorithms is, as described in chapter 3. An expert on human-centered data science advised validating the recommender system on different aspects that combined ensure the correct functioning (personal communication, 3 December 2015). Shani and Gunawardana (2011) name twelve dimensions to validate a recommender system: adaptivity, confidence, coverage, diversity, novelty, privacy, risk, robustness, scalability, serendipity, trust, and utility. These are expanded and categorized by Avazpour et al. (2014). The evaluation should always be a combination of multiple dimensions as they state: *"Just one dimension and metric for evaluating the wide variety of recommendation systems and application domains is far too simplistic to obtain a nuanced evaluation of the approach as applied to the domain"* (Avazpour et al., 2014, p.2). The categorized list of all these evaluation dimensions is shown in table 6.1.

	Recommendation-centric	User-centric	System-centric	Delivery-centric
Selected	Confidence Coverage	Trust Utility	Privacy Robustness Scalability	User preference
Not selected	Correctness Diversity	Novelty Risk Serendipity	Adaptivity (Learning rate) Stability	Usability

Table 6.1: Dimensions of evaluation, selected for the research
(Shani and Gunawardana, 2011; Avazpour et al., 2014).

Each category of evaluation dimensions is described in one sentence. The recommendation-centric category includes the dimensions that can be used to assess the generated list with recommendations. The user-centric category holds five dimensions that evaluate if the recommendations fulfill the target end-user needs. The system-centric category ignores the recommendations quality or user-perspective but reflects on the technology and system behind the results. The delivery-centric category is the functioning of the recommender system in its context of use. For each category at least one dimension is chosen (table 6.1).

The adaptivity, correctness, diversity, novelty, risk, serendipity, stability, and usability dimensions are not used for the following reasons. The adaptivity of the recommender system is expressed with the learning rate to incorporate new information in the recommendations list. The algorithm needs to be adaptive to changes in the user's interests, the user's profile, or team collection. This is a trade-off between the speed of shifts in trends and accuracy. This can be measured by determining the time that is needed to process the information before the algorithm can perform its analysis, which falls outside the scope of this research. Correctness influences the value of the recommender systems. The definition of correct is when the recommended team is affiliated with the intentions, interests, or applications of the user. This can be validated with a user interview, but this dimension was dropped in favor of trust, user-preference, and utility which combined do quite the same.

Diversity is the dissimilarity of the items in a recommendations list. According to Avazpour et al. (2014) in some cases generating a recommendation list with similar items is not functional. Users prefer

recommendations that reflect the diversity of the system. Diversity can be considered as an enhancement of the algorithm (Castells et al., 2015). This will not be a requirement for the current research. However, it is an interesting direction for future research within this subject. Novelty is the difference in user experience between past and present recommendations (Castells et al., 2015). In this offline research with a frozen dataset, it is not possible to test the recommendations on their novelty. That is why the novelty dimension is not included in this evaluation. Risk is connected to the expected utility variance. This is something that must be tested over a longer period and is therefore not part of this evaluation.

Serendipity involves including attractive, surprising, and unknown items to reveal unexpressed user's interests (Ge et al., 2010). A recommendation is only serendipitous when it is picked by a user. It negatively impacts accuracy, but it can also be an added value. That is why accuracy and serendipity need to be balanced. This dimension is hard to test. Stability is the consistency of the generated recommendations list over time. As this design research focuses on the definition and development phases it is not considered. The usability is defined by the convenience of adopting the recommender system in an appropriate way for the users. Guidelines for this dimension are effectivity, efficiency, and user satisfaction. This is strongly related to the user interface. This includes the forms that need to be filled in to generate recommendations, the option to easily explore target items, and the generation rate of the recommendation list as perceived by the user. The evaluation of this dimension is not possible as the recommender system still must be integrated into the Part-up marketplace.

This chapter structured the evaluation phase into three experiment types: quantitative validation, user interviews, and qualitative validation. Together these experiments validate the recommender system with its five custom filtering algorithms on eight dimensions: confidence, coverage, privacy, robustness, scalability, trust, user preference, and utility.

6.1 | Quantitative validation

The quantitative validation analyses the recommendations from a numerical point of view. This validation consists of the coverage and privacy dimensions. To validate the algorithms on these dimensions multiple Cypher queries were written for this specific logical data model. Without this artifact, the quality and privacy of an algorithm could not be guaranteed. These queries can be run in the simple web-interface of Neo4j or with code by using its Application Programming Interface (API), a software connection. In the web-interface, the queries are stored in a *'Validation'*-folder. This ensures that it is easy to do routine governance audits on the live database. All validation queries are included in appendix C.

Table 6.2 presents three metric criteria for quantitative validation. The category relates back to the dimensions of evaluation from table 6.1. The goal value is the desired outcome value for the user space coverage and items space coverage and the required outcome value for the privacy criteria. The unit column is the measurement unit of the goal value.

Criteria	Category	Type	Goal	Unit
User space coverage	Recommendation-centric	Quantitative	100	% (users with recommended teams / total users)
Item space coverage	Recommendation-centric	Quantitative	90	% (recommendable teams / total teams)
Privacy	System-centric	Quantitative	0	recommended teams that should not be accessible

Table 6.2: Quantitative validation criteria.

Coverage is the proportion of available data that can be used for generating recommendations. It is an indication of the quality of the recommendations because according to Shani and Gunawardana (2011) recommender systems that cover a wider range of users or teams are preferred. When a user or team holds insufficient information, it is impossible to project a recommendation. In technical terms, this is called a cold start. This is related to the low accuracy in the prediction. The coverage dimension can be split into two types. User space coverage (prediction coverage) is the percentage of users that a recommender system can generate predictions for. Item space coverage (catalog coverage) is the percentage of items (teams) that are recommended to a user by the system. In this validation the item-space coverage is less important for two reasons: first, this research is from the user perspective, and second, the configuration is set to a maximum of five recommendations for each user. The maximum of 13.715 recommendations is not necessarily equally distributed over the 287 active teams that have an end date after 27 January 2016. This is the result of the unequal distribution of user activity in teams and the privacy restrictions of teams. Both the user-space coverage and item-space coverage are used in this quantitative validation.

First, the user-space coverage of the 21 interviewed users is discussed as shown in table 6.3. Each algorithm column shows the number of users in each group that have a recommendation by this algorithm. The maximum count is seven, as there are seven users in each group. The last column ‘Total’ reflects the coverage of all algorithms. That the RAND-algorithm scores best is no surprise because the random-aspect requires no specific data points to suggest a team. That should be the same for the DF-algorithm. Early in the interview process, a small error was discovered with this algorithm. Due to a configuration error, this algorithm-generated no recommendation for one user. This was adjusted after the interview. Therefore, the coverage of the DF-algorithm is one case smaller than the RAND-algorithm.

User group	RAND	DF	KBF	UBCF	IBCF	Total
High	7	6	7	7	7	7
Medium	7	7	4	7	7	7
Low	7	7	2	4	0	7
Total interview group	21	20	13	18	14	21
	100%	95%	62%	86%	67%	100%
Total database	2.475	2.089	338	1.514	764	2.714
	90%	76%	12%	55%	28%	99%

Table 6.3: User-space coverage of the algorithms.

Second, the user-space coverage is validated for all users in the database with the active status. This is the standard status for all users and implies that the user is not deactivated on request. It does not represent the activity level of a user. The results are included in the bottom rows 'Total database' of table 6.4. The RAND-algorithm has generated recommendations for 90% of the users, which is the user-space coverage. It is remarkable that it does not have 100% coverage, as the algorithm requires no specific attributes or relationships. This must be an error in running the algorithm. The same counts for the DF-algorithm, which only has a 76% user-space coverage. The KBF-algorithm has a 12% user-space coverage which is very low. The algorithm requires that a user has discovered his two strengths via a small questionnaire. The questionnaire is voluntary and not well visible on the marketplace. This algorithm has 100% user-space coverage for the users who have completed this questionnaire. The UBCF-algorithm has 55% user-space coverage. Although it has 97% user-space coverage for the users that are active in a team or member of a network this should be 100%, an indication of a small error. The IBCF-algorithm has 28% user-space coverage - quite a low score. The algorithm has 87% user-space coverage for the users that are active in a team with a similarity relationship to another team. Reflecting on the user-space coverage for the algorithm, the conclusion is that currently none of the algorithms can be used for the Part-up marketplace. The user-space coverage of the KBF- and IBCF-algorithms is too low. And the RAND-, DF-, UBCF- and IBCF-algorithms seem not to be running error-free. There is room for improvement. Combining multiple algorithms into one hybrid algorithm can be the solution. Overall 99% of the users have at least one recommendation generated by one of the algorithms. The user-space coverage confirms that with a combination of multiple algorithms nearly all users receive recommendations.

The results of the item-space coverage validation are included in table 6.4. Some users had multiple recommendations generated by the RAND- and DF-algorithm, because of configuration issues. It was not possible to run the algorithms at the same time, so each algorithm ran once in sequence. Each user can have a maximum of five recommendations. As mentioned in chapter 3 are these recommendations not necessarily evenly distributed. This explains why only three-quarters of the teams were recommended via the algorithms. This is a satisfying percentage.

Items	RAND	DF	KBF	UBCF	IBCF	Total
Total	179	173	48	141	65	211
	62%	60%	17%	49%	23%	74%

Table 6.4: Item space coverage of the algorithms.

Privacy is an important dimension if the service processes the personal data of users. The risk is that the privacy of these users and their teams is harmed. The external privacy is not validated in this research. The Dutch Data Protection Authority drafted guidelines to guarantee that personal data of users is protected. All organizations in the Netherlands that store or process personal data must comply with the seven requirements of the National Personal Data Protection Law ('Wet bescherming persoonsgegevens'). Part-up follows a privacy policy[7] that covers these requirements. This validation is on internal privacy. The digital marketplace incorporated platform rules in order to enable private teams next to the public teams. The privacy of the teams is divided into five types (table 6.5). Each team holds a 'privacy type'-attribute with an integer value. Most teams are public and not part of a network, meaning they are type 1. Some teams are private and not part of a network (type 2). These teams can only be accessed by invited users. Pending invitations are for complexity reasons not included in this initial version. That means that these private teams (type 2) can only be recommended to their supporters. There are also public teams that might be part of a network, which are classified as type 3. The same accounts for private teams in an invitational network (type 4) and private teams in a closed network (type 5). The public teams can be recommended to all users irrespective of whether the teams are part of their network. This means that only the teams with the privacy types 2, 4 and 5 are validated on privacy. This dimension is measured by the number of recommended teams that should not be accessible.

Team privacy type	Network privacy type	Description	Number
1	-	Public teams	357
2	-	Private teams	37
3	1	Public teams in a public network	193
4	2	Private teams in an invitational network	174
5	3	Private teams in a closed network	55
		Total:	656

Table 6.5: Privacy types of teams.

For the validation of the privacy and platform rules a series of experiments were performed. There are no teams recommended in which a user was already active as a partner or creator. The recommendable distinction in table 6.6 covers again the teams which are not deactivated and have an end date after 27 January 2016. The validation confirmed that only the 287 teams with these two criteria are recommended and there are no incorrect recommendations of private teams that are not part of a network. However, during the validation, it was also discovered that not all relationships were created correctly. A total of sixteen teams were created

[7] privacy.part-up.com

without the network they are part of. This error is the result of an incomplete ETL script. Only the network teams with location information were imported correctly. The network teams without location information were imported, but without the relationship to the network they are part of. As a result, eleven teams with the privacy type 4 or 5 were not filtered out by the FilterOutNetwork element. This resulted in 576 possible incorrect recommendations, which poses a privacy risk. This is 73% of the total 785 recommendations for teams with the privacy type 4 or 5.

Team privacy type	Network privacy type	Description	Recommendable teams	Recommendations	
				Possible incorrect	Total
1	-	Public teams	135	-	7.841
2	-	Private teams	8	0	2
3	1	Public teams in a public network	65	-	3.405
4	2	Private teams in an invitational network	58	454	604
5	3	Private teams in a closed network	21	122	181
		Total:	287	576	12.033

Table 6.6: Privacy validation of all teams.

To still check the functioning of the FilterOutNetwork element a second validation round was performed (Table 6.7). Now only the recommendations of the correctly imported teams were reviewed. This validation shows that there were no incorrect recommendations. The FilterOutNetwork element functions as desired when the necessary information is included in the Neo4j-dataset. It can be concluded that the privacy of the teams and the platform rules are respected with the five context (blacklist and filter) elements.

Team privacy type	Network privacy type	Description	Recommendable teams	Recommendations	
				Incorrect	Total
1	-	Public teams	135	-	7.841
2	-	Private teams	8	0	2
3	1	Public teams in a public network	60	-	3.237
4	2	Private teams in an invitational network	49	0	150
5	3	Private teams in a closed network	19	0	59
		Total:	271	0	11.289

Table 6.7: Privacy validation of the teams with correct network relationships.

6.2 | User interviews

Criteria	Category	Type	Question
Trust	User-centric	Qualitative	Do these recommendations reflect your profile?
User-preference	Delivery-centric	Qualitative	Which part-up is the most likely you will join as a partner? How would you rank this recommendations set yourself?
Utility	User-centric	Qualitative	On a scale from 0-7, will you join it?

Table 6.8: evaluation criteria via user interviews

The user interviews perform a face validation of the recommender system. This is a subjective view test to gather the opinion of non-experts. This validation must be performed in a representative simulator. When the outcome is positive, then this is an endorsement for the application of recommender systems. In the user interviews, the recommender system is validated for the trust, user-preference and utility dimensions. Trust is the confidence of a user in the provided recommendation. Trust can be increased by combining known and unknown items or by delivering the recommendations with an explanation. Multiple incorrect recommendations decrease the trust in the recommender system, with the possible end state being ignorant of the recommendations by the users. The most common method to validate trust is asking the reasonability of recommendations in a user study. User preference is the ideal order of the recommended teams. This preference can be determined by letting a user rank his selected teams from best to worst recommendation. Utility is the value a user or the system gains from using the recommender system.

For the interviews were users selected from three groups. This split is made because Avazpour et al. (2014) note that users experience the recommendations differently based on their previous activities. Current and former Part-up team members or developers are excluded from the validation because they may have too much knowledge of the algorithms to give an independent opinion. There are 27 Part-up team members excluded from the user base for this reason. To group the users in three categories an engagement score (formula 6.1) is calculated. Each role a user holds in a team is multiplied with the role-weight (table 4.4) as described in paragraph 4.3. The engagement score approximates the engagement of a user with the Part-up marketplace. This formula 6.1 is constructed by this author.

$$engagement\ score = creator \cdot 2{,}0 + partner \cdot 1{,}5 + supporter \cdot 1{,}0$$

Formula 6.1: Engagement score of a user in all its teams.

The three interview groups for low, medium, and highly engaged users are presented in table 6.10. The two cut-off points for the groups were an arbitrary decision by this author. The *'Total'* column contains the number of all users with engagement within the group's range. The *'Cleaned total'* column is this number of users minus the Part-up team members that were excluded. Especially the group with highly engaged users is small. Still, it is valuable to also receive feedback from users who are active in a high number of teams.

Group	Engagement score	Total	Cleaned total
Low engaged users	0	1.608	1.608
Medium engaged users	0 <> 20	1.102	1.089
Highly engaged users	≥ 20	33	19

Table 6.10: Three user groups based on the engagement score.

From each group, seven people were randomly selected for a user interview by phone, a total of 21 users from three groups. This selection did not correct for age, gender, background, or location, because this information was not at all or not always available in the user profile. Each interview was performed following a drafted protocol (appendix D). Consistently following this protocol ensures the reproducibility of the validation. As this validation was not about the statistical validity, the protocol also provided room to use the interview to collect extra feedback about the algorithms. A limitation of the research set up is that it does not have a graphic user interface to present the generated recommendations. To retrieve these recommendations for the users' interviews in chapter 6 the following Cypher query (6.1) is used. The overview with the recommended teams resembles the 'Discover'-page of Part-up. Shortly before the interview, the overview was sent to ensure the first impression of the user. Some interviews were more extensive, as some users enjoyed elaborating more on their answers and giving broad feedback about the Part-up marketplace.

```
MATCH (u:User)-[r:RECO_RAND]->(t:Team {active})
WHERE u.name="maurits van der goes"
MATCH (t:Team)<-[a:ACTIVE_IN]-(p:User)
WHERE a.role>1
WITH COLLECT(p.name) AS partners, t
MATCH (t:Team)<-[a:ACTIVE_IN]-(s:User)
WHERE a.role=1
WITH COUNT(s) AS supporters, t, partners
OPTIONAL MATCH (t)-[:PART_OF]-(n:Network)
WITH n.name AS tribe, supporters, t, partners
RETURN t.name, t.tags, partners, t.activity_count, supporters,
      t.days_active, tribe, t.link
```

Cypher query 6.1: Retrieve generated recommendation of for example the RAND-algorithm.

A couple of users expressed that they were not searching for teams at all. This was for several reasons. Some used it only for work, others were too busy with their regular daytime job or they already had too many tools for digital collaboration. They tried Part-up out of curiosity. The question if people are actually searching for teams was not included until the last two interviews. However, some of the users also answered the question by themselves. Two questions should have been part of the interview: *"What is your main purpose to use Part-up?"* and *"On a scale from 0-7, are you searching for new part-ups to join?"*. The quality of the offered teams is also an issue. An often-mentioned reason for not joining a team was that it lacks a clear description and goal even though a user liked the recommendation because the subject matches his profile.

The trust in the recommender system with its custom algorithms was expressed with the interview question: *"Do these recommendations reflect your profile?"*. The average results per user group are included

in table 6.11. The first seven interviewees, coincidentally also all highly engaged users, did not express the valuation of their profile reflection as a number on a scale of 0-7. Instead, they described this valuation in multiple sentences. With these recorded answers this valuation was later projected as a number by this author. The results of all user groups are hardly a surprise. The high user group is active in many teams. These are valuable data points for the recommender system, but also leave fewer teams available to recommend especially when the catalog of teams is still relatively small. That is why their average projected score is halfway the range. Almost the same line of reasoning explains why the medium user group scored well. This group has enough data points, while there are still enough teams available to recommend. The low user group scores just below average. This was a bit disappointing and an indication that the algorithms do not perform well with few data points.

User group	**All**
High	3,6
Medium	5,1
Low	3,3
All	4,0

Table 6.11: Average reflection (scale 0-7) of the user's profile by the recommendations

The user-preference dimension was validated with two questions. The first question was: "*Which part-up is the most likely you will join as a partner?"*. This was an indication of which recommendation they liked the most. The sum of all first picks is included in table 6.12. The UBCF-algorithm scored by far the best. Even in the low user group three out of the seven interviewees chose it even though they are not active in a team yet. That is because the algorithm also uses networks in identifying similar users, which is why this algorithm scored best overall. Some interviewees confirmed this. They recognized teams they were co-initiator of or familiar with. The RAND-algorithm scores surprisingly good in the medium user group. This is important from a serendipity point of view. These serendipitous recommendations can uncover unknown interests and this validation shows that these random recommendations are indeed appreciated. A final note about the score of the DF-algorithm in the low user group: this gave the confidence that successful recommendations can also be generated based on limited attributes a user and team hold, which is an important discovery for preventing a cold start among users that suffer from data sparsity.

User group	**RAND**	**DF**	**KBF**	**UBCF**	**IBCF**
High	0	0	2	3	2
Medium	3	0	0	4	0
Low	0	4	0	3	0
Total	3	4	2	10	2

Table 6.12: Sum of first picks to join as a partner

The follow-up question for this dimension was: "*How would you rank this recommendations set yourself?"*. This gives a wider impression of the user preference for an algorithm. Again, the UBCF-algorithm scores best overall, but now also in all user groups. The DF-algorithm is a good second best, especially for the low user group, as was also noticed in the previous paragraph. The IBCF-algorithm scores surprisingly badly, also because it is equal with the RAND-algorithm. This latter algorithm, which functions partly as a control algorithm, scores exactly average. The disappointing score of the IBCF-algorithm is an indication that the participation formula is incorrect because this is a core element in the generation of recommendations. The KBF-algorithm did not score well because users do not understand why these recommended teams are relevant to them. They expect to receive recommendations based on their profile and activity, not especially on their role in a team. This shows that it is important to always present the recommendations in the right context, for example with a small explanation of what recommendations are based on.

Multiple interviewees mentioned two comments, independent of a specific algorithm. First was that they do not want to receive recommendations of teams they are already active in as a supporter. This contradicts the statement of Swearingen and Sinha (2001) as mentioned in chapter 5 that when some recommendations are familiar, the confidence in the recommender system increases. Second, users expressed that some recommended teams were quite old and had minimal activity. They perceived teams that started weeks or months ago as less interesting. Both raised issues should be addressed before the implementation of the recommender system in the marketplace.

User group	**RAND**	**DF**	**KBF**	**UBCF**	**IBCF**
High	2,3	3,2	3,3	3,9	2,7
Medium	3,1	2,6	2,0	4,6	3,3
Low	3,4	4,3	3,5	4,5	-
All	3,0	3,4	2,9	4,3	3,0

Table 6.13: Average rank (1-5) of the algorithms.

The utility dimension was validated by reflecting on their first pick to join as a partner. The question was: "*On a scale from 0-7, will you join it?"*. Table 6.14 shows the results of this question. The two collaborative filtering algorithms perform the best. The rating of the UBCF-algorithm is most relevant because it is based on ten ratings, against the two for the IBCF-algorithm. The results for the RAND-algorithm confirm that these recommendations can be useful from the mentioned serendipity factor. The average of the three ratings is just below average. The DF-algorithm scores very low with four ratings of three and lower. This makes it doubtful that this algorithm can overcome data sparsity. The score of the KBF-algorithm confirms that the two users are unaware of their potential value to the team. Some interviewees also expressed that their limited time was a reason why they rated their first pick quite low.

User group	RAND	DF	KBF	UBCF	IBCF
High	-	-	3,5	5	5
Medium	3,33	-	-	4,5	-
Low	-	2	-	5,33	-
All	3,33	2	3,5	4,9	5

Table 6.14: Average rating (scale 0-7) to join the first pick as a partner

6.3 | Qualitative validation

Criteria	Category	Type	Goal	Unit
Confidence	Recommendation-centric	Qualitative	-	-
Robustness	System-centric	Qualitative	-	-
Scalability	System-centric	Qualitative	-	-

Table 6.15: evaluation criteria via observation/analysis

Next to the face validation, construct validation is performed. The dimensions that are tested in this qualitative validation are confidence, robustness, and scalability. Confidence is the accuracy of the recommender system and its algorithms compared to the design. It is the technical trust that the system performs the steps and calculations indeed as defined. This is connected to the trust of the user-centric category (Herlocker et al., 2000). Robustness is the resistance of fake information on the recommendation. An ideal recommender system is resistant against biased or false information. The insertion of fake ratings or actions is also known as shilling attacks. Scalability is the capacity of the recommender system to keep performing when the number of users or teams increases. Both the architecture as the algorithms should scale along the growth without quality loss. This quality is perceived by the time it takes to generate the recommendation.

This research was already presented during seven occasions: the DataDonderdag meetup[8], Graph Database Amsterdam meetup[9], Big Data Analytics Europe 2016 conference[10], Utrecht Data School guest lecture, TPM Policy Analysis section lunch lecture Neo4j's GraphConnect Europe 2016 conference[11], and Data Science Utrecht meetup. The Q&A sessions and informal discussions afterwards also functioned as validation of this research on the confidence, robustness, and scalability dimensions. These audiences differed quite a bit. Both events by Neo4j, GraphConnect conference, and Graph Database Amsterdam meetup, had the most tech-savvy attendees. Most attendees use Neo4j or other database technologies professionally on a weekly basis. The DataDonderdag, Data Science Utrecht and Big Data Analytics events attracted technology aware people, but who are not necessarily developers such as managers and other people that are interested in

[8] https://issuu.com/cre-aid/docs/datadonderdag_editie11/17
[9] https://www.youtube.com/watch?v=ClJEoA0a4Rc
[10] https://www.youtube.com/watch?v=OBuDBC9y_KU
[11] https://neo4j.com/blog/collaborative-filtering-creating-teams/

innovations for strategic decisions. The academic groups, Utrecht Data School and TPM's Policy Analysis section, approached the research from a theoretical point of view.

The recommender system and its custom filtering algorithms as a solution were well received by the different audiences. The added value of a recommender system was acknowledged with some reservations. For example, the understanding of the relation between a user and a team was questioned, as collaborative filtering was never applied to self-managing team formation. The participation formula was especially disputed for its academic relevance. As this is the heart of the IBCF-algorithm, the confidence in the whole algorithm was challenged. During the presentations and in the discussions, it was emphasized that the research focus was on applying these methods to this case, not providing the optimal configuration of the algorithms. Because this is the first recommender system for the Part-up marketplace, the expressed concern of translating self-managing team characteristics into algorithms was acknowledged. This requires a multidisciplinary research team with at least a psychologist next to an IT-developer.

Questions were also asked about including latent user feedback in the dataset for example if a user leaves a team or dismisses a recommended team. It was suggested that with this information a user shall no longer get similar teams recommended. This was on purpose excluded in this research. As mentioned, the whole research was performed offline. Thus, it was not possible to gather that type of user feedback. However, it is worthwhile to investigate that. For example, if a user neglects a team, a new relationship between both entities is added with the label: NOT_INTERESTED_IN, improving the confidence in the model by including latent user feedback is a subject for future research.

The robustness dimension was addressed by raising importance for algorithms to be coalition proof. Both collaborative algorithms use an implicit rating. This rating can be influenced by becoming active in teams. The ratings that a user can receive for its activity in a team is a risk for this robustness. The implicit rating of a team can also be influenced, although this is harder. Trying to steer the rating by adding contributions or comments is especially troublesome, as both user actions are very visible on the marketplace. Such misuse can be easily identified and reported by users.

Multiple doubts were raised about the scalability of the similarity query. Currently, it is not memory-efficient for two reasons. First, because it is collecting the required data in an inefficient way, calculating the similarities was a tedious job. It takes a lot of time and machine power. With the database growing this even becomes a larger problem. Second, the query is running inside Neo4j next to the other tasks. This processing unit cannot be scaled independently of the whole Neo4j database. It was suggested to explore Apache Spark for the external calculation of the query. Future research was advised on the opportunities of applying steepest descent. The comparison of the algorithms is currently a time-consuming and complex process, and not scalable. The desire was expressed to run multiple algorithms in the recommender system. This would create the opportunity to A/B-test the algorithms and use the algorithm that functions best for a user group. A feature to perform an A/B-test with multiple algorithms was not part of the GraphAware Reco version used in this research.

6.4 | Conclusions

Reflecting on all evaluation sections and dimensions it can be concluded that a recommender system with these custom algorithms is promising. The users value it, experts support it, and technically it is functional. Based on the constraints and objectives it can be concluded that a recommender system can successfully support a user in discovering teams. The constraints were personalized recommendations for each user and the implementation of the platform rules. First, the recommender system-generated personalized recommendations for each user based on their activity or profile. Many users valued the recommended teams as they recognized themselves in it. Second, the algorithms ran correctly with the incorporation of the platform rules. This constraint was confirmed with the accompanying Cypher queries that were written to validate the recommended teams on coverage and privacy.

The objectives were regular updates of the recommendations, scalability, open-source software, and easy to set up. Most of these objectives were met. First, the framework regularly updates the recommendations. Second, the algorithms, framework, database, and architecture were designed to scale up. Especially when a model-based approach with an offline model and online recommendation framework is chosen. However, this scalability objective is not tested in detail with different scenarios. Third, all used software is open-source available. And this research also contributes to this generous movement of software development with publishing the code of the algorithms, the MongoDB to Neo4j ETL script, and the MeteorJS to Neo4j ETL API under an open-source license. Fourth and final, the architecture was easy to set up. Although existing software was used, there were only limited use cases and experience available. Despite this, developing the research setup went relatively smoothly. A new logical data model was designed that is suitable for a recommender system. This model proved to be diverse enough for using different algorithms and efficient in rapidly processing data. Most of the development time was spent on transforming well-known techniques like collaborative filtering to this recommender system with a graph database. By meeting the purpose, function constraints, and objectives as defined for the program of business demands this new recommender system is ready to support all workers struggling to find the teams, they will love to work in.

However, stimulating trust in the recommender system via the recognition of supported teams was not appreciated. This is the opposite of the mentioned conclusion by Swearingen and Sinha (2001) that the confidence in the recommender system increases when the recommendations list includes familiar teams. This can be explained by the assumption that the action of supporting a team not only makes a user familiar with a team but also connected. Familiarity is more related to teams that a user has seen in other search results before, or even checked out shortly. These viewed teams are not filtered out of the recommended teams, but also not included on purpose. With no longer including the supported teams, another way must be found that can create recognition that is satisfying and not distracting. A positive observation is that users tend to like teams more when those are recommended. Part-up should also clearly mention that the teams that are picked by the recommender system are recommended teams. When users get teams advised based on their team fit, they should be better presented.

A combination of multiple types of algorithms has arisen as a promising method. Multiple researchers (Claypool et al., 1999; Melville et al., 2002; Ogul & Ekmekciler, 2012) argue that a combination can overcome

limitations that are imposed by the data, leading to more stable results than collaborative filtering. Based on the results of the evaluation, a hybrid version of the UBCF-algorithm and RAND-algorithm was chosen (figure 6.1). This was because the UBCF-algorithm performed best and the RAND-algorithm provides a serendipitous element and addresses data sparsity. It covers 2,627 (96%) users while respecting privacy and platform rules. This hybrid algorithm provides qualitative recommendations to high, medium, and low engaged users - recommendations that users trust and can use. The straightforward algorithm structure without a complex similarity query makes it reliable, robust, and scalable. The recommender system with this hybrid algorithm and a couple of adjustments is implemented into the marketplace.

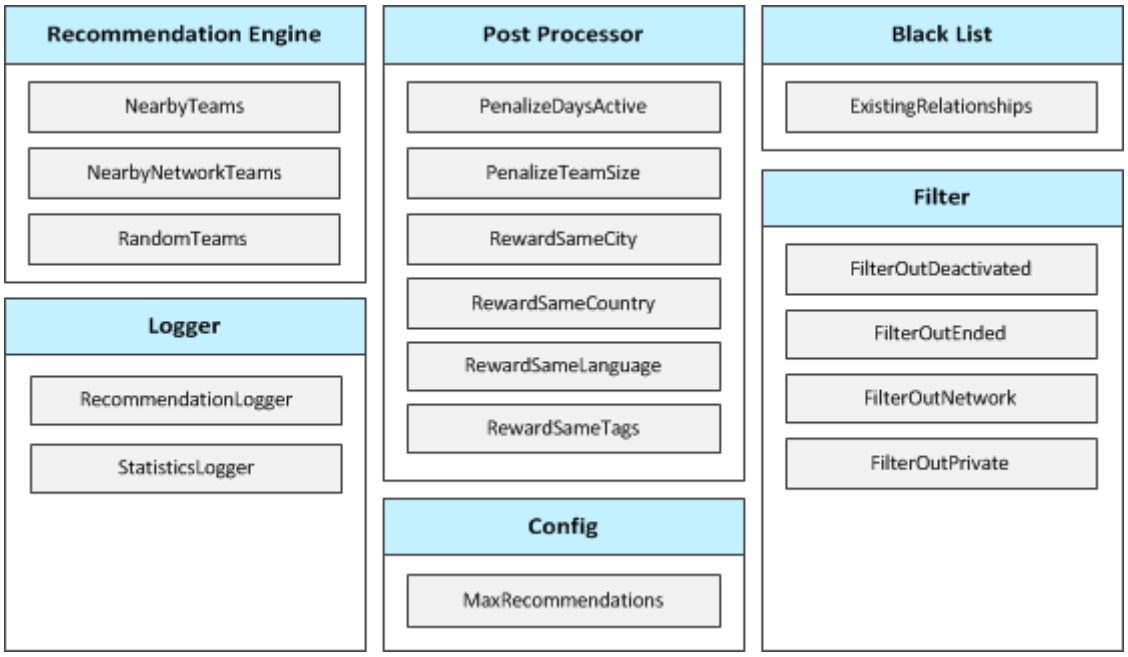

Figure 6.1: Hybrid algorithm modules in the GraphAware Reco framework.

7 | Deployment phase

The deployment of the recommender system is the final phase of this research. This chapter tests the technical feasibility of the research in the marketplace: *Is the designed recommender system with its custom filtering algorithms able to suggest relevant self-managing teams to users in a digital marketplace?* The evaluation was performed in an offline environment and without a graphic user interface, before moving to field testing. All elements were designed and developed to eventually run in an online production environment. The theory is more beautiful than practice. Robert Goodin (1996) defined inventions in socio-technical systems as an evolutionary process of trial-and-error. The eventual artifact is developed while trying to influence the system. The system is deployed followed by the DevOps methodology (Bass et al., 2015), aimed to continuous deploy improved releases. Hereby lowering the time to market and increasing customer satisfaction. The technical feasibility is discussed with four key aspects of the recommender system: infrastructure, data flow, algorithm, and graphical user interface.

7.1 | Infrastructure

Before the recommender system can be implemented in the Part-up marketplace a couple of steps must be taken. The architecture is visualized at a high level in figure 7.1. Each new or adjusted element and the implementation steps are described in this chapter.

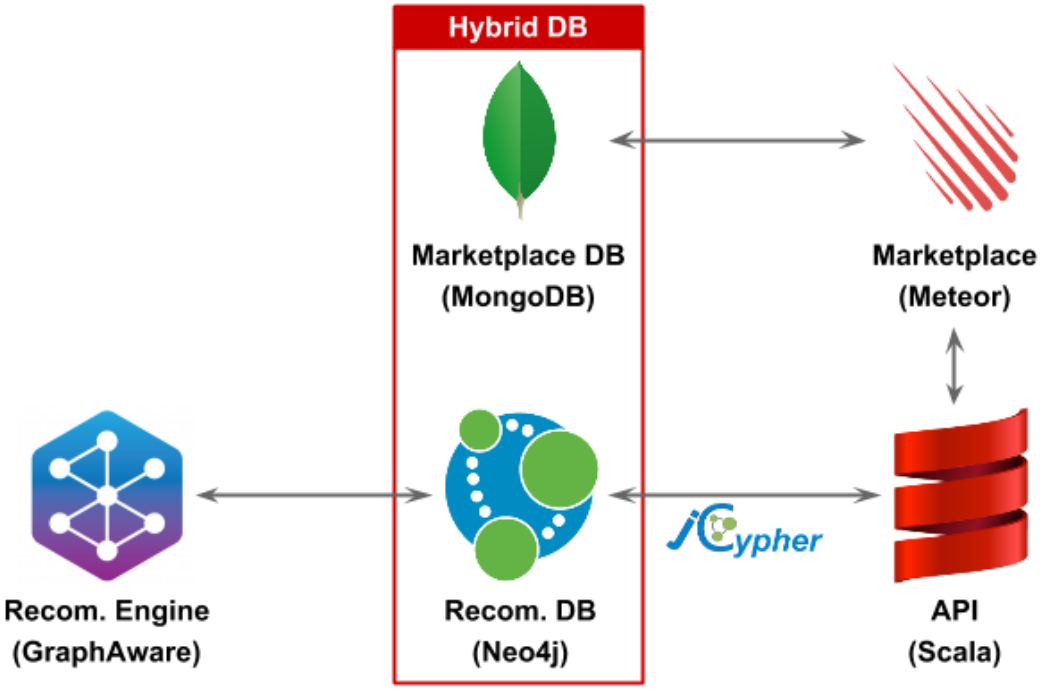

Figure 7.1: Design recommender system in production

The Part-up marketplace is as mentioned based on the open-source JavaScript web application framework MeteorJS. The main advantage of MeteorJS is that it is cross-platform and integrates with the

document-oriented database MongoDB. The marketplace database MongoDB and the recommendation database Neo4j act together as a hybrid database structure. Each optimally designed and scalable for their specific purpose. MongoDB rapidly delivers the content of the marketplace, while Neo4j excels at revealing relationships within this content. Data is flowing from MongoDB to Neo4j via the marketplace and the Application Programming Interface (API). All these parts of the recommender system are hosted on Amazon Web Services just like the marketplace. This is a scalable hosting solution. The elements of the marketplace and recommender system are packed in Docker containers. Docker provides virtual software containers. This container structure makes the architecture even more scalable. The four Docker containers are MeteorJS framework, MongoDB, API, and Neo4j with GraphAware Reco. The scalability of Neo4j can be separated into two parts: read and write. In the online version of the recommender system, there are not one but two software tools that read and write to Neo4j: the API and the GraphAware Reco. The size of the Neo4j database is rapidly increasing especially because of the relationships. That is why it is efficient that these functions are separated.

The code was written following the ten guidelines for future-proof code by Joost Visser (2015). These guidelines improved the readability of the code for a broad range of developers. Before the recommender system was sent to production each piece was reviewed by at least one other developer. The web-based repository service GitHub was used to co-develop. All code is also publicly available on GitHub with a GNU AGPLv3 open-source license. This gives the opportunity to a wide audience of interested developers to reflect on it.

7.2 | Data flow

The API is written in the programming language Scala by developers of Part-up and me. These developers were, like the other people that are mentioned in this chapter, all members of the Part-up development community. The code of this API is available on GitHub[12]. The API provides several functions to the marketplace. For this research, an ETL-module was added. It receives the events that are broadcasted by the marketplace to a specified endpoint. These events are unpacked into objects and transformed into queries. These queries are needed to insert, change, update or remove entities, relationships, or attributes in the recommendation database Neo4j. All entities or relationships that do not exist yet in Neo4j are created by the API. The JCypher library was used to construct the queries reading and writing in the recommendation database. With this library, it is possible to access Neo4j with commands written in the Scala. These Scala commands are translated by the library into the query language of Neo4j. Combining the functionality of Scala with the structure of Neo4j is an advantage of using this library. A disadvantage is that the library is young and incomplete. However, a developer of JCypher has pledged his support to this recommender system by delivering the required features in extra releases (personal communication, 26 April 2016). There are only thirteen events used to recreate the required part of the database (Table 7.1 & 7.2). The API sends also on request the recommended teams from the recommender database to the marketplace.

[12] https://github.com/part-up/api

Event	Inserted	Updated	Changed	Removed
Networks	✖	✖		
Teams		✖	✖	
Users	✖	✖	✖	

Table 7.1: Collection events

All these collection events are sent with the user ID and document in the payload. Networks, teams, and users can only be deactivated and not removed for marketplace technical reasons. They are interdependent for the correct functioning of the marketplace. Not removing these entities also ensures that the recommendation framework can continue processing this data for additional insights. There are also six custom events used (table 7.2).

Event	Eventname	Attributes
Network location changed	'networks.location.changed'	tribes, newLocation, oldLocation
Network member removed	'networks.uppers.removed'	user._id, network._id
Team location changed	'partups.location.changed'	partup, newLocation, oldLocation
Team supporter removed	'partups.supporters.removed'	team, user
Team unarchived	'partups.unarchived'	user, team
User settings updates	'settings.updated'	user._id, language

Table 7.2: Custom events

The logical data model was slightly adjusted for the implementation. It was the ambition to remove all unnecessary attributes in the online version of the Neo4j database. For generating the recommendations, the *'name'*-attributes are not used. This information only needs to be presented to the user. The MeteorJS-framework retrieves that data directly from the MongoDB database. For navigating in the simple GUI of the Neo4j database it was decided to keep these *'name'*-attributes. Otherwise are the entities only identifiable via the *'_id'*-attribute which is a string of random letters and numbers. Because the IBCF-algorithm was not chosen all the attributes of the *'ACTIVE_IN'*-relationship were dropped except from role (figure 7.2). The extensive data dictionary of the logical data model is included in appendix B.

A secondary reason to drop these attributes is that these were not fully functional yet in the API. Along with Eric Raymond's philosophy (1999): *"Release early. Release often. And listen to your customers."*, it was decided to deliver an initial recommender system first instead of trying to create the most complete database for future algorithms. The updated logical data model for the Neo4j database was designed explicitly for the UBCF-algorithm. As a result, the API also contains methods to structure the data according to this. If this algorithm is adjusted or replaced, then probably the logical data model and the API must also be updated.

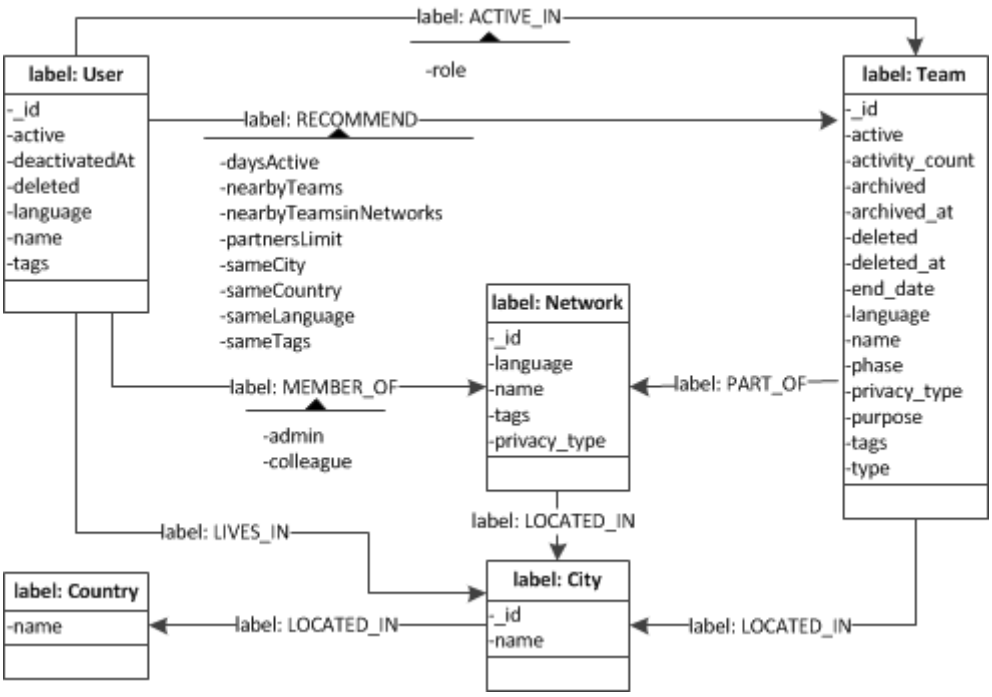

Figure 7.2: New logical data model

7.3 | Algorithm

The UBCF-algorithm was improved by a developer of Part-up and me for four reasons: evaluation results, changes in the marketplace, online environment, and small fixes. First, the feedback of the evaluation research was incorporated. This means that every team a user is active in is blacklisted for the recommendations (Cypher query 7.1, 7.2 & 7.3). This used to be only the teams in which a user was a creator or partner. Now also the supported teams are not included in the recommendations anymore.

```
MATCH (u:User)-[r:ACTIVE_IN]->(t:Team)
WHERE id(u)={id}
RETURN t as blacklist
```

Cypher query 7.1: New Blacklist for current relationships.

```
MATCH (u:User),
      (t:Team)
WHERE id(u)={id}
      AND t.privacy_type=2
RETURN t as blacklist
```

Cypher query 7.2: New filter out private teams.

```
MATCH (u:User),
      (n:Network)<-[:PART_OF]-(t:Team)
WHERE id(u)={id}
      AND (t.privacy_type=4 OR t.privacy_type=5)
      AND NOT (u)-[:MEMBER_OF]->(n)
RETURN t as blacklist
```

Cypher query 7.3: New filter out private teams in a network.

Older teams are less likely to be included because users experienced these as less relevant. It was considered to filter out all older teams, for example, teams that were started longer than 30 days ago. Still some teams exist for months, are still active, and are also relevant to join in a later stage, for example, teams that organize an event. That is why the new post-processing element PenalizePartnerAmount was developed. A team that is older than 7 days can get a maximum deduction of 5 points. The penalty follows a Pareto-scale with the 80% point at 90 days. This scoring hardly affects the active teams, while it naturally filters out the inactive teams. The new overview of the different post-processing elements is included in Table 7.3.

Element	Variable	Max negative score	Max positive score	Distribution
PenalizeDaysActive	daysActive	-5	0	Pareto
PenalizePartnerAmount	partnersLimit	-5	0	Pareto
RewardSameCity	sameCity	0	5	Degenerate
RewardSameCountry	sameCountry	0	1	Degenerate
RewardSameLanguae	sameLanguage	0	1	Degenerate
RewardSameTags	sameTags	0	50	Uniform distribution
	Total:	-10	56	

Table 7.3: Post-processing elements

The second reason why adjustments were made is that the marketplace was further developed. The privacy type of teams was expanded with two new types for teams in any network (table 7.4). Privacy type 6 is a hidden team for the co-creators in a network. Privacy type 7 is a hidden team for the co-creators and colleagues in a network. Co-creators function like the administrators of a network. Colleagues are users who are identified by the co-creators as regular contributors.

Team privacy type	Network privacy type	Description
1	-	Public teams
2	-	Private teams
3	1	Public teams in a public network
4	2	Private teams in an invitational network
5	3	Private teams in a closed network
6	1/2/3	Hidden teams for co-creators in a network
7	1/2/3	Hidden teams for co-creators and colleagues in a network

Table 7.4: New privacy types of the teams.

As a result, both the API and the algorithm were adjusted. The API supports multiple admins (co-creators). The *'colleague'*-attribute is also added to the *'MEMBER_OF'*-relationship. Two filter-modules were added in native Cypher: queries 7.4 and 7.5. This can be validated and governed with the Cypher queries C.12 and C.13 included in appendix C.

```
MATCH (u:User),
      (n:Network)<-[:PART_OF]-(t:Team)
WHERE id(u)={id} AND (t.privacy_type=6)
      AND NOT (u)-[:MEMBER_OF {admin:true}]->(n)
RETURN t as blacklist
```

Cypher query 7.4: Filter out hidden teams for co-creators in a network.

```
MATCH (u:User),
      (n:Network)<-[:PART_OF]-(t:Team)
WHERE id(u)={id} AND (t.privacy_type=7)
      AND NOT ( (u)-[:MEMBER_OF {admin:true}]->(n)
      OR (u)-[:MEMBER_OF {colleague:true}]->(n) )
RETURN t as blacklist
```

Cypher query 7.5: Filter out hidden teams for co-creators or colleagues in a network.

The third and fourth reasons for adjusting the algorithm were smaller. The algorithm was prepared for a database that is live updated, and improvements were made in the code style and error fixes. The code of this algorithm was reviewed by a senior data scientist of GraphAware. It was expressed that the recommendation framework is applied in a really good way by writing clean code and utilizing the different options it provides (personal communication, 16 March 2016).

Currently, it is not possible to return the context of the recommendations via GraphAware Reco. The context includes the filter criteria that are met. This can be for example the fellow team members who are active in the recommended team or the tags that the user and team share. For this purpose, the initial versions of the NearbyTeams and NearbyNetworkTeams recommendation engines also return the IDs of the entities in the queried path. For example, the team ID and user ID for the NearbyTeams element. These values were dropped for performance reasons as it is not possible to use this information yet. Another adjustment to slightly improve the performance is the addition of entity exclusions. When querying the defined path, it is no longer possible that both users are the same (`u1<>u2`) and the teams are the same (`t<>reco`). The new NearbyTeams and NearbyNetworkTeams elements are included as Cypher queries 7.6 and 7.7.

```
MATCH (u1:User)-[r1:ACTIVE_IN]->(t:Team)
      <-[r2:ACTIVE_IN]-(u2:User)-[r3:ACTIVE_IN]->(reco:Team)
WHERE id(u1)={id}
      AND u1<>u2
      AND t<>reco
RETURN reco,
      (r1.role + r2.role + r3.role)/3 AS score
```

Cypher query 7.6: User-based collaborative filtering algorithm via teams (NearbyTeams).

```
MATCH (u1:User)-[r1:MEMBER_OF]->(n:Network)
      <-[r2:MEMBER_OF]-(u2:User)-[r3:ACTIVE_IN]->(reco:Team)
WHERE id(u1)={id} AND r3.role>=1.5
      AND u1<>u2
RETURN reco,
      0.5 AS score
```

Cypher query 7.7: User-based collaborative filtering algorithm via networks (NearbyNetworkTeams).

To improve the models, it is advised to include a psychologist to better understand the team formation process and team interaction. Inside the recommender system, multiple algorithms can be tested easily to incrementally increase the quality of the recommendations. Still, this first hybrid filtering algorithm generates relevant recommendations for the users and is a building block for future developments in this direction.

7.4 | Graphical user interface

The recommendations are presented in a separate page of the marketplace. It was not possible to replace the teams with the recommendations on the *'Discover'*-page of Part-up. This page shows the newest or most popular teams with a search function. The latter is the showstopper for integrating recommended teams into this page. The database-wise search function would stop working with only the recommendations. For this reason, a button was added to the *'Discover'*-page, opening a page (figure 7.3) with a maximum of four recommendations. This page also contains a clear title and short explanation on why these teams are recommended. Providing this context is an important element as Holten (2001) and Xiang (2011) mentioned. The page was developed by two web developers of Part-up. With this last piece of the puzzle ready, deployment to the marketplace could start.

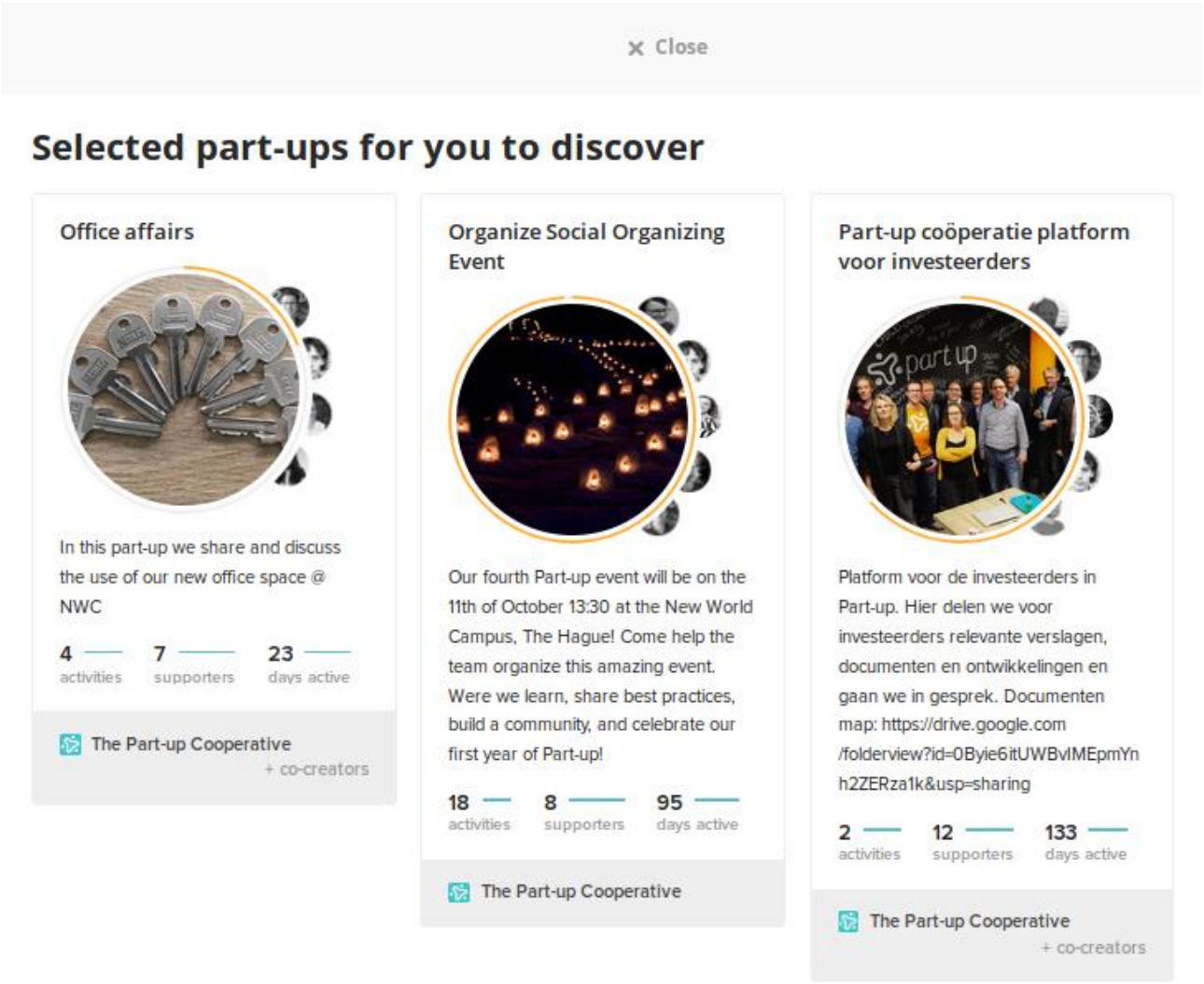

Figure 7.3: Recommendations page on part-up.com.

The recommender system was deployed in four steps. First, the API was added to the architecture. Second, the Neo4j database with GraphAware Reco and algorithms was added to the architecture and connected to the API. Third, on 3 October 2016, the recommendation page was added to the marketplace and connected to the API without being exposed to the users. Only informed users with a direct link could access the page. Fourth and finally, the recommender system was opened to the public on 27 November 2016. This step-by-step process gave us the opportunity to track down and fix all errors.

Research by Kujala (2003) shows that user involvement in software development leads to systems success and user satisfaction. For the development of this matching solution the users were included during the model development. However, they were not involved in the final phase. The output on the platform was not tested with users. Although this involved a slight change in the model. Additionally, the final interface of the recommendations was neither tested with the users. Still, this research contributed to the smooth integration of the recommender system into the marketplace.

7.5 | Conclusions

The successful integration of the recommender system into the marketplace is again a confirmation that the proposed solution is very suitable. The technical feasibility of the solution was proven. The infrastructure with a hybrid database and additional API can process the data events, run the algorithm, and return recommendations to the marketplace. The chosen merge-method either creates or updates an item or relationship. This decision not only prevents data duplication, but it also makes the database stateless. Any failures of the database or data loss are automatically recovered. This data was stored following an adjusted logical data model for the functioning of the new hybrid algorithm. Based on user feedback a new penalize module was developed, and some smaller modules updated. Also, developments of the marketplace itself were successfully incorporated in the deployed version. The lacking ability to return the context of a recommendation is considered not a vital feature for the first version of the recommender system. The graphical user interface of the recommendations provides enough information about the teams and presents the recommendations in the right context. However, this part was not validated with users.

In data projects, there are often three people involved: a data scientist, a data engineer, and a development-operations (DevOps) engineer. The data scientist is responsible for algorithm development. The data engineer builds for the recommender system framework. The DevOps engineer deploys and monitors the system. The algorithm and the framework are closely related and require coordination. Therefore, it was an advantage that both the algorithm and the framework were developed by one person, this author. Considering the technical achievements in the deployment phase and considering the positive three-step evaluation, the recommender system is feasible in production.

8 | Conclusions & future research

This final chapter discusses how the findings of the research impact organizations and their workers.

Reflecting on the design objective of supporting users in self-managing team formation with a recommender system and custom filtering algorithms. While this may indicate the end, it is, perhaps, also a beginning - the conclusions are accompanied by the recommendations for future research.

8.1 | Conclusions

8.1.1 | Conclusions

Custom filtering algorithms can generate recommendations of self-managing teams. The five algorithms described all generated recommendations for teams to join and some also met the constraint of personalized recommendations. These recommendations are suggestions for available self-managing teams that reflect the profile of a user. Above all, the user-based collaborative filtering algorithm performed best in recommending teams which indicates that this domain-independent approach is also applicable in this context of self-managing team formation. Furthermore, the hybrid approach with additional random filtering addresses data sparsity challenges of collaborative filtering. The enrichment provides serendipity to the recommendation while maintaining performance.

By incorporating a recommender system, the marketplace is addressing the issue of bounded rationality of humans. Organizational structures are flattening, and workers receive more responsibilities with the introduction of self-managing teams. Workers can perceive this new freedom both as a pleasure and a challenge. When function profiles and departments no longer form the perimeter of their professional activities, workers are overwhelmed by the opportunities in different teams. The bounded rationality of humans results in the risk of non-optimal team choices, which is not beneficial for the workers and the organization. It also threatens the successful adoption of the new structures that rely on this decentralized coordination of work. These new structures are essential for organizations to remain competitive in a globalizing and increasingly competitive environment.

This choice dilemma needed to be addressed because combining the supply in a digital marketplace can cause information overload. It is for this reason that a socio-technical system must support users in discovering and joining self-managing teams within and across organizational borders. The recommender system addresses this need for the marketplace in two ways. First, the marketplace ensures a soft landing. The recommender system suggests teams to workers who do not know which teams are relevant to them. Instead of dropping out due to overload, they can use the recommender system as a starting point. Second, for workers that feel already familiar with this new way of working, the recommender system can function as a control mechanism. Do they have an overview of all teams that are relevant for them to join? A significant note is that not all the interactions of the workers are tracked by the marketplace. In the end, only a fraction of all professional interactions take place within the online domain. The marketplace functions for the decentralized

coordination of work as a supporting institution for the workers and the organization. It must be clear that the marketplace is not the enforcing institution. Rather, the marketplace that has a smart filter can support self-organization in teams.

The adoption of a marketplace to structure the organization is an investment in easing the process via technology instead of more management institutionalization. It is about trusting the workers to make their own decisions. The future of organizations has no longer a top-down, but a bottom-up approach. The marketplace is an instrument to empower workers to enforce this change within their organizations. By simplifying communication, information gets decentralized. This enables workers to organize their tasks and teams themselves. The marketplace brings together workers and teams. While the recommender system is a digital form of coaching. The introduction of this technology leads to the discovery of colleagues with unknown skills. As skills are more diverse and specialized nowadays, supporting technology is necessary to match the capabilities of workers with the goals of an organization. With this digital institution, the optimal usage of the talents can be realized. The design of the recommender system ensures that it keeps adapting to the changing activities and interests of the workers. For these reasons, a marketplace with a recommender system is essential in this pioneering way of professional collaboration. It is this agile approach that enables organizations to better respond to the changing demands of the market. This socio-technical system fundamentally reshapes how organizations are structured and workers function. Workers will rely on the overview, easy communication, and vigorous organizational functionalities it provides. Repeated use of the recommender system increases their trust in the system, which means that the recommended teams will be followed more often by the workers. A new way of working that makes the workers more effective and happier, while also contributing to the goals of the organization. This modern way of organizing is impossible without a supporting institution like this marketplace with a smart filter.

8.1.2 | Limitations

The limitations of this research are the governance of algorithms, switching supply and demand, and detailed input. While designing the system there was too little attention to the governance of algorithms. There was no data bias check or a review on the algorithmic bias. But the algorithm is transparent as it is publicly available on GitHub for inspection. Within the marketplace though, there is little information presented on the version of the algorithm, any A/B-test experiments, or the context of the recommendations. Monitoring with the validation queries showed that roughly 76 % of the users have recommendations because these users have provided any demographic details in their profile or have joined a team. Nonetheless, also a significant group of users is missing the benefits of the recommendations. 60 % of the teams are recommended to minimal one user. For both the left-out users and teams the level of activities is unknown. It is advised to monitor these coverage metrics, plus additional metrics to have insights on the performance and health of the hybrid algorithm. Governance and the ethical implications were not neglected, but with the growing attention for algorithmic bias, this should have been addressed more.

On switching supply-and-demand, there is not a preferred position for team creators. All users can use the recommender system for discovering new teams. This system does not work the other way around yet,

demand-driven instead of supply-driven instead of supply-driven. A feature to invite users in a team. Although, this new feature can be quite easily developed with new data retrieval and graphic user interface components, relying on the same hybrid database, recommendation framework, and algorithms.

Human interaction and judgment are two very difficult issues for a marketplace including the recommender system. Simply not all interactions are tracked by the marketplace or judgments understood by the recommender system. Although it is artificial that it is. Users meet in real life, use several tools to collaborate, or have other creative ways of interacting outside the marketplace. That is a fact to consider when developing an algorithm. For example, by analyzing and predicting users' behavior based on the fraction of interactions that were tracked.

8.1.3 | Reflection

The powerful symbiosis between technology and management is confirmed once again, as this research shows that a recommender system can contribute to the process of self-organizing. The theoretical findings from the literature review were successfully translated into technology. The evaluation and deployment prove that it is technically feasible, and the user feedback confirmed both the desire and appreciation for such a supporting institution. Self-managing teams are the cornerstone of a modern organization with a network structure. The network structure makes the organization flexible to adjust to market demands and use their available resources optimally. This lean structure is required to compete in a competitive economy that is increasingly global. For the continuity of the organization and its workers, it is important that their structure can face these new complexities. organizations are essential to society, accordingly this evolution has even more stakeholders. This line of reasoning does not mean that placing a filter on the supply of teams is enough to adjust to the new complexities, but it does confirm that workers are better able to face these complexities with a filter that supports them.

8.2 | Future research

It is vital to explore directions for future research, as subsequent research can challenge and improve the present research. Four recommendations for future research in different directions are formulated: case study with soft skills, artificial neural network, hybrid algorithm, and presentation of the recommendations. Each recommended research is briefly described in the following paragraphs.

The first and foremost recommendation for future research is a case study on the effects of the recommender system on an organization and its workers. Including a study into the incorporation of soft factors by the algorithms. Examples of these soft factors are psychological factors of workers and their working styles and preferences. The present research reasoned this effect based on the literature study and evaluation. Extensive empirical research within an organization was outside the scope of this present research. A case study can analyze multiple assumptions of this report. Testing the described scenarios of use: soft landing and control mechanism. And explore if there are also other notable scenarios. The next step is relating this influence on the worker's overview of their organization and his organizational choices. On a higher level, it is relevant to know if the recommender system stimulates activity in teams throughout the whole organization. Ultimately

it is very valuable to assess the relationship between the usage of the recommender system and the improved human allocation within the organization, which is important for the organization's productivity. Performing this formal research method on this subject would be a significant contribution to agile organizations.

A technical direction for future research is testing an artificial neural network as the engine of the recommender system. An artificial neural network is a self-learning system with great computational capacities. The system is a collection of loosely modeled neurons. Each neuron is connected to other neurons by multiple axons. This network of input, hidden and output neurons can be trained to perform complex computations. A wide range of tasks can be solved by this smaller and digital version of the human brain. YouTube already replaced the core of its recommender system with this method (Covington et al., 2016). In the past YouTube applied matrix factorization methods just like the item-based collaborative filtering algorithm in this research. A big advantage of neural networks over matrix factorization methods is the increased domain independence. Neural networks take all the available parameters and learn the right combination autonomously. Google open-sourced the technology for this method as TensorFlow. While scalable processing power is provided by cloud hosting services like Amazon Web Services. That is why no longer the technology nor the processing power, but the dataset is the limiting factor for applying this approach to this research subject. Neural networks are beneficial for recommender systems as they can discover and compare many more relations to build a model. This highly relevant model is used by the recommender system to generate recommended teams for each user. The performance of the recommender system can be enhanced by integrating live user data. In the offline setup of this research that specific type of information was not available. Questions to be researched include: On which recommended teams does the user click? Or which teams does he dismiss? In which teams does he become a supporter? And even better, in which teams does he become active as a partner? The research should focus on identifying what user feedback correlates with the quality of the recommended teams. Being part of the marketplace, this specific user feedback can be directly returned to the recommender system too. Novel reinforcement learning methods use these broad data volumes to generate recommendations. In combination with the previous paragraph, this live feedback can be fed to a neural network to ensure scale. This research subject would enable continuous improvements to the algorithms.

The chosen hybrid algorithm is just the start of a far more complicated blending of algorithms. Netflix for example also combines multiple algorithms to get to the best prediction of their users' preferences. Effectively blending the algorithms can further solve the weak aspects of each algorithm, like the cold start problem for collaborative filtering algorithms. However, this research expresses doubts about the computing capacities of Neo4j. Especially for machine learning applications, which are even more complex, this is a blockade. These limitations can be resolved with an external computing framework like Apache Spark. This is an open-source cluster computing framework that performs clustered, in-memory data processing tasks (Neo4j, 2016b). A subset of the Neo4j-graph is exported to Spark where these complex analytical tasks are performed, and the results are then written back into Neo4j. Separating the computation and storage functions ensures that the recommender system is scalable. However, blending algorithms is not just a simple sum. The coverage and weight of each algorithm can be adjusted to create the best mix. The most optimal mix can be

identified by applying A/B-tests with different algorithms combinations. These specific merge details are why there is more research needed in the combination of different algorithms.

The opportunities for practical improvements are based on experiences with the recommender system in the Part-up marketplace. The presentation of the generated recommendations can be improved and diversified. As concluded, it is important for a user to present on which criteria a team is recommended. This is called the context information of a recommendation. Most recommendations are generated while assessing multiple criteria. Which criterion or combination of criteria to show next to the recommended team should be further researched. A successful presentation of this context information can be measured with a higher click-through-rate and users becoming active in a team as a partner. Ideally are the marketplace pages with search results and recommendations of teams combined. Currently, there are two separate pages as described in chapter 7. An option to realize this is by combining the technology of Elasticsearch, GraphAware Reco, and Neo4j. Elasticsearch is the most popular open-source search engine according to solid IT's DB-Engines Ranking (2016). It is schema-free, multitenant-capable, and distributed. The connecting element between Elasticsearch and Neo4j is Graph-Aided Search by GraphAware. This new module combines the score of the search engine with the score of the recommender system (Bachman, 2015). Thus, for example, when a user searches for a team, based on this textual match he does not get the most relevant teams overall. He gets the most relevant teams specifically for him. The integration of recommendations into the search engine creates one central and familiar point on the marketplace for discovering teams. This development not only makes it easier to access the recommended teams but also ensures that when a user is searching for teams himself, these presented teams are an optimal selection too. Next to this improved presentation, the presence of the recommended teams on the marketplace can also be diversified. Three examples for different usage are: to show on a team page a couple of teams that are similar, to send out a weekly email with recommended teams, or to push a message via the mobile app when an interesting team is started. Other digital marketplaces already do this; however, it is unknown what users desire in a marketplace for team formation. Research into the preferred context information and communication channels are important design information.

This future research section shows that there are many opportunities to continue this research on matching supply and demand in a marketplace for team formation. There is not one specific next step, nor an exact roadmap. Some described subjects are in even different directions. These future research suggestions seem to be exclusive for team formation. However, this research shows that the methods of different industries can be successfully transferred. With the volume of data increases in this age of information. This should be another argument to further invest in the research on recommender systems.

8.3 | Reflection CoSEM program

Graduation research for the Complex Systems Engineering and Management (CoSEM) master program must design a complex system in a socio-technical environment. In detail, this research confirms this criterium on multiple aspects. This is interdisciplinary research that takes technical, institutional, economic, and social knowledge into account. It manages multiple stakeholders of a recommender system for the formation of self-managing teams. Each with diverse interests towards collaboration in a digital marketplace.

The research applies several CoSEM methods, tools, and techniques for design and impact. Examples are design science research, architecture design meta-framework, and the best-of-class chart. The research designs a complex system in a socio-technical environment with technical innovation achieved by creating a recommender system for team formation. Combining management strategies and systems engineering approaches. This system includes a cultural aspect of people collaborating in teams and in a virtual environment. It involves human behavior and users' demands in team formation. But also included user involvement in the design process. The research addresses the platform governance and algorithmic bias as ethical issues that are part of the system. Although the application of this research is domestic. The knowledge base of literature, available artifacts, and personal communication has certainly an international character. Corresponding with the information and communication track, which I followed, there is a strong information technology component that is specially designed for this research. It designs a system architecture with technical complexity. While designing and developing the recommender system this research dealt systematically with design issues. Reasoning from this reflection, it can be concluded that this research fits the CoSEM master program.

Reference list

Allen, R. C. (2009). *The British industrial revolution in global perspective* (pp. 135-181). Cambridge: Cambridge University Press.

AMS-IX. (2020). 17% traffic increase on the AMS-IX platform due to Corona/ COVID-19 crisis. Retrieved on 18 April 2020 from https://www.ams-ix.net/ams/news/17-traffic-increase-on-the-ams-ix-platform-due-to-corona-covid-19-crisis

Aughton, P. (1996). Participative design within a strategic context. *The Journal for Quality and Participation*, *19*(2), 68.

Avazpour, I., Pitakrat, T., Grunske, L., & Grundy, J. (2014). Dimensions and metrics for evaluating recommendation systems. In *Recommendation systems in software engineering* (pp. 245-273). Springer Berlin Heidelberg.

Bachman, M. (2015). *Recommendations with Neo4j and Graph-Aided Search.* Retrieved on 4 July 2016 from http://graphaware.com/neo4j/2015/09/30/recommendations-with-neo4j-and-graph-aided-search.html

Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A software architect's perspective*. Addison-Wesley Professional.

Bell, S. T. (2007). Deep-level composition variables as predictors of team performance: a meta-analysis. *Journal of applied psychology*, *92*(3), 595.

Bell, R. M., & Koren, Y. (2007, August). Improved neighborhood-based collaborative filtering. In *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 7-14). sn.

Bechky, B. A. (2006). Gaffers, gofers, and grips: Role-based coordination in temporary organizations. *Organization Science*, *17*(1), 3-21.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109-132.

Bozdag, E. (2013). Bias in algorithmic filtering and personalization. *Ethics and information technology*, *15*(3), 209-227.

Breese, J. S., Heckerman, D., & Kadie, C. (1998, July). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* (pp. 43-52). Morgan Kaufmann Publishers Inc.

Bruijn, H. de, Voort, H. van der, Warmelink, H., Wendel de Joode, R. van, & Willems, N. (2014). *Nieuwerwets organiseren: Strategisch omgaan met de factor arbeid.* Assen: Van Gorcum.

Burke, R. (1999, July). Integrating knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce* (pp. 69-72).

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*(4), 331-370.

Cantador, I., Fernández-Tobías, I., Berkovsky, S., & Cremonesi, P. (2015). Cross-domain recommender systems. In *Recommender Systems Handbook* (pp. 919-959). Springer US.

Castells, P., Hurley, N. J., & Vargas, S. (2015). Novelty and diversity in recommender systems. In *Recommender Systems Handbook* (pp. 881-918). Springer US.

Castro, D., & McQuinn, A. (2015). Cross-border data flows enable growth in all industries. *Information Technology and Innovation Foundation*, 9-10.

Church, Z. (2017). *Platform strategy, explained.* Retrieved on 6 June 2020 from https://mitsloan.mit.edu/ideas-made-to-matter/platform-strategy-explained

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. (1999, August). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems* (Vol. 60).

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT press.

Covington, P., Adams, J., & Sargin, E. (2016, September). Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (pp. 191-198). ACM.

Cramer, H., Garcia-Gathright, J., Reddy, S., Springer, A., & Takeo Bouyer, R. (2019, May). Translation, tracks & data: an algorithmic bias effort in practice. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1-8).

Cummings, J. N. (2004). Work groups, structural diversity, and knowledge sharing in a global organization. *Management science*, *50*(3), 352-364.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *JAsIs*, *41*(6), 391-407.

Dekker, S. (2013, November 15). Toezegging over verdere ontwikkelingen open access van wetenschappelijke publicaties [Brief regering]. Retrieved on 17 January 2016 from https://www.tweedekamer.nl/kamerstukken/brieven_regering/detail?id=2013Z22375&did=2013D45933

Deloitte. (2016). *Global Human Capital Trends 2016: The new organization: Different by design.* Deloitte University Press.

De Rond, M. (2012). Why less is more in teams. *Harvard Business Review*, *224*.

DeRosa, D., & Lepsinger, R. (2010). *Virtual team success.* Pfeiffer.

Dym, C.L., Little, P. & Orwin, E. (2014). *Engineering Design: A Project-Based Introduction, Fourth Edition.* Wiley.

Fenwick, M., McCahery, J. A., & Vermeulen, E. P. (2019). The end of 'corporate'governance: hello 'platform'governance. *European Business Organization Law Review*, *20*(1), 171-199.

Ge, M., Delgado-Battenfeld, C., & Jannach, D. (2010, September). Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems* (pp. 257-260). ACM.

Gibson, C. B., & Cohen, S. G. (Eds.). (2003). *Virtual teams that work: Creating conditions for virtual team effectiveness*. John Wiley & Sons.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61-70.

Goodin, R. E. (1996). Institutions and their design. *The theory of institutional design*, 1-53.

GraphAware. (2015). Products. Retrieved on 2 September 2015 from http://graphaware.com/products/

Gripenstraw, K. (2020). How We Experience Anxiety Today. *Harvard Business Review*. Retrieved on 7 June 2020 from https://hbr.org/2020/05/how-we-experience-anxiety-today

Gunes, I., Kaleli, C., Bilge, A., & Polat, H. (2014). Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review*, *42*(4), 767-799.

Hackman, J. R. (1983). A normative model of work team effectiveness (No. TR-2). *Yale School of organization and Management.*

Hajian, S., Bonchi, F., & Castillo, C. (2016, August). Algorithmic bias: From discrimination discovery to fairness-aware data mining. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 2125-2126).

Harrison, D. A., Price, K. H., Gavin, J. H., & Florey, A. T. (2002). Time, teams, and task performance: Changing effects of surface-and deep-level diversity on group functioning. *Academy of management journal*, *45*(5), 1029-1045.

Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000, December). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (pp. 241-250). ACM.

Herlocker, J., Konstan, J. A., & Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, *5*(4), 287-310.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 75-105.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, *19*(2), 4.

Hidalgo, C. A., & Hausmann, R. (2009). The building blocks of economic complexity. *proceedings of the national academy of sciences*, *106*(26), 10570-10575.

Hindle, T. (2008). *Guide to management ideas and gurus* (Vol. 42). John Wiley & Sons.

Hobday, M. (2000). The project-based organization: an ideal form for managing complex products and systems?. *Research policy*, *29*(7), 871-893.

Holton, J. A. (2001). Building trust and collaboration in a virtual team. *Team performance management: an international journal*, *7*(3/4), 36-47.

Ingham, A. G., Levinger, G., Graves, J., & Peckham, V. (1974). The Ringelmann effect: Studies of group size and group performance. *Journal of Experimental Social Psychology*, *10*(4), 371-384.

Janssen, M.F.W.H.A. (2009). Framing Enterprise Architecture: A meta-framework for analyzing architectural efforts in organizations. In: *Coherency Management: Architecting the Enterprise for Alignment, Agility and Assurance*, Authorhouse.

Jarvenpaa, S. L., & Leidner, D. E. (1998). Communication and trust in global virtual teams. *Journal of Computer-Mediated Communication*, *3*(4).

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, (8), 30-37.

Langfred, C. W. (2004). Too much of a good thing? Negative effects of high trust and individual autonomy in self-managing teams. *Academy of management journal*, *47*(3), 385-399.

Lehmann, M., Biørn-Hansen, A., Ghinea, G., Grønli, T. M., & Younas, M. (2015, August). Data Analysis as a Service: An Infrastructure for Storing and Analyzing the Internet of Things. In *International Conference on Mobile Web and Information Systems* (pp. 161-169). Springer International Publishing.

Lakhani, K. R., & von Hippel, E. (2003). How open source software works:"free" user-to-user assistance. *Research Policy*, *32*, 923-943.

Lee, M. Y., & Edmondson, A. C. (2017). Self-managing organizations: Exploring the limits of less-hierarchical organizing. *Research in organizational behavior*, *37*, 35-58.

Lemire, D., & Maclachlan, A. (2005, April). Slope One Predictors for Online Rating-Based Collaborative Filtering. In *SDM* (Vol. 5, pp. 1-5).

Levy, P., Bessant, J., Tranfield, D., Smith, S., & Ley, C. (1993). Organization design implications of computer-integrated technologies. *International Journal of Human Factors in Manufacturing*, *3*(2), 169-182.

Linden, G. D., Jacobi, J. A., & Benson, E. A. (2001a). *U.S. Patent No. 6,266,649*. Washington, DC: U.S. Patent and Trademark Office.

Linden, G. D., Jacobi, J. A., & Benson, E. A. (2001b). *European Patent No. 1,121,658*. Munich: European Patent Office.

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, *7*(1), 76-80.

Linstone, H. A. (2002). Corporate planning, forecasting, and the long wave. *Futures*, *34*(3), 317-336.

Lopez, K. (2015). *Your Master Data Is a Graph: Are You Ready?* InfoAdvisors.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002, July). Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI* (pp. 187-192).

Melville, P., & Sindhwani, V. (2011). Recommender systems. In *Encyclopedia of machine learning* (pp. 829-838). Springer US.

Mintzberg, H. (1979). The structuring of organization. *A Synthesis of the Research. Englewood Cliffs, NJ*.

Moede, W. (1927). Die Richtlinien der Leistungs-psychologie. *Industrielle Psychotechnik*.

Nelson, R. R., & Winter, S. G. (1997). An evolutionary theory of economic change. *Resources, Firms, and Strategies. A Reader in the Resource-Based Perspective. Oxford UAS*, 82-99.

Neo4j (2016a). Neo4j – Open Source, Big Community. Retrieved on 2 January 2016 from http://neo4j.com/open-source-project/

Neo4j (2016b). Neo4j and Apache Spark. Retrieved on 4 July 2016 from https://neo4j.com/developer/apache-spark/

Nijhuis, R. (2006). Enterprise Architectures as a Guideline for the Implementation of Strategy. *Realising Sustainable Performance Improvement for Pension Providers and Insurance Companies DCE Holding b.v., Schiphol.*

Ogul, H., & Ekmekciler, E. (2012, June). Two-way collaborative filtering on semantically enhanced movie ratings. In *Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference on* (pp. 361-366). IEEE.

O'Hara-Devereaux, M., & Johansen, R. (1994). *Globalwork—Bridging Distance, Culture, and Time.* Jossey-Bass, San Francisco.

Offermann, P., Blom, S., Schönherr, M., & Bub, U. (2010, June). Artifact types in information systems design science–a literature review. In *International Conference on Design Science Research in Information Systems* (pp. 77-92). Springer, Berlin, Heidelberg.

Pariser, E. (2011). *The filter bubble: What the Internet is hiding from you*. Penguin UK.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, *13*(5-6), 393-408.

Peart, S., & Levy, D. M. (2009). *The" vanity of the philosopher": From equality to hierarchy in post-classical economics*. University of Michigan Press.

Pinch, S., Henry, N., Jenkins, M., & Tallman, S. (2003). From 'industrial districts' to 'knowledge clusters': a model of knowledge dissemination and competitive advantage in industrial agglomerations. *Journal of economic geography*, *3*(4), 373-388.

Porter, M. E. (2000). Location, competition, and economic development: Local clusters in a global economy. *Economic development quarterly*, *14*(1), 15-34.

Powell, W. (2003). Neither market nor hierarchy. *The sociology of organizations: classic, contemporary, and critical readings*, *315*, 104-117.

Ranney, J., & Deck, M. (1995). Making Travis work: Lessons from the leaders in new product development. *Planning Review*, *23*(4), 6-12.

Raymond, E. (1999). The cathedral and the bazaar. *Knowledge, Technology & Policy*, *12*(3), 23-49.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994, October). GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (pp. 175-186). ACM.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, *40*(3), 56-58.

Rijksoverheid (2020, March 20). Nieuwe maatregelen tegen verspreiding coronavirus in Nederland. Retrieved on 12 April 2020 from https://www.rijksoverheid.nl/actueel/nieuws/2020/03/12/
nieuwe-maatregelen-tegen-verspreiding-coronavirus-in-nederland

Said, A., & Bellogín, A. (2014, October). Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems* (pp. 129-136). ACM.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285-295). ACM.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002, December). Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science* (pp. 27-28).

Schwartz, B. (2000). Self-determination: The tyranny of freedom. *American psychologist*, *55*(1), 79.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook* (pp. 257-297). Springer US.

Schumpeter, J. A. (1939). *Business cycles* (Vol. 1, pp. 161-74). New York: McGraw-Hill.

Simon, H. A. (1991). Bounded rationality and organizational learning. *Organization science*, *2*(1), 125-134.

Sinha, R., & Swearingen, K. (2001, June). Comparing Recommendations Made by Online Systems and Friends. In *DELOS workshop: personalisation and recommender systems in digital libraries* (Vol. 1).

solid IT. (2016). *DB-Engines Ranking.* Retrieved on 6 March 2016 from http://db-engines.com/en/ranking

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, *2009*, 4.

Swearingen, K., & Sinha, R. (2001, September). Beyond algorithms: An HCI perspective on recommender systems. In *ACM SIGIR 2001 Workshop on Recommender Systems* (Vol. 13, No. 5-6, pp. 1-11).

Tang, J., Hu, X., & Liu, H. (2013). Social recommendation: a review. *Social Network Analysis and Mining*, *3*(4), 1113-1133.

Townsend, A. M., DeMarie, S. M., & Hendrickson, A. R. (1998). Virtual teams: Technology and the workplace of the future. *The Academy of Management Executive*, *12*(3), 17-29.

Trewin, S. (2000). Knowledge-based recommender systems. *Encyclopedia of library and information science*, *69*(Supplement 32), 180.

Utrecht University (2019). Abstracts of the 6th International Workshop on the Sharing Economy. Retrieved on 4 June 2020 from https://6thiwse.sites.uu.nl/program-2/abstracts/

Venkatraman, N., & Henderson, J. C. (1998). Real strategies for virtual organizing. *Sloan management review*, *40*(1), 33-48.

Visser, J. (2015). *Building Maintainable Software.* O'Reilly Media, Inc.

Walsh, D. (2020). How to manage the hidden risks in remote work. *MIT Sloan.* Retrieved on 8 June 2020 from: https://mitsloan.mit.edu/ideas-made-to-matter/what-does-remote-work-mean-to-you-workers-share-experiences

Walther, J. B. (1997). Group and interpersonal effects in international computer-mediated collaboration. *Human Communication Research*, *23*, 342-369.

Wang, J., De Vries, A. P., & Reinders, M. J. (2006, August). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 501-508). ACM.

Wikimedia Commons (2009). Simplified Kondratieff Wave Pattern. Retrieved on 26 October 2016 from https://en.wikipedia.org/wiki/File:Kondratieff_Wave.svg

Xiang, L. (2011). Hulu's Recommendation System. *Retrieved 14 August* 2015.

Zenger, T. R., & Hesterly, W. S. (1997). The disaggregation of corporations: Selective intervention, high-powered incentives, and molecular units. *Organization Science*, *8*(3), 209-222.

Appendices

Appendix A | Article

Filtering algorithms for team formation

Matching supply and demand in the workplace

Maurits van der Goes, June 25th, 2020

Abstract

This paper presents the design and comparison of four filtering algorithms for self-managing team formation. Organizations introduce self-organization to reduce dependency on managers, subsequently, workers need support in the new team formation process. There is no known research on information filtering during this process. This design science research developed algorithm artifacts to compare demographic, knowledge-based, user-based collaborative filtering, and item-based collaborative filtering. User-based collaborative filtering has important advantages and is valued best by the interviewed users. Algorithmic bias is a governance limitation for implementing this in the marketplace for team formation. The research adds extensions to original theory, new meta-artifacts, and experiences gained. For society, the filtering algorithm saves time of managers in finding worker-activity matches and increases the workers' happiness in selecting from activities of interest. Stimulating a productivity increase.

Introduction

Organizational structures are changing. Hierarchies are replaced by flat organizations. This is a shift from central to decentral coordination (Bechky, 2006). Workers need support with this new responsibility of self-organization. A digital marketplace can unite the supply and demand of self-managing teams. However, this accumulation of work also creates an information overload and bounded rationality for users. This can be reduced by the introduction of a recommender system that applies filters to the available teams (Bobadilla et al., 2013). Only the most relevant teams for a user are shown. The core of the recommender system is the algorithm. An algorithm is a sequence of automatic step-by-step rules (Cormen et al., 2009). Algorithms became part of people's daily lives. For example, Google's search engine runs the PageRank algorithm, and the Elo-rating is indispensable for the chess world, but also for Tinder. A recommender system can be driven by several other algorithm types (Melville et al., 2002). Burke (2002) distinguishes five recommendation techniques: collaborative filtering, content-based, demographic, utility-based, and knowledge-based recommendation. Next to a random filtering algorithm, four options are further explored: demographic filtering, knowledge-based filtering, user-based collaborative filtering, and item-based collaborative filtering. The algorithm must create a filter by awarding a score to several teams. This can also be a combination of algorithms, but for evaluation purposes, each algorithm is tested separately. With design science research known algorithms adapted to this organizational context.

The algorithms were designed as part of a larger design science research. The design objective is to support users in self-managing team formation with a recommender system and custom filtering algorithms. This is achieved with design artifacts like personalization, recommender system, custom filtering algorithms, and validation queries. The designed system was built with the requirements of the program of business demands. The constraints for the recommender system with custom algorithms are personalized recommendations for each worker and the incorporation of platform rules. The objectives are regular updates, scalable, open-source, and easy to set up. This article focusses on the design and comparison of four information filtering algorithms for self-managing team formation.

Recommendation algorithms

The six modules in the recommendation framework are recommendation engine, post-processor, blacklist, filter, config, and logger. The context determines if the recommendations can be served to a user based on predefined rules. It consists of the Blacklist and the Filter modules. These modules incorporate the platform rules of the Part-up marketplace. That means that those modules are the same for all five algorithms.

The ExistingRelationships element checks if the user is already a partner or creator of a team. These teams should not be suggested. Being a supporter of a team does not affect the recommendations list. Even while there already is a relationship it makes sense to still recommend the team for two reasons. The goal of the recommender system is to suggest teams to become a partner in and not just a supporter. Swearingen and Sinha (2001) discovered that the confidence of users in the recommender system increases when some recommendations are familiar.

The marketplace has a couple of platform rules to offer private teams next to the public teams. The closed teams cannot be suggested to all users. The FilterOutPrivate element inspects if a team can be joined by a user. There are five privacy options for a team: open, closed, part of an open network, part of an invitational network and part of a closed network. The privacy restrictions for teams that are part of a network are covered by the FilterOutNetwork element. Only the private teams without a network are filtered out where the user is not already active in as a supporter. As mentioned, pending invitations for a team are neglected by this filter. Teams in a network can only be suggested if this network is open or the user is a member of the invitational or closed network. The FilterOutNetwork element verifies this relationship.

To exclude teams with a passed end date, the FilterOutEnded element was included. The FilterOutDeactivated element was developed to exclude deactivated teams. Each team holds the Boolean attribute active which can be set to false for deactivation. This is not an automatic process. Thus, teams can have a passed end date but still categorized as active. Both filters are written in Java. Applying these filters, 287 teams can be recommended, because these have an active status and an end date after 27 January 2016. This was the start date of the user interviews. For this reason, the dataset including the recommendations was frozen on this date.

The post-processor elements influence the generated list with recommendations more gently. Instead of excluding teams, the score of some teams is positively or negatively adjusted. The DF-, KBF- and UBCF-algorithms apply the five post-processor elements. The RAND-algorithm must be strictly random. The IBCF-algorithm cannot use these post-processor elements, because it calculates the initial score of each recommended team differently. All post-processor elements are written in Java. The code is included in each GitHub repository. GitHub is a digital platform for code sharing and collaboration. Table 1 presents an overview of the five post-processor elements. The attribute is added to the RECOMMEND-relation between a user-team pair for the total score. The score can be negative or positive and multiple distributions are supported. Each element is discussed in the following four paragraphs.

Attribute	Score range	Distribution
Penalize		
Team size	[-5, 0]	Pareto
Reward		
Same city	[0, 5]	Degenerate
Same country	[0, 1]	Degenerate
Same language	[0, 1]	Degenerate
Same tags	[0,50]	Uniform distribution
Total:	[-5, 57]	

Table 1: Post-processing score

The PenalizeTeamSize element negatively influences the recommendation of large teams following the mentioned Ringelmann effect. For the team size, only the creator and partners count as team members. The supporters are ignored because they are not active in activities. This element awards a negative score based on a Pareto-distribution starting at 5 team members. With the 80% point at 13 team members. 612 (93%) teams have five team members or less and almost all teams have 13 or fewer team members (645; 98%). The maximum negative score is five points.

There are two geographical post-processing elements: RewardSameCity and RewardSameCountry. If a user and a team are located

in the same city the total score is increased with five points. For the same country, one point is awarded. Also, if the user and team are located in the same city. It should be noted that only 376 (14%) users and 563 (86%) teams hold geographical information of a city and a country. This strongly limits the applicability of these elements.

The RewardSameLanguage element increases the score with one point when a user and team have the same language. All users and all teams hold the 'language'-attribute. The user language is chosen by the user. The team language is determined by analyzing the title and description via a Google Cloud service. English and Dutch are by far the most popular languages in the marketplace. These are also the only two languages that are currently present within both the users and teams.

The RewardSameTags searches for similar tags of a user and team. It was hard to design an effective element to compare the tags because a user is free to choose them. The marketplace does not contain a bag-of-words model. This is a method to steer a user voluntarily towards tags that were already submitted by other users. If the tag differs, the user is still free to submit and pick this new tag. This especially prevents multiple tags that mean the same but have a couple of different letters. To restrict the development complexity a full string match and not a fuzzy match was chosen. This strongly reduces the chance of a match; however, this specific element was not identified as crucial for this research. For the same reason was synonymy not addressed in this element. This even further decreases the chance of a match between a user and a team based on tags. It should also be noted that half of the users do not even have tags added to their profile. For each tag that matches ten points are rewarded. With a maximum score of 50 points. Compared to the other elements this a very high reward, but as described the chance of an exact tag match is very small. When such a match occurs, this indicates a shared interest.

Also, the logger modules are the same for each of the developed algorithms. Two loggers present information on the activities by the recommendation framework. The RecommendationLogger shows the entity for which the recommendation is produced. The StatisticsLogger shows the composition of the recommendation score.

Algorithms can be executed with a memory-based or model-based approach. The advantages of memory-based collaborative filtering are according to Breese et al. (1998): the good quality of predictions, the simplicity of implementing this algorithm, and the ability to update the database as the algorithm uses the entire database every time. Although they also see disadvantages like using the entire database takes time, an extensive database requires large available memory and it does not generalize the data and thereby overfits. Due to this lack of offline calculations, memory-based collaborative filtering is now considered unpractical for large datasets (Linden et al., 2003). In the research setup, all algorithms are run with a model-based approach.

Random filtering algorithm

The random filtering (RAND) algorithm produces complete random recommendations. This is why this RAND-algorithm can function as a control algorithm. For collecting the random recommended teams, a Java method of the GraphAware framework is used. The generated list is also not post-processed. Only the blacklist and the regular filters are applied.

Demographic filtering algorithm

The technique of demographic filtering uses chosen demographic attributes such as language, location, and preferences to filter the available teams. Pazzani (1999) amongst others conclude that demographic information can be used to identify which items a user likes. The recommendation engine of the demographic filtering (DF) algorithm is the same as the RAND algorithm, but by using all described post-processor elements a filter on demographic information is realized.

Knowledge-based filtering algorithm

The technique of knowledge-based filtering applies knowledge engineering to the reason which items meet the requirements of a user (Trewin, 2000). This knowledge engineering process involves understanding the specific features of users and teams. Investing in understanding the features and requirements prevents a cold start, but also results in

a static recommendation ability (Burke, 1999). This knowledge-based filtering (KBF) algorithm uses the results of a personality test. Human resource specialist Meurs HRM developed a personality test for the Part-up marketplace. Based on ten questions a user receives two out of five available individual strengths: influential strength, managerial strength, mental strength, social strength, and personal strength. As Sinha and Swearingen (2001) argued, in exchange for better ratings, users are willing to fulfill these kinds of system requests for more information but only if the outcome is likely to improve.

User-based collaborative filtering algorithm

The DF- and KBF-algorithms strongly depend on a couple of chosen attributes. This static structure makes the algorithm domain dependent. As a result, every time the interests of the users change the algorithms need to be adjusted. From a scalability perspective, a domain-independent algorithm is preferred. Independent of the domain is the purpose of collaborative filtering to find user-item pairs based on the user's interests and interaction. This replaced straightforward correlation statistics and predictive modeling (Melville & Sindhwani, 2011). The technology was further developed by Amazon for shopping suggestions. Later similar technology was implemented and developed by Facebook, Google, Hulu, LinkedIn, Netflix, and Twitter to serve their customers.

Collaborative filtering holds great momentum as multiple researchers describe it as one of the most successful approaches for a recommender system (Lemire & Maclachlan, 2005; Su & Khoshgoftaar, 2009). It delivers the most complete package with its ability to identify cross-genre niches, the lack of required domain knowledge, quality improvements and sufficient implicit feedback. The main advantage of collaborative filtering is that it performs well while a user or team holds limited content, but has relationships with other entities (Melville et al., 2002). Koren et al. (2009) agree by stating that technology is domain independent. This aspect means that algorithms can be applied to various subjects without adjustments. A second advantage is that it can deliver serendipitous recommendations. Melville et al. (2002) describe this as relevant recommendations of items (teams), although not matched on content supplied by the user via its profile. Multiple researchers conclude that these two reasons led to the successes of recommender systems in different domains (Goldberg et al., 1992; Resnick et al., 1994).

According to Koren et al. (2009), there are two main areas of collaborative filtering: neighborhood methods and latent factor models. The former focuses on the relationships between users and teams. User-based collaborative filtering is based on this principle. The latter calculates the user's appreciation of a team by analyzing their activity. Item-based collaborative filtering is a successful implementation of this approach.

User-based collaborative filtering is by Wang et al. (2006) illustrated as the *"Word of Mouth"* phenomenon. This is the traditional version of collaborative filtering where team suggestions are generated based on users with similar interests (Breese et al., 1998). In figure 1 is a simplified version of the user-based collaborative filtering approach visualized for users active in teams. To generate recommended teams the algorithm follows a four-step process. First, the algorithm inspects in which teams a user is active. Second, for each team, the algorithm inspects which team members are active in that team. Third, the algorithm gathers the other teams these team members are active. Fourth and final, these discovered teams are ranked by the algorithm. All these items and relationships are usually merged in a table. However, as figure 1 shows, this method is very suitable for path querying in a graph database.

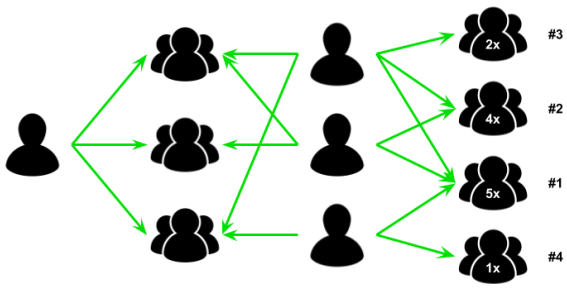

Figure 1: User-based collaborative filtering for teams.

The core of the collaborative filtering algorithm is the NearbyTeams element. The approach is shown in figure 1. For each discovered path the average of the three roles is awarded to the team. The value can vary between 1 and 2. For

example, if the path consists of a creator, partner, and supporter in the two teams. Then the score is 1.5 points. Further supported by the NearbyNetworkTeams element. This second element analyses in which the other members of a network are active as a creator or partner. For each discovered path a score of 0.5 points is awarded to the team. The code of this algorithm is reviewed by a lead data scientist of GraphAware and a developer of Part-up. The code of this algorithm is published on GitHub[13] under a GPLv4 open source license.

Item-based collaborative filtering algorithm

User-based collaborative filtering algorithms suffer scalability issues (Sarwar et al., 2001). Linden et al. (2003) addressed this problem especially while designing the item-based collaborative filtering algorithm for Amazon. It scales independently of the number of users or items (teams) on their marketplace by building a similar-items offline table. While the recommendation framework still runs online.

For the recommender system, the relationships between a user and a team are defined by several attributes. The input data of the user on a team can be split into two types: explicit feedback and implicit feedback (Koren et al., 2009; Xiang, 2011). The explicit feedback is the most valuable one because this is the rating a user gives to a team. However, the availability of this information is low, as users give only ratings to fellow team members after they finished a collaborative task. This is usually at the end of a team. The implicit feedback includes browsing history, mouse movements, page visits, search patterns, or other interactions. This user preference can be modeled as a pseudo rating with this matrix factorization. This information is used by the collaborative filtering algorithms to produce a list of raw recommendations. These cannot be distributed to the users without filtering and ranking (Linden et al., 2003).

Therefore, Amazon's developers Linden et al. (2003) developed an item-to-item approach for collaborative filtering. This collaborative filtering version focuses on the rating distributions per item (team). Usually, a digital marketplace holds more users than items. As the user rating of an item is expected to be stable, this model-based algorithm can create a model of the database. That is why the algorithm does not have to identify the neighborhood of similar users first. Thus, the recommender system can produce much faster results note Sarwar et al. (2001). The first step is the analysis of the user-item matrix as presented in figure 1. The similarity between items is often determined by the methods: cosine-based similarity (vector-based similarity), correlation-based similarity (Pearson-r) or adjusted cosine similarity.

As the collaborative filtering algorithms still require a rating for each relationship between a user and a team, this must be composed of a defined formula. For the Part-up marketplace, a participation formula (formula 1) was developed that combines four attributes from the logical data model: ratings, role, contributions, and comments. The selection and the weight of these attributes were determined based on the problem definition. The ratings (*'rat'*) a user receives on his activities in a team by another user is the most valuable information, as this is the only available explicit feedback on the quality of a user's activities in a team. When a user receives high ratings he is actively participating in that team. Each rating is divided by twenty to transform it to a scale of one to five. Next to these available explicit ratings also an implicit rating is constructed based on three attributes that every user-team-relation contains: role, number of contributions, and number of comments. The role a user holds in a team is expressed following an ordinal scale (creator: 2 points, partner: 1,5 points, supporter: 1 point). The number of contributions (*'cont'*) the user has in the team is divided by the maximum number of contributions (*'maxCont'*) this user has in all of its teams. The same calculation is chosen to normalize the number of comments (*'com'*). Note that the normalization is applied with a smoothing term to prevent the dividing by zero when a user has no contributions or comments. The normalized contributions are weighted double compared to the normalized comments because a contribution is a commitment to a team and scarcer than a comment. The implicit rating is the sum of the role-weight, normalized contributions and normalized comments and results in a value between

[13] https://github.com/part-up/UBCF-algorithm

one and five. The average of these explicit ratings and implicit rating is the participation score of a user in a team. That means that when the number of explicit ratings increases, the influence of the implicit rating on the participation score directly decreases. A high predicted participation score is a high likelihood for this user to actively participate in a team. With this formula, the similarity between teams can be calculated to perform item-based collaborative filtering.

$$R_u = \frac{\sum_{i=1}^{rat} \frac{rat(i)}{20} + role + \frac{cont}{maxCont} \cdot 2.0 + \frac{com}{maxCom}}{rat + 1}$$

Formula 1: Participation formula.

Sarwar et al. (2001) advise three types of similarity to compare items (teams). For this research, the Pearson correlation-based similarity (Formula 2) is chosen. This formula corrects the result with the average rating of all users on a team. Not only the ratings of the users that are active in both teams. With the requirement that there is a minimum of 3 users to determine the similarity. Between two teams there can only be one similarity relationship. There is no maximum in the number of similar relationships that a team can hold. The discovered relationships between the teams are used to indirectly compute ratings to base the recommendations (figure 2). Algorithms that can be applied for the model building are according to Sarwar et al. (2001) for example Bayesian network, clustering, and rule based. In this research, the rule-based approach is chosen. The prediction of the user rating of a team is calculated based on the user ratings on similar teams.

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}}$$

Formula 2: Pearson correlation-based similarity (Sarwar et al., 2001).

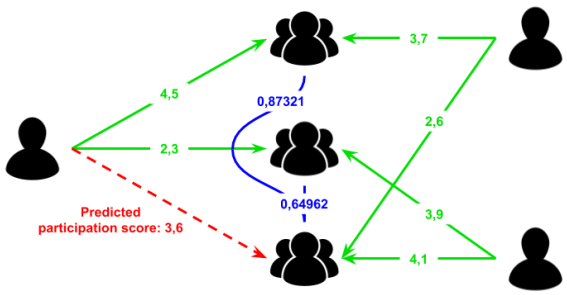

Figure 2: Item-based collaborative filtering (Some relationships are left out for overview reasons).

This is called the model learning stage, followed-up by the recommendation stage. This model is used to make predictions (Breese et al., 1998). The user gets teams suggested that are similar to teams this user has interacted with earlier. This prediction is either a weighted average or a regression according to Sarwar et al. (2001). For this research, the weighted average was used.

The item-based collaborative filtering approach has three disadvantages. First, when an entity changes the model needs to be updated. Second, calculating the similarities between teams is complex. Third, with data sparsity the system performance is weak. The code of this algorithm is published on GitHub[14] under a GPLv4 open source license.

As described, there are multiple information filtering techniques to apply to this case of self-managing teams. It has been mentioned that there is little research in these fields. That is why these algorithms first must be tested with validation experiments. Although some algorithms are only slightly different, it can still lead to different results. Hence, it is important to compare multiple algorithms. Table 5.6 shows the advantages and disadvantages of the four algorithms. Random filtering is not covered by this comparison as it operates as a control algorithm.

[14] https://github.com/part-up/IBCF-algorithm

	Advantage	Disadvantage
RF	(control)	(control)
DF	No cold start	Domain dependent Static profile Only demographic values included
KBF	Based on soft skills (personality test) Cold start mitigation	Domain dependent Static profile Perfect composition per team unknown
UBCF	Domain independent Performs well with entities that have relationships with other entities, even with limited entity content Dynamic profile with team frequency Serendipitous recommendation	Cold start (data sparsity) Unstable rating
IBCF	Domain independent Scales independently Detailed, dynamic profile with participation score Stable rating	Frequent update of the full model Complex similarity calculation Cold start (data sparsity) Formerly patented

Table 2: Algorithm comparison.

The demographic filtering and knowledge-based filtering algorithms have the advantage that their approach does not suffer from a cold start or mitigates this. The other two collaborative filtering algorithms require user activity before they can generate any recommendations. Both the demographic and knowledge-based filtering algorithms are domain-dependent, which means that they do not automatically adapt to changes in the interests of users. An aspect that is successfully addressed in the design of the collaborative filtering algorithms. These algorithms also have dynamic profiles based on either the team frequency for user-based filtering or the participation score for item-based filtering. User-based collaborative filtering performs well with teams that hold little information but have several active team members. As a result, this algorithm generates serendipitous recommendations. If the team is just created it suffers from the cold start problem, hence it is hard to recommend. Item-based collaborative filtering is expected to be a stable predictor of a user's interest in a team. The stability of this rating is essential for scaling a collaborative filtering algorithm for a large volume of users. Otherwise frequent updates of the full model are required. This requires calculating the team similarity, a complex and thus resource-intensive process. This comparison shows that all algorithms have different advantages and disadvantages. However, none of the algorithms is unsuitable or a clear winner based on this comparison. In the evaluation, the algorithms are further and extensively compared.

Evaluation

The evaluation of the five algorithms is divided into two parts: quantitative validation and user interviews. Multiple researchers (Shani and Gunawardana, 2011; Avazpour et al., 2014) name twelve dimensions to validate a recommender system. Of these dimensions are coverage, privacy, trust, user-preference, and utility chosen. The other dimensions are not chosen for various reasons. Primarily because the offline research setup with static data lacks user interaction. Although some algorithms are only slightly different, it can still lead to different results. Hence, it is interesting to compare multiple algorithms. It would be even more interesting to improve the performance of each algorithm by changing the score profile. However, this falls outside the scope of this research.

Quantitative validation

The quantitative validation analyses the recommendations from a numerical point of view. This validation consists of the coverage and privacy dimensions. Coverage is the proportion of available data that can be used for generating recommendations. It is an indication of the quality of the recommendations because according to Shani and Gunawardana (2011) recommender systems that cover a wider range of users or teams are preferred. When a user or team holds insufficient information, it is impossible to project a recommendation. In technical terms, this is called a cold start. This is related to the low accuracy in the prediction. The coverage dimension can be split into two types. User-space coverage (prediction coverage) is the percentage of users that a recommender system can generate predictions for. Item space coverage

(catalog coverage) is the percentage of teams (items) that are recommended to a user by the system. In this validation, the item space coverage is less important for two reasons. This research is from the user perspective. And the configuration is set to a maximum of five recommendations for each user. The maximum of 13.715 recommendations is not necessarily equally distributed over the 287 active teams with an end date after 27 January 2016. Again, is active the standard attribute setting that indicates that it is not deactivated. Still, both types are used in this quantitative validation.

Privacy is an important dimension if the service processes the personal data of users. The risk is that the privacy of these users and their teams is harmed. The external privacy is not validated in this research. The Dutch Data Protection Authority drafted guidelines to guarantee that personal data of users is protected. All organizations in the Netherlands that store, or process personal data must comply with the seven requirements of the national Personal data protection law ('Wet bescherming persoonsgegevens'). Part-up has a privacy policy[15] that covers these requirements.

This validation is on internal privacy. The digital marketplace incorporated platform rules in order to enable private teams next to the public teams. The privacy of the teams is divided into five types. Each team holds a *'privacy_type'*-attribute with an integer value. Most teams are public and not part of a network. Some teams are private and not part of a network. These teams can only be accessed by invited users. Pending invitations are for complexity reasons not included in this initial version. That means that these private teams can only be recommended to their supporters. There are also public teams that might be part of a network. The same accounts for private teams. The public teams can be recommended to all users. Irrespective of the teams are part of a network. This means that only the private teams are validated on privacy. This dimension is measured by the number of recommended teams that should not be accessible.

First, the user-space coverage of the 21 interviewed users is discussed as shown in table 3. The last column *'Total'* reflects the coverage of all algorithms. That the RAND-algorithm scores best is no surprise because the random-aspect requires no specific data points to suggest a team. That should be the same for the DF-algorithm. Early in the interview process, a small error was discovered with this algorithm. Due to a configuration error, this algorithm produced no recommendation for one user. This was adjusted after the interview. This is why the coverage of the DF-algorithm is one case smaller than the RAND-algorithm.

User group	RAND	DF	KBF	UBCF	IBCF	Total
High	7	6	7	7	7	7
Medium	7	7	4	7	7	7
Low	7	7	2	4	0	7
Total interviews	21	20	13	18	14	21
	100%	95%	62%	86%	67%	100%
Total database	2475	2089	338	1514	764	2714
	90%	76%	12%	55%	28%	99%

Table 3: User-space coverage of the algorithms.

Second, the user-space coverage is validated for all users in the database with the active status. This is the standard status for all users and implies that the user is not deactivated on request. It does not represent the activity level of a user. The results are included in the bottom rows 'Total database' of table 3. The RAND-algorithm has a 90% user-space coverage. It is remarkable that it has not 100% coverage. The same counts for the DF-algorithm. Which only has a 76% user-space coverage. As both algorithms require no specific attributes or relationships, this must be an error in running the algorithms. The KBF-algorithm has a 12% user-space coverage. Which is very low. The algorithm requires that a user has discovered his two strengths via a small questionnaire. The questionnaire is voluntary and not well visible on the marketplace. This algorithm has a 100% user-space coverage for the users who have completed this questionnaire. The UBCF-algorithm has a 55% user-space coverage. Although it has a 97% user-space coverage for the users that are active in a team or member of a network. This should be 100%. An indication of a small error. The IBCF-algorithm has a 28% user-space coverage. Quite a low score. The algorithm has 87% user-space coverage for the users that are active in a team with a similarity relationship to another team. Reflecting on the user-space coverage for the

[15] privacy.part-up.com

algorithm the conclusion is that currently none of the algorithms can be used for the Part-up marketplace. The user-space coverage of the KBF- and IBCF-algorithms is too low. And the RAND-, DF-, UBCF- and IBCF-algorithms seem not to be running error-free. There is room for improvement. Also combining multiple algorithms into one hybrid algorithm can be the solution. Overall 99% of the users have at least one recommendation generated by one of the algorithms. The user-space coverage confirms that with a combination of multiple algorithms nearly all users receive recommendations.

The results of the item space coverage validation are included in table 4. Some users had multiple recommendations generated by the RAND- and DF-algorithm, because of configuration issues. It was not possible to run the algorithms at the same time. Each algorithm ran once in sequence. Each user can have a maximum of five recommendations. As mentioned, these recommendations are not necessarily evenly distributed. This explains why only three-quarters of the teams were recommended via the algorithms. This is a satisfying percentage.

Items	RAND	DF	KBF	UBCF	IBCF	Total
Total	179	173	48	141	65	211
	62%	60%	17%	49%	23%	74%

Table 4: Item space coverage of the algorithms.

For the validation of the privacy and platform rules a series of experiments were performed. There are no teams recommended in which a user was already active as a partner or creator. Nor are there any incorrect recommendations of private teams that are not part of a network. Further validation shows that there were no incorrect recommendations of teams that are part of a network. It can be concluded that the privacy of the teams and the platform rules are respected with the five context (blacklist and filter) elements.

User interviews

The user interviews perform a face validation of the recommender system. This is a subjectively view test to gather the opinion of non-experts. This validation must be performed in a representative simulator. When the outcome is positive, then this is an endorsement for the application of recommender systems. In the user interviews, the recommender system is validated for the trust, user-preference, and utility dimensions. Trust is the confidence of a user in the provided recommendation. Trust can be increased by combining known and unknown items. Or by delivering the recommendations with an explanation. Multiple incorrect recommendations decrease trust in the recommender system. The most common method to validate trust is asking the reasonability of recommendations in a user study. The user preference is its perception of the configuration of the algorithm or combination of algorithms that are running under the hood. A user study with recommendation lists generated by different algorithms is an appropriate way to test this. Utility is the value a user or the system gains from using the recommender system. The preference of recommendations by users can be determined by ranking the options.

For the interviews were users selected from three groups. This split is made because Avazpour et al. (2014) note that users experience the recommendations differently based on their previous activities. Current and former Part-up team members or developers are excluded from the validation because they may have too much knowledge of the designed algorithms to give an independent opinion. There are 27 Part-up team members excluded from the user base for this reason. To group the users in three categories an engagement score is calculated. Each role a user holds in a team is multiplied with the role-weight. The engagement score is an approximation of the engagement of a user with the Part-up marketplace. The three interview groups for low, medium and highly engaged users are presented in table 5. The two cut-off points for the groups were an arbitrary decision by this author.

Group	Engagement score	Cleaned total
Low engaged users	0	1608
Medium engaged users	0 <> 20.0	1089
Highly engaged users	≥ 20	19

Table 5: Three user groups based on the engagement score.

From each group randomly seven people are selected for a user interview by phone. This selection is not corrected for age, gender, background or location. Each interview is performed following a

drafted protocol. Shortly before the interview, the overview with the recommended teams was sent to ensure the first impression of the user. This overview resembles the Discover-page of Part-up. A couple of users expressed that they were not searching for teams at all. This was for several reasons. Some used the marketplace only for work, others were too busy with their regular daytime job or they had already too many tools for digital collaboration. They tried Part-up out of curiosity. The question if people are actually searching for teams was not included until the last two interviews. However, some of the users also answered the question by themselves. Two questions should have been part of the interview: *"What is your main purpose to use Part-up?"* and *"On a scale from 0-7, are you searching for new part-ups to join?".* The quality of the offered teams is also an issue. An often-mentioned reason for not joining a team is that it lacks a clear description and goal. Even though a user likes the recommendation because the subject matches his profile.

The trust in the recommender system with its custom algorithms was expressed with the interview question: *"Do these recommendations reflect your profile?"*. The average results per user group are included in table 6. For the first seven interviews, coincidentally also all highly engaged users, the valuation of the reflection of their profile was not expressed as a number on a scale of 0-7. This was later projected based on their recorded answers. The results of all user groups are hardly a surprise. The high user group is active in many teams. These are interesting data points for the recommender system, but also leaves fewer teams available to recommend. Especially when the catalog of teams is still relatively small. That is why their average projected score is halfway the range. Almost the same line of reasoning explains why the medium user group scores well. A sufficient number of data points, while still enough teams available. The low user group scores just below average. A bit disappointing and an indication that the algorithms do not perform well with little data points.

User group	All
High	3.6
Medium	5.1
Low	3.3
All	4.0

Table 6: Average reflection (scale 0-7) of the user's profile by the recommendations

The user-preference dimension was validated with two questions. The first question was: "*Which part-up is the most likely you will join as a partner?"*. This is an indication which recommendation they like the most. The sum of all first picks is included in table 7. The UBCF-algorithm scored by far the best. Even in the low user group three out of the seven interviewees chose it. Even though they are not active in a team yet. That is because the algorithm also uses networks in identifying similar users. That is why this algorithm scored best overall. Some interviewees confirmed this. They recognized teams they were co-initiator of or familiar with. The RAND-algorithm scores surprisingly good in the medium user group. This is important from a serendipity point of view. These serendipitous recommendations can uncover unknown interests. And this validation shows that these random recommendations are indeed appreciated. A final note about the score of the DF-algorithm in the low user group. This gives the confidence that successful recommendations can also be generated based on limited attributes a user and team hold. An important discovery for preventing a cold start among users that suffer of data sparsity.

User group	RAND	DF	KBF	UBCF	IBCF
High	0	0	2	3	2
Medium	3	0	0	4	0
Low	0	4	0	3	0
Total	3	4	2	10	2

Table 7: Sum of first picks to join as a partner.

The follow-up question for this dimension was: "*How would you rank this recommendations set yourself?"*. This gives a wider impression of the user preference for an algorithm (table 8). Again, the UBCF-algorithm scores best overall, but now also in all user groups. The DF-algorithm is a good second best. Especially for the low user group. Like also was noticed in the previous paragraph. The IBCF-algorithm scores surprisingly bad. Also, because it is equal with the RAND-algorithm. This latter algorithm, which functions partly as a control algorithm, scores exactly average. The disappointing score of the IBCF-algorithm is an indication that the participation formula is incorrect. Because this is the

core element in the generation of recommendations. The score of the KBF-algorithm is not good, because users do not recognize themselves. They expect to receive recommendations based on their profile and activity. Not especially on their role in a team. This shows that it is important to always present recommendations in the right context. For example, with a small explanation where recommendations are based on.

Multiple interviewees mentioned two comments, independent of a specific algorithm. First that they do not want to receive recommendations of teams they are already active in as a supporter. This contradicts the mentioned statement of Swearingen and Sinha (2001) that when some recommendations are familiar, the confidence in the recommender system increases. It shows that for these users, recommendations are about discovering new teams. Second, users expressed that some recommended teams were quite old and with minimal activity. They perceived teams that started weeks or months ago as less interesting. Both raised issues should be addressed before the implementation of the recommender system in the marketplace.

User group	RAND	DF	KBF	UBCF	IBCF
High	2.3	3.2	3.3	3.9	2.7
Medium	3.1	2.6	2.0	4.6	3.3
Low	3.4	4.3	3.5	4.5	-
All	3.0	3.4	2.9	4.3	3.0

Table 8: Average rank (1-5) of the algorithms.

The utility dimension was validated by reflecting on their first pick to join as a partner. The question was: "*On a scale from 0-7, will you join it?"*. Table 9 shows the results of this question. The two collaborative filtering algorithms perform the best. Where the rating of the UBCF-algorithm is most relevant because it is based on ten ratings, against the two for the IBCF-algorithm. The results for the RAND-algorithm confirm that these recommendations can be useful from the mentioned serendipity factor. The average of the three ratings is just below average. The DF-algorithm scores very low with four ratings of three and lower. This makes it doubtful that this algorithm can overcome data sparsity. The score of the KBF-algorithm confirms that the two users are unaware of their potential value to the team. Some interviewees also expressed that their limited time was a reason why they rated their first pick quite low.

User group	RAND	DF	KBF	UBCF	IBCF
High	-	-	3.5	5	5
Medium	3.33	-	-	4.5	-
Low	-	2	-	5.33	-
All	3.33	2	3.5	4.9	5

Table 9: Average rating (scale 0-7) to join the first pick as a partner.

Conclusions

Reflecting on this research it can be concluded that a recommender system with these custom algorithms is valued by users while being technically functional. A combination of multiple types of algorithms has arisen as a promising method. Multiple researchers (Ogul & Ekmekciler, 2012; Melville et al., 2002) argue that a combination can overcome limitations that are imposed by the data. Leading to more stable results than collaborative filtering. Based on the results of the evaluation a hybrid version of the UBCF-algorithm and RAND-algorithm was chosen. The UBCF-algorithm performed best. And the RAND-algorithm provides a serendipitous element and addresses data sparsity. It covers 2.627 (96%) users while respecting privacy and platform rules. This hybrid algorithm provides qualitative recommendations to high, medium and low engaged users. Recommendations that users trust and can use. The straightforward algorithm structure without a complex similarity query makes it reliable, robust and scalable. The recommender system with this hybrid algorithm and a couple of adjustments is implemented into the marketplace.

References

Avazpour, I., Pitakrat, T., Grunske, L., & Grundy, J. (2014). Dimensions and metrics for evaluating recommendation systems. In *Recommendation systems in software engineering* (pp. 245-273). Springer Berlin Heidelberg.

Bechky, B. A. (2006). Gaffers, gofers, and grips: Role-based coordination in temporary organizations. *Organization Science*, *17*(1), 3-21.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109-132.

Breese, J. S., Heckerman, D., & Kadie, C. (1998, July). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence* (pp. 43-52). Morgan Kaufmann Publishers Inc.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*(4), 331-370.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT press.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*(12), 61-70.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, (8), 30-37.

Lemire, D., & Maclachlan, A. (2005, April). Slope One Predictors for Online Rating-Based Collaborative Filtering. In *SDM* (Vol. 5, pp. 1-5).

Linden, G. D., Jacobi, J. A., & Benson, E. A. (2001). *U.S. Patent No. 6,266,649*. Washington, DC: U.S. Patent and Trademark Office.

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE, 7*(1), 76-80.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002, July). Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI* (pp. 187-192).

Melville, P., & Sindhwani, V. (2011). Recommender systems. In *Encyclopedia of machine learning* (pp. 829-838). Springer US.

Ogul, H., & Ekmekciler, E. (2012, June). Two-way collaborative filtering on semantically enhanced movie ratings. In *Information Technology Interfaces (ITI), Proceedings of the ITI 2012 34th International Conference on* (pp. 361-366). IEEE.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, *13*(5-6), 393-408.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994, October). GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (pp. 175-186). ACM.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285-295). ACM.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook* (pp. 257-297). Springer US.

Sinha, R., & Swearingen, K. (2001, June). Comparing Recommendations Made by Online Systems and Friends. In *DELOS workshop: personalisation and recommender systems in digital libraries* (Vol. 1).

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, *2009*, 4.

Swearingen, K., & Sinha, R. (2001, September). Beyond algorithms: An HCI perspective on recommender systems. In *ACM SIGIR 2001 Workshop on Recommender Systems* (Vol. 13, No. 5-6, pp. 1-11).

Trewin, S. (2000). Knowledge-based recommender systems. *Encyclopedia of library and information science*, *69*(Supplement 32), 180.

Wang, J., De Vries, A. P., & Reinders, M. J. (2006, August). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 501-508). ACM.

Xiang, L. (2011). Hulu's Recommendation System. *Retrieved 14 August* 2015.

Appendix B | Data dictionary

Each attribute that is in the logical data model of either the research architecture or deployed architecture is included in the tables B.1 and B.2. Table B.1 contains all the attributes of the entities. The columns hold for these attributes the name, the type in MongoDB, type in Neo4j, scale of measure, part of the offline solution for research and part of the online solution for deployment. The *'<id>'*-attribute is automatically created by Neo4j for all entities and that is why it is not available in MongoDB. The sub-headers represent the labels of the entities to which the attributes belong to.

Attribute	MongoDB type	Neo4j type	Scale of measure	Offline	Online
City					
<id>	-	id	Nominal	✖	✖
_id	String	String	Nominal	✖	✖
name	String	String	Nominal	✖	✖
Country					
<id>	-	id	Nominal	✖	✖
name	String	String	Nominal	✖	✖
Network					
<id>	-	id	Nominal	✖	✖
_id	String	String	Nominal	✖	✖
language	String	String	Nominal	✖	✖
name	String	String	Nominal	✖	✖
tags	Array	String[]	Nominal	✖	✖
privacy	String	String	Nominal	✖	✖
Strength					
<id>	-	id	Nominal	✖	
code	Int32	int	Nominal	✖	
name	String	String	Nominal	✖	
Team					
<id>	-	id	Nominal	✖	✖
_id	String	String	Nominal	✖	✖
active	-	Boolean	Nominal	✖	✖
activity_count	Int32	int	Ratio	✖	✖
archived	-	Boolean	Nominal		✖
archived_at	Date	int	Interval		✖
deleted	-	Boolean	Nominal		✖
deleted_at	Date	int	Interval	✖	✖

end_date	Date	int / String	Interval	✖	✖
language	String	String	Nominal	✖	✖
link	String	String	Nominal	✖	
name	String	String	Nominal	✖	✖
partners	-	int	Ratio	✖	
phase	String	String	Nominal	✖	✖
privacy_type	Int32	int	Nominal	✖	✖
purpose	String	String	Nominal	✖	✖
tags	Array	String[]	Nominal	✖	✖
type	String	String	Nominal	✖	✖
User					
<id>	-	id	-Nominal	✖	✖
_id	String	String	Nominal	✖	✖
active	-	Boolean	Nominal	✖	✖
deleted	-	Boolean	Nominal		✖
deactivated_at	Date	int	Interval	✖	✖
name	String	String	Nominal	✖	✖
language	String	String	Nominal	✖	✖
name	String	String	Nominal	✖	✖
tags	Array	String[]	Nominal	✖	✖

Table B.1: Data dictionary entities

Table B.2 contains all the attributes of the relationships. The columns hold for these attributes the name, type in Neo4j, scale of measure, part of the offline solution for research and part of the online solution for deployment. The *'<id>'*-attribute is automatically created by Neo4j for all relationships. The other attributes do not exactly exist in MongoDB but are constructed with one or more attributes from MongoDB or are the scores of the recommender system. The sub-headers represent the labels of the relationships to which the attributes belong to.

Attribute	Neo4j type	Scale of measure	Offline	Online
ACTIVE_IN				
<id>	id	Nominal	✖	✖
comments	int	Ratio	✖	
contributions	int	Ratio	✖	
participation	float	Ratio	✖	
ratings	int[]	Ratio	✖	

role	int	Nominal	✖	✖
HOLDS				
<id>	id	Nominal	✖	
score	int	Interval	✖	
LIVES_IN				
<id>	id	Nominal	✖	✖
LOCATED_IN				
<id>	id	Nominal	✖	✖
MEMBER_OF				
<id>	id	Nominal	✖	✖
admin	Boolean	Nominal	✖	✖
colleague	Boolean	Nominal		✖
PART_OF				
<id>	id	Nominal	✖	✖
RECOMMEND				
<id>	id	Nominal	✖	✖
daysActive	float	Ratio		✖
nearbyTeams	int	Ratio	✖	✖
nearbyTeamsinNetworks	int	Ratio	✖	✖
partnersLimit	float	Ratio	✖	✖
pred_part	float	Ratio	✖	
sameCity	int	Nominal	✖	✖
sameCountry	int	Nominal	✖	✖
sameLanguage	int	Nominal	✖	✖
sameTags	int	Ratio	✖	✖
SIMILARITY				
<id>	id	Nominal	✖	
coefficient	float	Ratio	✖	

Table B.2: Data dictionary relationships

Appendix C | Validation queries

All Cypher queries to validate the different dimensions of the algorithms are included in this appendix. Cypher query C.1 returns the number of users that have recommendations of one algorithm. If a user has multiple recommendations by a recommendation this is counted as one. An example is chosen for the RAND-algorithm. Also, the total number of active users is returned.

```
MATCH (x:User)
WHERE x.active
WITH COUNT(x) AS Total
MATCH (u:User)
WHERE u.active
      AND (u)-[:RECO_RAND]->(:Team)
WITH COUNT(u) AS Coverage, Total
RETURN Coverage, Total
```

Cypher query C.1: User space coverage of the RAND-algorithm

Cypher query C.2 functions almost the same as the previous query. The only difference is that it returns the user-space coverage of all algorithms. Again, if a user has multiple recommendations, also from different algorithms, this still counts as one.

```
MATCH (x:User)
WHERE x.active
WITH COUNT(x) AS Total
MATCH (u:User)
WHERE u.active
      AND ( (u)-[:RECO_RAND]->(:Team)
      OR (u)-[:RECO_DF]->(:Team)
      OR (u)-[:RECO_KBF]-(:Team)
      OR (u)-[:RECO_UBCF]-(:Team)
      OR (u)-[:RECO_IBCF]-(:Team) )
WITH COUNT(u) AS Coverage, Total
RETURN Coverage, Total
```

Cypher query C.2: User space coverage of all algorithms

The item-space coverage is the percentage of teams that have a recommendation by an algorithm. Cypher query C.3 only inspects teams that are active with an end date after 27 January 2016. All teams that have an earlier end date are ignored by the algorithm regardless of whether these teams still hold the *'active'*-attribute. The coverage is the number of teams that are at least recommended once by an algorithm. For this example, is again the RAND-algorithm chosen The query also returns the total number of teams within the mentioned criteria.

```
MATCH (x:Team)
WHERE x.active
```

```
      AND x.end_date > 20160127
WITH COUNT(x) AS Total
MATCH (t:Team)
WHERE t.active
      AND t.end_date > 20160127
      AND (t)<-[:RECO_RAND]-(:User)
WITH COUNT(t) as Coverage, Total
RETURN Coverage, Total
```

Cypher query C.3: Item space coverage of the RAND-algorithms

Cypher query C.4 returns the item-space coverage for all algorithms and the total number of teams. Again, only for the teams that are active and have an end date after 27 January 2016. When a team is recommended by multiple algorithms it still counts as one.

```
MATCH (x:Team)
WHERE x.active
      AND x.end_date > 20160127
WITH COUNT(x) AS Total
MATCH (t:Team)
WHERE t.active AND t.end_date > 20160127
      AND ( (t)<-[:RECO_RAND]-(:User)
      OR (t)<-[:RECO_DF]-(:User)
      OR (t)<-[:RECO_KBF]-(:User)
      OR (t)<-[:RECO_UBCF]-(:User)
      OR (t)<-[:RECO_IBCF]-(:User) )
WITH COUNT(t) as Coverage, Total
RETURN Coverage, Total
```

Cypher query C.4: Item space coverage of all algorithms

To validate the ExistingRelations element Cypher query C.5 was written to execute in the web-interface of Neo4j or from code using its API. It returns the number of team recommendations where a user is already a partner or creator. These are incorrect.

```
MATCH (u:User)-[r]->(t:Team)
WHERE ( (u)-[:RECO_RAND]->(t)
      OR (u)-[:RECO_DF]->(t)
      OR (u)-[:RECO_KBF]-(t)
      OR (u)-[:RECO_UBCF]-(t)
      OR (u)-[:RECO_IBCF]-(t) )
      AND ( (u)-[:ACTIVE_IN {role: 1.5}]->(t)
      OR (u)-[:ACTIVE_IN {role: 2}]->(t) )
RETURN COUNT(r) as Errors
```

Cypher query C.5: Current relationships validation.

The FilterOutPrivate element is validated with Cypher query C.6. It checks for all algorithms if there are teams with privacy type 2 that are recommended to users who are not active in this team as a supporter. As

mentioned in chapter 3 are only the teams with privacy types 2, 4, and 5 validated on privacy. It returns the total number of incorrect recommendations as errors.

```
MATCH (u:User)-->(t:Team)
WHERE t.privacy_type=2
      AND ( (u)-[:RECO_RAND]->(t)
      OR (u)-[:RECO_DF]->(t)
      OR (u)-[:RECO_KBF]-(t)
      OR (u)-[:RECO_UBCF]-(t)
      OR (u)-[:RECO_IBCF]-(t) )
      AND NOT (u)-[:ACTIVE_IN {role:1}]->(t)
RETURN COUNT(u) as Errors
```

Cypher query C.6: Privacy validation for recommended teams which are private but the user is active in as a supporter.

The FilterOutNetwork element is validated with Cypher query C.7. All recommended teams with privacy type 4 or 5 are checked on two criteria. If the user is a member of the network the team belongs to or if the user is already active in the team as a supporter. The total number of incorrect recommendations are returned as errors.

```
MATCH (u:User)-->(t:Team)-->(n:Network)
WHERE ( t.privacy_type=4 OR t.privacy_type=5 )
      AND ( (u)-[:RECO_RAND]->(t)
      OR (u)-[:RECO_DF]->(t)
      OR (u)-[:RECO_KBF]-(t)
      OR (u)-[:RECO_UBCF]-(t)
      OR (u)-[:RECO_IBCF]-(t) )
      AND NOT ( (u)-->(n)
      OR (u)-[:ACTIVE_IN {role:1}]->(t) )
RETURN COUNT(u) AS Errors
```

Cypher query C.7: Privacy validation for recommended teams in networks.

The FilterOutEnded element is validated with Cypher query C.8. The *'end_date'*-attribute is compared with the current date. In this query example, this is 27 January 2016 (20160127), because this was the date of the data export from the marketplace.

```
MATCH (u:User)-[r]->(t:Team)
WHERE t.end_date < 20160127
      AND ((u)-[:RECO_RAND]->(t)
      OR (u)-[:RECO_DF]->(t)
      OR (u)-[:RECO_KBF]-(t)
      OR (u)-[:RECO_UBCF]-(t)
      OR (u)-[:RECO_IBCF]-(t))
RETURN COUNT(r) AS Errors
```

Cypher query C.8: Validation of no teams with a passed end date as recommendations.

The FilterOutDeactivated element is validated with Cypher query C.9.

```
MATCH (u:User)-[r]->(t:Team)
WHERE t.active=false
      AND ( (u)-[:RECO_RAND]->(t)
      OR (u)-[:RECO_DF]->(t)
      OR (u)-[:RECO_KBF]-(t)
      OR (u)-[:RECO_UBCF]-(t)
      OR (u)-[:RECO_IBCF]-(t) )
RETURN COUNT(r) AS Errors
```

Cypher query C.9: Validation of only active teams as recommendations.

The deployed FilterOutPrivate element is validated with Cypher query C.10, which is a slightly adjusted version of Cypher Query C.6. No longer teams are included in which the user is already active as a supporter.

```
MATCH (u:User)-[:RECOMMEND]->(t:Team)
WHERE t.privacy_type=2
RETURN COUNT(u) AS Errors
```

Cypher query C.10: New privacy check for recommended teams that are private.

The deployed FilterOutNetwork element is validated with Cypher queries C.11, C.12, and C.13.

```
MATCH (u:User)-[:RECOMMEND]->(t:Team)-[:PART_OF]->(n:Network)
WHERE (t.privacy_type=4 OR t.privacy_type=5)
      AND NOT (u)-[:MEMBER_OF]->(n)
RETURN COUNT(u) AS Errors
```

Cypher query C.11: New privacy check for recommended teams that are part of a private network.

```
MATCH (u:User)-[:RECOMMEND]->(t:Team)-[:PART_OF]->(n:Network)
WHERE t.privacy_type=6
      AND NOT (u)-[:MEMBER_OF {admin:true}]->(n)
RETURN COUNT(u) AS Errors
```

Cypher query C.12: Privacy check for recommended teams
that are exclusive for the co-creators in a private network.

```
MATCH (u:User)-[:RECOMMEND]->(t:Team)-[:PART_OF]->(n:Network)
WHERE t.privacy_type=7
      AND NOT ( (u)-[:MEMBER_OF {admin:true}]->(n)
      OR (u)-[:MEMBER_OF {colleague:true}]->(n) )
RETURN COUNT(u) AS Errors
```

Cypher query C.13: Privacy check for recommended teams that are exclusive
for the co-creators or colleagues in a private network.

The data type of the *'end_date'* was changed from an integer to string. As a result, was Cypher query C.8, to validate the FilterOutEnded, also slightly adjusted with the addition of str() encapsulation (Cypher query C.14).

```
MATCH (u:User)-[r]->(t:Team)
WHERE t.end_date < str(20160127)
      AND ((u)-[:RECO_RAND]->(t)
      OR (u)-[:RECO_DF]->(t)
      OR (u)-[:RECO_KBF]-(t)
      OR (u)-[:RECO_UBCF]-(t)
      OR (u)-[:RECO_IBCF]-(t))
RETURN COUNT(r) AS Errors
```

Cypher query C.14: Validation of no teams with a passed end date as recommendations.

Appendix D | Interview protocol

D.1 | Preliminary

A PDF-document with a maximum of five recommended teams is sent via email a couple of minutes in advance.

- Each recommended team is generated by one of the five algorithms:
    - Random (RAND) algorithm
    - Demographic filtering (DF) algorithm
    - Knowledge-based filtering (KBF) algorithm
    - User-based collaborative filtering (UBCF) algorithm
    - Item-based collaborative filtering (IBCF) algorithm
- Each team is presented with almost the same information as on the 'Discover'-page of Part-up. Only the picture is excluded for technical reasons. These seven elements are presented in a basic tabular format:
    - Title
    - Description
    - Names of the partners
    - Tags
    - Number of activities
    - Number of supporters
    - Number of days active

D.2 | User interview

All interviews are performed via a telephone, Skype or Hangouts conversation. During the short introduction and informal conversation, permission is asked for recording the interview.

The first question is not related to the provided recommendations:

1. On a scale from 0-7, are you able to find interesting part-ups?

Questions about the provided recommendations:

2. Which part-up is the most likely you will join as a partner?
3. On a scale from 0-7, will you join it?
4. How would you rank these recommendations yourself?
5. On a scale from 0-7, do these recommendations reflect your profile?
6. What is the most important element for each recommendation in making your decision?
7. My colleague Oana, a student of Erasmus University, will do further research on the relation of users to a team. Can she approach you later this month for an interview too?

8. Do you have other comments or suggestions?

The user is thanked for sharing his opinion. He is also informed that he will receive the key takeaways after all interviews are done. The conversation has ended.

D.3 | Processing

The numerical values and element options are transformed into a tabular format. The most important comments are transcribed for the appendix. These comments are aggregated on the subject and similarity for the results & evaluation chapter.