Date of acceptance Grade

Instructor

Applying soft computing for effort estimation in agile software projects - Literature review

Axel Baumgartner

September 12, 2022

Master

ÅBO AKADEMI UNIVERSITY
Department of Economics

Master's thesis

ÅBO AKADEMI UNIVERSITY

Faculty	Department
Faculty of Social Sciences, Business and Economics	Department of Economics

Author
Axel Baumgartner

Title

Applying soft computing for effort estimation in agile software projects - Literature review

Subject
Governance of Digitalization

Level	Month and year	Number of pages
Master	September 12, 2022	35 pages + 0 appendix pages

Abstract

Effort estimation is a challenging and essential part of any software development project. For this reason, the subject has been extensively researched and several techniques have been developed to facilitate the work. For agile software projects, expert judgement is the most popular method of conducting the effort estimation. The method is relying on the experience and knowledge of experts. Despite the popularity of the method, its results are seen as relatively poor. This work explores the possibility of improving the estimation process by utilizing computing power.

In agile software projects, value-producing changes can be made to the product throughout the project. Planning and documenting is conducted incrementally to avoid wasting resources in case of changes. The data and parameters required by algorithmic methods are often not available because few details are known about the final product. Thus, the utilization of computing power for effort estimation in agile projects is challenging. The lack of data is also the reason for the popularity of expert judgement, where the knowledge and experience of experts is the main resource for the estimation method. Soft computing aims to mimic the way the human brain deals with approximations and uncertainties, which enables it to function with partial and noisy data. This is what makes soft computing a potential approach for utilizing computing power in the agile effort estimation process.

The first objective of this thesis is to find out what is the state-of-art in utilizing soft computing techniques for effort estimation in agile projects. The research is conducted as a literature review, and the results are compared to older similar studies. The other objective is to focus on how soft computing has been utilized for planning poker in particular. Planning poker is chosen, because it is the most popular method for using expert judgement in agile teams.

The results imply that the number of publications on the subject has increased since the last related literature review in 2016. The publications were divided into the following categories according to topics: literature reviews, optimization of existing estimates, machine learning models, tools for team use, and the use of non-agile methods by utilizing soft computing. The selected articles did not include any studies focusing explicitly on expert systems or genetic algorithms. Three publications were found that were seen to present a solution that had been used or was intended to be used in planning poker by the development team.

Keywords
Agile, Cost Estimation, Effort Estimation, Soft Computing, Machine Learning

Where deposited

Additional information

Contents

1 Introduction

Effort estimation is a crucial part of planning and executing a software development project. The estimations guide the team and the stakeholders when prioritizing work and resources, while balancing between the four project boundaries: scope, quality, schedule and cost. Several different methods can be used for estimating effort, and the choice is usually depending on the methodology used in the project. This work focuses on agile methodologies, where the estimation process is iterative.

Unlike the traditional waterfall method, agile projects are expected to start with only a high-level understanding of the resulting product. Requirements are then included and modified during the development process in iterations. This creates flexibility for the team and the stakeholders when new needs emerge during the development process. Agile methods aim to avoid planning too far in the future to avoid wasting the work done if changes are made to the project plan later.

The minimal planning and an iterative way of working creates challenging conditions for estimating effort. Data-driven estimation methods are hard to implement due to the little information in use. This leaves the team with expert judgment, which means that the estimations are made based on the knowledge and experience of the participants. Planning poker is used to reduce excessive optimism and to improve the understanding through discussion. Although expert judgement is seen as a more efficient solution compared to formal methods in agile projects [Jørgensen et al., 2009], the estimation process is experienced challenging. Using the whole team for the estimation process in every iteration is also expensive and time consuming [Usman et al., 2014].

This work aims to find out whether soft computing could enable the utilization of computing power in the agile estimation process. The goal is to achieve even better results than estimation methods based purely on expert judgement. The potential of soft computing lies in its nature to handle uncertainties and approximations. The human brain is seen as a role model for technologies, and despite the limited data in agile projects, experts are able to make meaningful estimates.

Based on existing literature, it is desired to find out what soft computing techniques have been used for effort estimation in the context of agile software projects. We also want to understand how and for what purpose it has been utilized. Finally, we'll investigate how the techniques have been used to enhance planning poker, which is the most popular expert judgment based estimation method for agile teams.

1.1 Structure of the thesis

The first chapter, the introduction, presents the subject, research questions, expectations, and the structure of the thesis.

The second chapter provides background information about the topic, which is required for understanding the following chapters. The chapter begins with a brief

introduction to effort estimation in software projects in general. Then the agile methodology is presented and compared to the traditional counterpart. SCRUM is used as an example of an agile framework to provides a more practical view into the effort estimation process in agile projects. Soft computing is presented as a concept and the independent soft computing techniques are briefly introduced.

The third chapter introduces related work conducted in the scope of intelligent techniques utilized for effort estimation in agile projects.

The fourth chapter presents the research method used in the thesis. The resources are presented along with the search string and inclusion and exclusion criterion. The selected articles are listed in the table including titles and references.

The fifth chapter presents the results of the literature review. The chapter is divided into sections following the research questions.

The sixth chapter is the discussion. The results and the answers to the research questions are discussed. The chapter is divided in two main sections: research direction and using soft computing for planning poker. Suggested future work and threats to validity are presented in the end of the sixth chapter.

The conclusion is the seventh and final chapter. The work is summed up, and the main findings are presented.

1.2 Research questions

RQ1: *Which soft computing techniques has been discussed in literature?*

With **RQ1**, it is desired to find out what soft computing techniques have been covered in the literature in the context of agile software project effort estimation in general.

RQ2: *In which direction has the research developed since 2016?*

In the literature review conducted by Bilgayan et al. (2016), it is mentioned that the agile methodologies are gaining popularity, while the use of soft computing in this context has received very little attention. **RQ2** is used to map how much research has been done since 2016. It is also desirable to gain an understanding of the direction in which the research is developing.

RQ3: *How have soft computing techniques been used in relation to effort estimation in agile software development?*

The aim of **RQ3** is to understand how the techniques mentioned in the literature have been utilized in the context of agile software development. The goal is to group and divide the findings of **RQ1**, in order to present what the goals of utilizing soft computing techniques have been in the literature.

RQ4: *Can soft computing techniques be used to support planning poker?*

The background section refers to a study [Usman et al., 2014] in which planning poker is highlighted as one of the most popular estimation techniques in agile software development. Only expert judgment is ranked before planning poker, which itself is in the core of planning poker. Planning poker is also seen to include all of the agile characteristics [Sudarmaningtyas and Mohamed, 2021]. Because of these factors, we want to focus on planning poker in this work. **RQ4** aims to uncover potential frameworks or tools presented in the literature that could be used to support planning poker.

2 Background

2.1 Effort estimation in software development

One formal definition of the verb "estimate" can be found in Merriam-Webster's dictionary [dic, 2016]: "to judge tentatively or approximately the value, worth, or significance of something". The noun "estimate" is defined as "a statement of the cost of work to be done". In the activity of estimating software development effort, the aim is to make as accurate predictions as possible about the duration and price of development and maintenance work. The estimations are sometimes made based on a heavily noisy and incomplete input.

Traditionally the effort estimation has been a matter of project management. The project manager needs accurate predictions for creating reliable development plans, financial plans and preparations for bidding rounds. Successful estimates are seen as essential for the success of a software project. How often the estimates are used and updated after the initialization of the project largely depends on the type of project. Due to several different methodologies and project types, several different estimation methods have emerged in the software industry. A lot of research has also been produced on the subject. Only by querying IEEE Xplore [IEE, 2022] with the search string "software and effort estimation", 1,730 results are produced. Despite the extensive research work and several practices, estimation of effort is experienced very challenging and often unsuccessful [Jalil and Shahid, 2008].

The estimation methods have been categorized in previous studies in several different ways ([Boehm, 1984],[Trendowicz and Jeffery, 2014]). Sudarmaningtyas and Mohammed (2021) used three aspects to map the methods: principle, strategy and the required data. The identified estimation methods can belong to each category gradually, forming hybrid methods belonging to several categories. The hybrid methods are usually combinations of two or more traditional estimation methods. The following subsections are following Sudarmaningtyas and Mohammed's (2021) division of categories.

2.1.1 Strategy

The strategy aspect consists of the extremes bottom-up and top-down. In a bottom-up strategy the team estimates smaller parts of the system, which are together combined to an overall project size estimation. The opposite method is the top-down method where the complete project size is known before the estimation, which is then broken down into smaller tasks by the estimating team. The initial project size might be formulated based on an existing budget or schedule constraint.

2.1.2 Principle

The principle is divided into algorithmic and non-algorithmic models. Algorithmic estimation models are based on mathematical formulas that use project properties as parameters, such as project size and team attributes. Popular models that are considered algorithmic models are COCOMO (Constructive Cost Model) [Pandey, 2013] and Functional point base [Sudarmaningtyas and Mohamed, 2021].

2.1.3 Data

The third aspect is the need for historical data in the estimation process. Some methods are heavily dependent on actual data or knowledge, such as machine learning models and expert judgement. In the division by Sudarmaningtyas and Mohammed (2021), algorithmic models are not dependent on data in general.

2.2 Effort estimation in agile projects

2.2.1 Agile principles and values

The agile methodologies gained popularity in the software development field during the early 2000's. It is still a popular methodology in modern software development and the principles and values have also been adopted outside of the software development field. The original manifesto was published by Kent Beck et al. (2021). The authors wanted to develop the development process to support their values, which prioritized individuals and interactions, working software, customer collaboration and responding to change. The following agile principles [Beck et al., 2001] are important to understand, as they lay boundaries for the effort estimation process:

1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3. *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
4. *Business people and developers must work together daily throughout the project.*
5. *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7. *Working software is the primary measure of progress.*

8. *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*

9. *Continuous attention to technical excellence and good design enhances agility.*

10. *Simplicity–the art of maximizing the amount of work not done–is essential.*

11. *The best architectures, requirements, and designs emerge from self-organizing teams.*

12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

2.2.2 Project management in agile projects

A fundamental difference between traditional and agile software projects lies in the project manager's role. In fact, the need of a project manager in its traditional meaning in agile projects is debated. Agile teams are expected to be self-organized (principle 11) and self-managed, and every team member is seen as responsible for the success of the project. Even though the role is not recognized in the agile methodologies, the need for it usually exists [Gandomani et al., 2020]. The role might be fulfilled by several other roles, such as the scrum master and product owner in the Scrum framework.

In order to maintain agility in a project, it is not desirable to do unnecessary planning and preparation too far into the future. This makes it possible to make changes without losing any work done. However, it is not always possible to avoid making long-term schedules and cost estimations. This can happen, for example, when a project plan is created to make a bid, or when a customer requests estimates to draw up their own budget for further development of an ongoing project. Stakeholders may not always have an agile way of working, which causes friction at the interfaces of operations.

When creating a long-term plan for an agile project that includes cost and schedule estimates, agile specific challenges are encountered. The main challenge can be seen to be that the estimates must be made on the basis of indicative and potentially changing requirements. The customer and other stakeholders are actively participating in the development process (agile principle 3) and allowed to request for changes to both planned and developed features (agile principle 2). This ongoing collaboration ensures that the product generates as much value as possible for stakeholders, but also means that a precise fixed estimation of effort in agile projects is unlikely to be achieved.

One way of managing these uncertainties is to leave enough leeway in the estimates. In this way, it is also possible to transfer the increase in the effort of one feature to the reservation of an undersized feature. In order to avoid exact values, t-shirt sizes (or equivalent) can be used as units when estimating features. The team can more

easily agree between a few size categories (XS, S, M, L, XL) on how the effort of the feature is estimated [Ravi and S., 2021].

2.2.3 Effort estimation in Scrum

To ensure that agile values are followed in a software project, ready-made frameworks can be implemented in the work. Scrum is the most popular of these frameworks in the software development field [Collabnetversionone, 2018]. Understanding Scrum also concretizes well what challenges are involved in the effort estimation processes of agile projects.

The work in Scrum is divided into time-boxed iterations where the development team is expected to deliver a predefined amount of work. The iterations are commonly called sprints and their lengths are varying from a week up to one month. During the sprint, several predefined meetings are held. Some events focus primarily on information sharing, but most of them address the product backlog.

The backlog is a collection of the features of the product to be developed. The most important events in terms of effort estimation are the backlog refinement and sprint planning. In the backlog refinement meeting, the team divides the features into smaller tasks that can be fit to the length of a sprint. These tasks are then used in the sprint planning meetings to create a set of tasks for the upcoming sprint.

Because the content of the sprint can not be modified, it has to be carefully planned. If too much work is planned for the sprint, the team will not have time to complete everything. The planning has also failed if everything has been done too early. For this reason, the effort of the tasks selected for the sprint must be carefully estimated. Scrum itself is not defining how the estimation process should be conducted [Schwaber and Sutherland, 2020].

Ideally, in Scrum each sprint can be considered an independent small project with its own schedule and price. The velocity is a key metric in Scrum, and it represents how much work the development team can do during one sprint. If the budget is tight and there is not enough velocity to complete the whole project, the most important features have to be prioritized. Due to the prioritization during the process, it is not advisable to evaluate and define tasks too far into the future.

2.2.4 Popular estimation techniques in agile projects

According to the systematic literature review on effort estimation techniques in agile software development by Usman et al. (2014), the most used estimation methods are based on subjective assessment. The most popular techniques are expert judgment, planning poker and the use case points method. The overall prediction accuracy for these techniques was considered low. Expert judgment is based on the practitioner's expertise, trusting in the human mind as the measurement instrument [Matsubara et al., 2022], making it judgemental rather than mechanical [Halkjelsvik and Jørgensen, 2012].

The subjectivity of expert judgment can be seen as a weakness. Human characteristics do not always promote truthful estimates, and the estimates might get affected by group dynamics. It has been argued that humans tend to be over-optimistic, at least when they are estimating their own work [Jørgensen, 2004]. Being aware of expectations can also mislead the estimation. This is a problem, for example, in a situation where the estimators are aware of the client's budget and time constraints [Jørgensen, 2014]. The presence of dominant persons, such as supervisors, project managers or more senior developers, might also influence the estimators [Aranda and Easterbrook, 2005]. No significant ways for correcting a resulting bias have been identified. In order to prevent it, the focus must be held on developing the estimation conditions [Jørgensen, 2014].

2.2.5 Planning poker

Planning poker is a popular technique for using expert judgment together in a team. The estimation process starts by selecting a task to be estimated. The task description is read aloud and team members can discuss and ask questions about the task. When everyone has a good enough understanding, the participants estimate the task individually. The individual estimates are revealed at the same time and compared with each other. If everyone suggests the same task size, it will be chosen as the final estimate. If not, the task is openly discussed and the previous steps are repeated until everyone has come to an agreement on the size. When the final size is chosen, the process can start over with a new task [Sudarmaningtyas and Mohamed, 2020].

Planning poker can be played using physical cards with a size class written on them. The scale can be freely chosen, and it does not have to represent any real-world units (e.g. hours/days). As an alternative, a scale representing relative size can be used, such as a modified Fibonacci sequence (0, $\frac{1}{2}$, 1, 2, 3, 5, 8, 13, 20, 40, 100) [ppC, 2022].

Planning poker helps to make the voices of all the team members heard, regardless of their experience and influence in the group. Estimating tasks in groups also ensures that everybody has a better overall understanding of the tasks. Expert judgment based effort estimation has been suggested to provide better results when conducted in teams rather than individually [Haugen, 2006].

2.3 Soft computing techniques

Soft computing has been an area of research in the field of computer science since the early 1990's [Rao and G., 2011]. It has its roots in the publication "Fuzzy sets" from 1965 by Zadeh (1965), where performing computer processes with natural values was introduced. Today soft computing has been applied in several different industries and domains. The role model for soft computing is seen to be the human mind [Ibrahim, 2016], which makes it a potential solution for the subjective effort

estimation process in agile software development.

The limitations of traditional hard computing are encountered when problems deal with approximations, partial truth and uncertainty. These are all features of complex real-life problems, such as effort estimation in agile software development. Soft computing works similarly to the human mind, which deals in environments of uncertainty, but still manages to make rational decisions. Soft computing is a umbrella term for many independent techniques, such as fuzzy logic, neural networks, genetic algorithms, machine learning and expert systems [Ibrahim, 2016]. The techniques can be used successfully complementing each other, of which "neurofuzzy systems" is a good example [Rao and G., 2011].

In the following subsections, a brief overview of the techniques suggested by Ibrahim (2016) is presented.

2.3.1 Fuzzy logic

In Fuzzy logic, the truth is seen as a degree of truth, rather than a boolean value. In other words, the truth is a value between completely true and completely false [Zadeh, 1988]. As an example, in natural language, people rarely talk about the outside air temperature as "cold" or "hot". Instead, spaces between these extremes are described as "quite cold" or "medium warm". Formulating these linguistic values to be understood by computers, enables us to create software that mimics human problem solving [Chen and Poo, 2003].

Fuzzy logic has been used in image processing and identification systems, but in most cases practical implementations are found in the form of fuzzy control systems. These are used in cameras, air conditioners, refrigerators, automatic gearboxes, washing machines and other everyday objects [Broesch, 2009].

The process in a fuzzy logic system consists of tree components: fuzzification, fuzzy logic rules and defuzzification [Ibrahim, 2016]. The numerical input variables are transformed into linguistic values in the fuzzification process. For example, the input could be crisp sensor data. This is done by assigning the value to one or more predefined fuzzy sets representing linguistic values, using a membership function. The linguistic values are needed for further processing with the fuzzy rules. The linguistic description can then be turned back into a crisp value using different methods. On popular defuzzification method is the center-of-area method, which computes the centroid of the composite area representing a output fuzzy term [Zhang, 2010] .

2.3.2 Machine learning

Machine learning is part of the general field of artificial intelligence. The term machine learning was formed already in the 1950's, when Arthur Samuel from IBM developed a computer program that utilized a strategy for choosing the next move in the game of checkers. The program used the information of past moves and their success to make the decisions [Samuel, 1959]. The key factor in machine

learning is to utilize existing data to perform a task more efficiently. In practice, this means building models that can be trained using training data, which can then independently make decisions without explicit instructions [Wang et al., 2009].

Machine learning can be divided into supervised and unsupervised learning. The concepts deal with how the data is interpreted in the learning process. Supervised learning requires prelabeled data, which the machine can use for finding patterns and relationships in the data. The downside of supervised learning is that it requires structured data, which can be laborious to produce. Alternatively, unsupervised learning does not set requirements for data quality. The relationships and patterns are formed on the basis of raw data, not having a measured outcome [Jiang et al., 2020].

The problems that are solved with machine learning can be divided into tree types: classification, regression and clustering. Depending on the data in use, different strategies has been used when predicting the effort of software projects. The following subsections introduce the basics of the problem types.

Classification

Classification is one of the problems that are sought to be solved using machine learning. The goal is to divide the data into the most appropriate categories or classes. The computer is taught to group data based on its characteristics. The training data is labeled to teach the model what are the characteristics of each class. This means that the technique is supervised [Kotsiantis, 2007].

Popular classification machine learning techniques:

- Logistic Regression
- Naive Bayes
- K-Nearest Neighbors
- Decision Tree
- Support Vector Machines

Regression

Regression is another supervised machine learning technique. Compared to classification, regression predicts continuous outcomes. Use cases of regression can be found in the analytics of finance for predicting market fluctuations or trading outcomes, where numerical values are preferred over predefined classes. The model is trained to understand the relationship between the expected output and input based on labeled training data.

Popular supervised learning regression algorithms [Jav, 2022]:

- Simple Linear Regression

- Multiple Linear Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression

Clustering

In the process of clustering, raw data (unlabelled) is assigned into subsets based on their similarities and their differences. The amount of resulting sets might be regulated with parameters. An example of a well-known clustering algorithm is the k-means algorithm [Kononenko and M, 2007].

2.3.3 Artificial neural networks

Neural networks is a subset of machine learning. It consists of interconnected elements signaling to one another, mimicking biological neurons in the human brain. Neural networks offer powerful solutions for several specific applications, such as patterns recognition, image processing and stock market predictions. Depending on the problem, the networks might need large amounts of training data. One advantage of the method is its ability to deal with incomplete data sets. The training can be completed either supervised or unsupervised [Ibrahim, 2016].

2.3.4 Genetic algorithms

Genetic algorithm is an optimization algorithm developed by John Holland et al. in the 1960's [Yang, 2020]. It is nature-inspired and follows the biological evolution theory by Charles Darwin. The algorithm has been developed into several versions and used in many different domains, such as pattern matching [Auwatanamongkol, 2000], discrete systems and time table optimization [Assi et al., 2018].

The process of finding the best solution for a problem begins with a randomly chosen set of alternatives. The candidates are evaluated and only the fittest will continue to the next phase. The chosen candidates are then crossed to create new alternatives, and the process is repeated until the requirements are satisfied. Each iteration is called a generation. The algorithms are well suited for generating optimization and search algorithms [Ludwig, 2004].

2.3.5 Expert systems

The Britannica dictionary [EPB, 2016] defines expert systems as "a computer program that uses artificial-intelligence methods to solve problems within a specialized

domain that ordinarily requires human expertise". Expert systems has its roots in the 1960's, when a prototype of expert systems was created by Edward Feigenbaum and Joshua Lederberg. It was called Dendral and created for analyzing chemical compounds.

In general, expert systems are using artificial intelligence to imitate the judgment and decision making of human experts. The systems are designed to provide expertise for beginners and assistance for experts. The domain of knowledge is not specified in any way. The knowledge from experts is transferred to computers and stored in a knowledge base. The method of organizing the knowledge is important, to ensure that the interface engine can efficiently retrieve information. The organizing can be done using several methods, such as decision trees, rule bases (IF - THEN), scripts and semantic networks. The interface engine contains the logic of controlling the knowledge acquisition from the knowledge base, based on the input from the user interface [Aronson, 2003] [Ibrahim, 2016].

Expert systems as a concept is considered to have peaked in popularity in the 1980's, but today it is even considered as a somewhat outdated technology. In its prime, business organizations invested in extracting the knowledge of experts to databases. Today, the term "expert systems" is rarely used, and the constructed databases have been used in descendants of expert systems such as data mining, business intelligence, and CRM/ERP systems. The problematic of expert systems lies in its fundamentals of transferring knowledge from humans to computers, rather than considering the truth to be found in data. This way of thinking is the opposite of modern data analysis [Nettleton, 2014].

3 Related work

Perkusich et al. (2020) published a review on applying intelligent techniques on software engineering in 2020. The study was based on 92 unique studies. The amount of applications of intelligent techniques for agile software development was identified to be increasing. The most used techniques were machine learning, reasoning under uncertainty and search-based solutions. Effort estimation was stated as one of the predominant purposes of applying intelligent techniques to agile software engineering.

Different estimation methods used in agile software development was studied by Sudarmaningtyas and Mohammed (2021) in the form of a literature review. The estimation methods found were divided in the following categories: expert judgment, algorithmic, machine learning and static. Planning poker, a method implementing expert judgment, was seen to best implement the agile characteristics. Purely machine learning based estimation techniques had very few similarities with the agile characteristics.

Arora et al. (2020) conducted a systematic literature review on machine learning estimation approaches in Scrum projects. The article is published in 2020. The results present which machine learning models have been used and how they performed. 17 different machine learning models were identified in the literature. The publication dates are between 2014 and 2018. Arora et al. (2020) concludes that there are many machine learning approaches that have not been discussed in the literature in this context. No generic models for effort estimation in Scrum were identified.

In 2021, Arora et al. (2021) compared the state-of-the-art regressor models used in effort estimation of agile software projects. The results of applying six regressors on preprocessed data sets indicates that the CatBoost regressor is outperforming other regressors. The CatBoost regressor had a prediction accuracy of 0.95.

The most similar study on the utilization of soft computing for cost and effort estimation has been conducted by Bilgayan et al. (2016). The work concentrates on agile software development projects and is conducted as a methodical literature review. The results are based on 8 articles and present soft computing techniques that have been used in cost and effort estimation. The accuracy and the error rates are compared between the techniques and a technique based on Bayesian networks is stated as the most accurate. The authors conclude that agile software development projects are increasing in percentage, but the amount of research done regarding agile software development and soft computing is still small.

The data collection method for Bilgayan et al.'s (2016) study is described only briefly without details. The search strings used are inexact and produce numerous results. The study will be used as a reference for this thesis, but the literature review will have a more systematic approach when investigating the state of art in the utilization of soft computing for effort estimation. It is also desired to present an updated version of the results in 2022.

4 Method

A systematic literature review is a research method where the goal is to answer defined research questions with all the existing and available scientific literature. The resulting review's type is called a *secondary study*, while the works included in the study are considered *primary studies*. A systematic literature review can be used as a part of another type of research, providing background for the study. Other reasons for using the method can be the need for summarizing empirical evidence for a topic or phenomena of interest, or for identifying research gaps in the current scientific literature [Kitchenham and Charters, 2007].

In this work the method is used for answering the research questions presented in section 3.1. In the latter sections the following features of the research are documented: search string, sources of literature, inclusion and exclusion criteria, and the steps taken for the selection of articles.

4.1 Search string

The search string consists of three keywords that must be met and which are separated with the "AND" operator: *software*, *effort estimation* and *agile*. We only want to use software-related publications and want them to deal with agile methods. The second part of the search string is enclosed in parentheses, and the words are separated by the "OR" operator, meaning that at least one of the keywords must be met. The keywords chosen includes *soft computing* and the soft computing techniques presented in the article by Ibrahim (2016).

(
"fuzzy logic"
OR "genetic algorithms"
OR "artificial neural networks"
OR "machine learning"
OR "expert systems"
OR "soft computing"
)
AND "software"
AND "effort estimation"
AND "agile"

4.2 Databases

The databases chosen as resources are the following: *SienceDirect*, *Scopus*, *ACM Digital library* and *IEEE/IEE Electroic Library*. (Table 1.) The queries were made using the search string defined in section 3.1. Apart from the default database settings, no other filtering rules were set. There were no restrictions on the time

period for search results.

Table 1: List of resources

Source Name	**Description**
SienceDirect	A database with 1.4 million open access articles. Includes publications from the domains of physical sciences and engineering [Sci, 2022] .
Scopus	Scopus is an abstract and citation database of peer-reviewed literature including scientific journals, books and conference proceedings. The database contains publications from fields of science, technology, medicine, social sciences, and arts and humanities [Sco, 2022].
ACM Digital Library	ACM is the world's largest educational and scientific computing society. The digital library contains ACM publications, including journals, newsletters, books, conference proceedings and technical magazines [ACM, 2022].
IEEE/IEE Electronic Library	IEEE is the world's largest technical professional organization with 400 000 members around the world [IEE, 2021]. IEEE Xplore contains over 5million publications in the technical field [IEE, 2022].

4.3 Inclusion and exclusion criteria

The selection criteria used for the inclusion and exclusion of publications are presented in tables 2. and 3. The same criteria was used in each phase. Figure 1. presents the amount of publications processed after each step. Table 4. includes all the titles selected. To avoid biased selection of articles, the snowballing of publications has been excluded. In order to use snowballing without the threat of selecting only papers that support the author's desired perspective, more time should be reserved for the research. The selection process was conducted in the following phases:

1. The initial search was conducted using the search string and the databases defined in sections 3.3 and 3.2.

2. After the initial search, the relevance of the resulting papers were decided using the abstract and the title.

3. The remaining relevant papers were reviewed in their entirety.

Table 2: Inclusion criteria

Criteria
The publication discusses the application of soft computing for effort estimation in an agile context.

Table 3: Exclusion criteria

Criteria
The publication is not written in English.
The publication is not peer-reviewed.
The publication appears repeatedly (only the latest version is considered).

Figure 1: Number of studies after each phase.

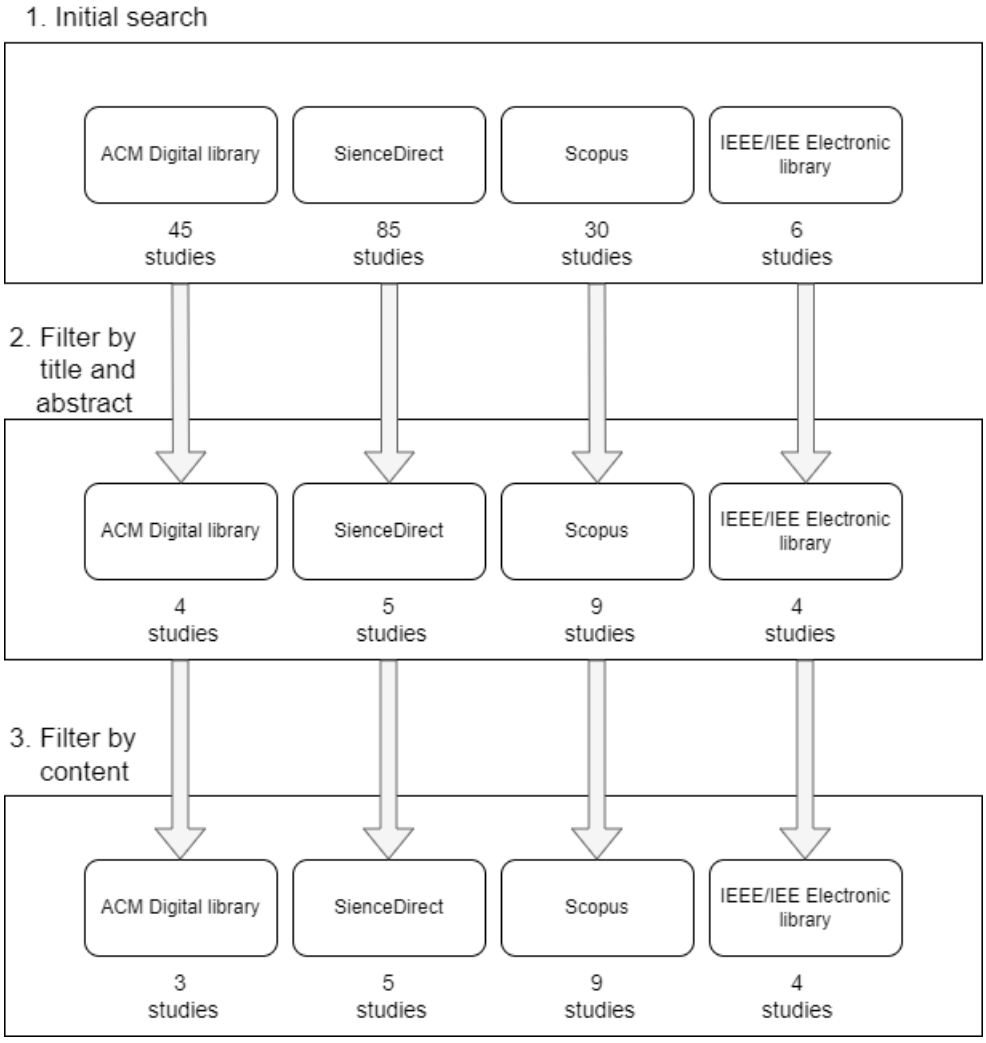


Table 4: Selected articles for the literature review

Title	**Ref.**
Effort Estimation of Agile Development using Fuzzy Logic	[Saini et al., 2018]
Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points	[Panda et al., 2015]
An Enhanced Framework for Effort Estimation of Agile Projects	[Raslan and Nagy, 2018]
A hybrid methodology for effort estimation in agile development	[Tanveer et al., 2018]
Effort Estimation using Bayesian Networks for Agile Development	[Ratke et al., 2019]
An Effort Estimation Support Tool for Agile Software Development: An Empirical Evaluation	[Dantas et al., 2019]
Using Developers' Features to Estimate Story Points	[Scott and Pfahl, 2018]
Approximation of COSMIC functional size to support early effort estimation in Agile	[Hussain et al., 2013]
Cost-Effective Supervised Learning Models for Software Effort Estimation in Agile Environments	[Moharreri et al., 2016]
A predictive model to estimate effort in a sprint using machine learning techniques	[Ramessur and Nagowah, 2021]
Intelligent software engineering in the context of agile software development: A systematic literature review	[Perkusich, 2020]
Deep learning model for end-to-end approximation of COSMIC functional size based on use-case names	[Ochodek et al., 2020]
A State of the Art Regressor Model's comparison for Effort Estimation of Agile software	[Arora, 2021]
A Systematic Literature Review of Machine Learning Estimation Approaches in Scrum Projects	[Arora et al., 2020]
An efficient effort and cost estimation framework for Scrum Based Projects	[Arora and Verma, 2018]
A Review Article on Software Effort Estimation in Agile Methodology	[Sudarmaningtyas and Mohamed, 2021]
A Review of Software Cost Estimation in Agile Software Development Using Soft Computing Techniques	[Bilgaiyan et al., 2016]
Effort estimation in agile software development using experimental validation of neural network models	[Bilgaiyan et al., 2019]
Empirical assessment of machine learning models for agile software development effort estimation using story points	[Satapathy and Rath, 2017]
Estimating Story Points from Issue Reports	[Porru, 2016]
Bayesian network model for task effort estimation in agile software development	[Dragicevic et al., 2017]

5 Results

5.1 Techniques covered in literature

Table 5. includes all the soft computing and machine learning techniques discussed in the literature. By forming an overview on the techniques that have been used in the context of agile software effort estimation, RQ1 is answered. Publications that are based on literature reviews are excluded. Fuzzy logic is mentioned in only four publications and genetic algorithms and expert systems are not discussed at all.

Table 5: Techniques covered in literature

Technique	**Ref.**	**Tot.**
Decision Trees	[Dantas et al., 2019] [Hussain et al., 2013] [Moharreri et al., 2016] [Ramessur and Nagowah, 2021] [Arora, 2021] [Satapathy and Rath, 2017] [Porru, 2016]	7
Fuzzy logic	[Raslan and Nagy, 2018] [Arora and Verma, 2018] [Saini et al., 2018] [Hussain et al., 2013]	4
Random Forest	[Moharreri et al., 2016] [Arora, 2021] [Satapathy and Rath, 2017]	3
Support Vector Machines	[Scott and Pfahl, 2018] [Porru, 2016] [Ramessur and Nagowah, 2021]	3
Bayesian network	[Ratke et al., 2019] [Dragicevic et al., 2017]	2
Linear Regression	[Ramessur and Nagowah, 2021] [Arora, 2021]	2
k-Nearest Neighbours	[Ramessur and Nagowah, 2021] [Porru, 2016]	2
Naıve Bayes	[Moharreri et al., 2016] [Porru, 2016]	2
Convolutional Neural Network	[Ochodek et al., 2020]	1
Cascade correlation Neural Network	[Bilgaiyan et al., 2019]	1
Elman Neural Network	[Bilgaiyan et al., 2019]	1
Feedforward back-propagation Neural Network	[Bilgaiyan et al., 2019]	1
Probabilistic Neural Networks	[Panda et al., 2015]	1
GMDH Polynomial Neural Network	[Panda et al., 2015]	1
Cascade Correlation Neural Network	[Panda et al., 2015]	1
MLP (Artificial Neural Networks)	[Ramessur and Nagowah, 2021]	1
General Regression Neural Networks	[Panda et al., 2015]	1
Gradiant boosted trees	[Tanveer et al., 2018]	1
CatBoost Regressor	[Arora, 2021]	1
XGB Regressor	[Arora, 2021]	1
AdaBoost Regressor	[Arora, 2021]	1
Stochastic Gradient Boosting	[Satapathy and Rath, 2017]	1
Logistic Model Tree	[Moharreri et al., 2016]	1

5.2 Direction of research

Table 6. displays the number of publications published each year, forming an answer to RQ2. The earliest publication is from 2013 and the newest is from 2021. The total number of publications considered is 21. Using the search string presented in section 3.2, only two articles are found before Bilgayan et al.'s (2016) corresponding work in 2016. The only shared article is Panda et al.'s (2015) article on using neural networks for optimizing estimations. The median of the publication year is 2018.5, indicating that the amount of research in the subject has increased.

Eleven new publications are found after Arora et al.'s (2020) study on applying machine learning with effort estimation in Scrum. Arora has contributed with two publications after 2018. Excluding Arora's own publications, six of the eleven new articles are discussing machine learning techniques.

Table 6: Number of publications per year

YOP	**Publications by**	**Total**
2013	[Hussain et al., 2013]	1
2014		0
2015	[Panda et al., 2015]	1
2016	[Porru, 2016] [Bilgaiyan et al., 2016] [Moharreri et al., 2016]	3
2017	[Satapathy and Rath, 2017] [Dragicevic et al., 2017]	2
2018	[Saini et al., 2018] [Scott and Pfahl, 2018] [Arora and Verma, 2018] [Raslan and Nagy, 2018] [Tanveer et al., 2018]	5
2019	[Ratke et al., 2019] [Dantas et al., 2019] [Bilgaiyan et al., 2019]	3
2020	[Arora et al., 2020] [Perkusich, 2020] [Ochodek et al., 2020]	3
2021	[Arora and Verma, 2018] [Ramessur and Nagowah, 2021] [Sudarmaningtyas and Mohamed, 2021]	3

5.3 What has soft computing been used for?

Table 7. is used to answer the RQ3. The categories in the table are intended to provide the reader with an overview on how the soft computing techniques have

been used in the context of agile effort estimation. For simplicity, each article can only be found in one category, even though some of the articles might fit in several categories. Figure 2. visualizes how the categories are related to each other. The following subsections describe the articles under the categories in more detail.

Table 7: Categories representing what soft computing has been used for in the literature

Category	**Publications by**	**Total**
Literature review	[Arora et al., 2020] [Bilgaiyan et al., 2016] [Perkusich, 2020] [Sudarmaningtyas and Mohamed, 2021]	4
Soft computing with a non-agile method or model	[Hussain et al., 2013] [Ochodek et al., 2020] [Raslan and Nagy, 2018]	3
Improving existing estimates	[Panda et al., 2015] [Satapathy and Rath, 2017]	2
Support for the development team	[Porru, 2016] [Dantas et al., 2019] [Tanveer et al., 2018]	3
Machine learning models for effort estimation	[Ratke et al., 2019] [Scott and Pfahl, 2018] [Moharreri et al., 2016] [Ramessur and Nagowah, 2021] [Bilgaiyan et al., 2019] [Dragicevic et al., 2017]	6

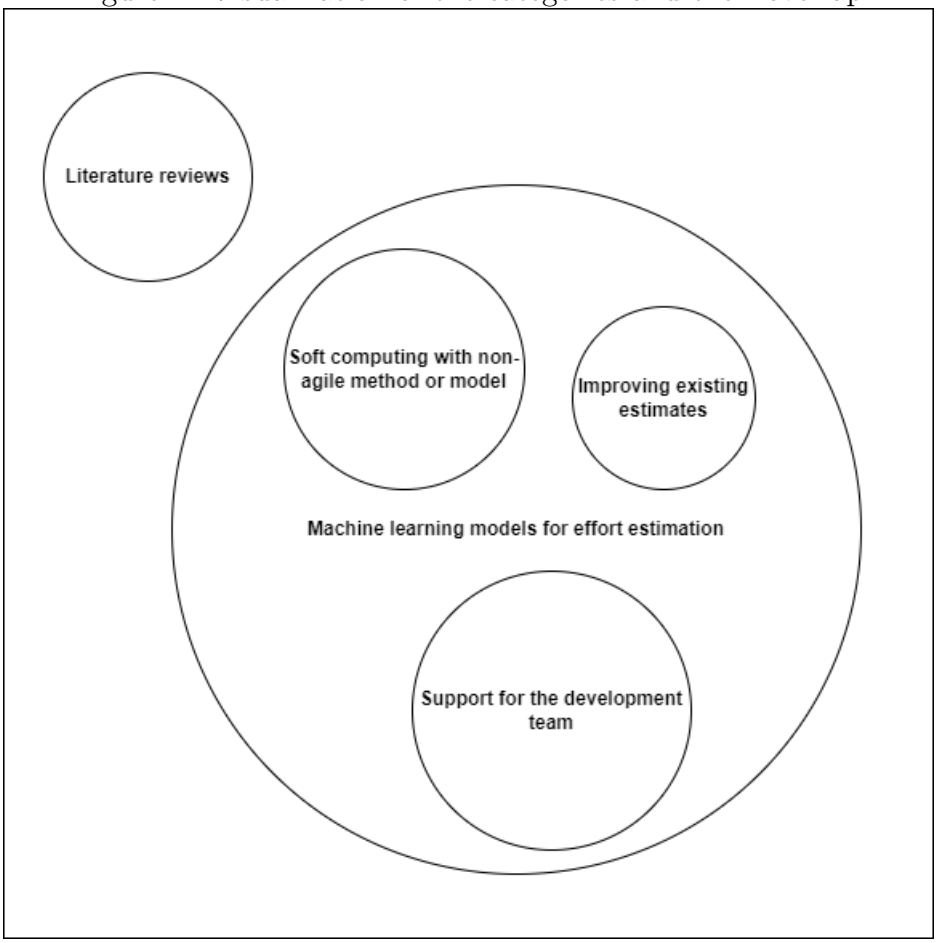
5.3.1 Using a non-agile model or method for agile projects

Soft computing techniques have been used in combination with methods and models that have not traditionally been intended for agile projects, such as the COSMIC functional size measurement method and the constructive cost model (COCOMO). In the review articles, the methods and models are suggested to be used partly or fully in agile projects with the use of soft computing techniques.

COSMIC method

The COSMIC method is an algorithmic estimation method, which can be problematic to use in an agile context. In the review by Sudarmaningtyas and Mohamed (2021), the COSMIC method is fulfilling only two of the five characteristics of agile methodologies. The method requires a complete and detailed list of functional user requirements in the early stages of the project, which is directly in conflict with

Figure 2: Visualization of the categories and their overlap



the agile way of gathering and specifying requirements as the project progresses. Hussain et al.[Hussain et al., 2013] and Ochodek et al.[Ochodek et al., 2020] have identified this contradiction, but managed to build classification models designed for agile projects.

To meet the agile expectations, Hussain et al.'s (2013) model is designed to cope with informally written functional processes (i.e use cases). The model is using supervised text mining for measuring the COSMIC functional size from the use cases. The functional sizes are then used for creating a historical database for the training of the system. The manual work is slow, which means that a faster practical implementation would require a very consistent way of writing requirements. The resulting system can then automatically classify the size of new textual requirements belonging to a functional process.

Ochodek et al. (2020) has used a similar approach for the estimation process using the textual properties of use cases for measuring the functional size. To make the process faster and to require less detail for the estimation, Ochodek et al.'s model is using only names of the use cases. The model was evaluated using a dataset of 437 use cases and the best results were achieved using a convolutional neural network.

COCOMO II

COCOMO II is a software cost estimation model that is based on 163 projects implementing the waterfall method. The model considers product-, hardware-, personnel- and project attributes for measuring the effort. COCOMO II shares the same challenges of following the principles of agile methodologies with the COSMIC method. To produce accurate results, the model needs detailed requirements.

Raslan and Nagy (2018) used the cost drivers from COCOMO II in combination with story points to build a framework for effort estimation. The cost drivers and story points are used to create fuzzy sets. The framework is designed to be used in two phases of the project: the early stages for early estimations and later for supporting estimation of iterations. Using the model, the value of PRED was increased from 70% to 80%.

5.3.2 Improving existing estimates

Panda et al. (2015) utilized soft computing techniques, more specifically neural networks, for enhancing existing effort estimations. The original estimations were created using regression and they were based on project data consisting of three dimensions: story points, actual effort and team velocity. Four different neural network models were tested: *general regression neural networks*, *probabilistic neural networks*, *GMDH polynomial neural networks* and *cascade correlation neural networks*. Cascade correlation neural networks outperformed the others in optimizing the results.

A similar approach for optimizing estimations was taken by Satapathy and Rath (2017). Existing effort estimates created with the story point approach was further improved with machine learning techniques. The following techniques were tested on the dataset: *Decision trees*, *Stochastic gradient boosting* and *Random forest*. The results were compared and the Stochastic gradient boosting technique was identified as the most suitable technique giving the best results.

5.3.3 Support for the development team

Publications under this category present machine learning models that can be used by the team. Instead of just explaining the model, the publication is expected to provide clarification on how the method or model has been used, or is intended to be used by the development team.

In the article by Porru (2016), a classifier that supports the team with making story point estimations is presented. The classifier is designed to be used within the team and portrayed as an extra team member. The estimates produced should be constructed in real-time to be useful during the planning poker session. The resulting classifier uses the summary, description, name and the type of issues to

make predictions. The solution achieved a MMRE between 0.16 and 0.61, which corresponds to estimations of humans. The advantage of using the classifier is the speed of creating estimations, which is less than 6 seconds, and the fact that it is not affected by human bias.

Dantas et al. (2019) used a similar approach and studied the use of a tool that provides the team with support in estimation sessions. The tool uses historical project data to create a decision tree. A web user interface was implemented for the tool to enable easy access from mobile phones and computers. It has been tested by 21 professionals who found it easy and useful during the estimation sessions.

Tanveer et al. (2018) used impact analysis techniques to support the team using planning poker. A prototype based on a model and a method using impact analysis was tested in a live iteration of a case company. The estimates made by the prototype could then be used by the team members as a base for their own estimation. An expert-based estimation model together with the method created was found more accurate than a purely data-driven model.

5.3.4 Machine learning models for effort estimation

This category contains publications presenting machine learning models for effort estimation. The difference with section 4.3.3 (Support for the development team) is that the publication does not provide insights into the practical application of the model in team work.

Both Dragievic et al. (2017) and Ratke et al. (2019) used Bayesian networks for creating an effort estimation model. Ratke et al. (2019) trained the Bayesian network for effort estimation using text analysis of user stories. An accuracy of 81% was achieved. Unfortunately the language quality of the article is really poor, which prohibits a more detailed analysis. Dragievic et al. (2017) saw the minimal impact on agility and its suitability for all different agile methods as important requirements. The finished model was seen to meet all these requirements and achieved a good accuracy of almost 100%, using a test set of 160 tasks from real agile projects.

Scott and Pfahl (2018) studied how developer features could be used in models for effort estimation. Several different models were used and the results were compared with models using textual features. Models using developer features outperformed models using text features in the overall comparison, but the opposite results were also observed in some individual projects.

Moharreri et al. (2016) tried to develop a model that could beat the accuracy of planning poker. Several classification methods were used, such as Random forest, Naive Bayes and J48 decision tree. J48 did outperform manual planning poker alone. Using these methods in combination did produce even better results than using auto-estimation or planning poker alone.

Ramessur and Nagowah (2021) used the sprint in Scrum as the scope for their estimation model. A survey was conducted on what factors were experienced to

affect the effort estimation in a sprint. Based on the results, twelve factors were chosen and a model based on regression was built. The model was tested with several different regression algorithms: *linear regression*, *k-nearest neighbors*, *decision tree* and *support vector regression*. MLP (Artifial neural network) was chosen for the final model as it outperformed the other alternatives in accuracy.

Bilgayan et al. (2019) applied the Elman neural network and ANN-feedforward back-propagation neural network for a dataset including 21 projects using agile methodologies. The results of the estimations are compared to results presented in existing literature, where the cascade correlation network has been identified as the most accurate model. Unlike previous literature, the feedforward back-propagation network provided the best results.

5.4 Soft computing and planning poker

The effort estimation methods have been classified in literature by the estimation principles and the input data. This classification results in three categories: Expert-based, Data-driven and Hybrid [Sudarmaningtyas and Mohamed, 2021]. The most straightforward way to take advantage of the soft computing techniques in planning poker is to approach the subject with hybrid methods.

The potential of using hybrid estimation methods is also evident in the study by Moharreri et al. (2016). Different machine learning approaches were tested to find a model that would be more efficient than manual planning poker. The results indicate that the machine learning models are more accurate than planning poker alone, but the most accurate solution was achieved by using manual planning poker in combination with machine learning.

The articles presented in section 4.3.3 ([Porru, 2016], [Tanveer et al., 2018], [Dantas et al., 2019]) are all selected because they were identified as presenting a tool or framework that a development team could use for effort estimation. Instead of only presenting the model, it has also been considered how the team could use it as a tool. Porru et al. (2016) and Tanveer et al. (2018) directly mentioned that the tools could be used in planning poker. Although the article by Dantas et al. (2019) does not explicitly mention planning poker, it is clear that the tool could be utilized in this particular method.

6 Discussion

In this section, the results are reflected to the research questions and further elaborated. The main findings of this work are focused on two themes. The first theme answers how and in what direction the research work has progressed since the related literature reviews presented in the background section [Bilgaiyan et al., 2016]. The response to this theme has been formed by responding to RQ1, RQ2 and RQ3, and combining the results. The second theme revolves around RQ4, which is formulated to explore how soft computing could be utilized for planning poker. The categories created for RQ3 are also working as a foundation for the second theme.

6.1 Research direction

Considering RQ1, the soft computing techniques discussed in the literature have been identified from the selected papers. The list provides an overview of what technologies have been used in the context of effort estimation using soft computing. The resulting list includes 23 techniques. Combining this list with the results of RQ2, we can see what techniques have been used since the related literature reviews by Bilgayan et al. (2016) and Arora et al. (2020). We can also identify an increase in research work in general after the publication date of both studies.

Regarding machine learning techniques studied in the review by Arora et al. (2020), several new techniques were mentioned in more recent studies. The following techniques are discussed in papers published after 2017 and not mentioned in the review by Arora et al. (2020): *convolutional Neural Network* [Ochodek et al., 2020], *Support vector machines* [Ramessur and Nagowah, 2021] [Scott and Pfahl, 2018], *Linear regression* [Ramessur and Nagowah, 2021], *k-nearest neighbors* [Ramessur and Nagowah, 2021], *Elman network* [Bilgaiyan et al., 2019], *Feedforward backpropagation network* [Bilgaiyan et al., 2019] and *Gradiant boosted trees* [Tanveer et al., 2018]. Among these publications, one can be found in the category of "*Soft computing with a non-agile method or model*", three in "*machine learning models for effort estimation*" and one in "*support for development team*".

Looking at the evolution of publications since the literature review on soft computing in agile cost estimation by Bilgayan et al. (2016), we can see that all but one publication in this literature review is new. The only publication [Panda et al., 2015] in common is discussing different neural networks for improving estimations and falls in the category of "*Improving existing estimates*". Bilgayan et al. (2016) concludes in his work that very less work is done in the field of cost and effort estimation of agile software projects, even though the popularity of agile methodologies is continuing to rise. Based on this literature review, this research gap has been addressed with an increasing amount of research publications since 2016.

The fact that the genetic algorithms and expert systems were not explicitly mentioned in any of the articles as solutions is noteworthy. One explanation for the lack of research on expert systems in this field may be that it is considered outdated.

As mentioned in the background section [Nettleton, 2014], the philosophy of transferring knowledge from experts to computers, rather than trusting that the truth is found in data, has been abandoned in modern data analysis. Although genetic algorithms are not proposed as a solution for effort estimation, they have been mentioned as a further development proposal in the article by Ratke et al. (2019). He suggests that a genetic algorithm could be used as a training algorithm for tuning parameters of fuzzy sets.

6.2 Using soft computing for planning poker

Based on the background section, planning poker is the most popular estimation method for teams using agile methodologies. RQ4 was intended to examine whether ways to support planning poker with soft computing have been presented in the literature. The choice of concentrating on planning poker was also supported by the fact that refining it with soft computing would preserve the agile value of the team being involved at every stage. Mentions of hybrid techniques were already found in the background section. Hybrid techniques refer to methods in which both data and expert judgment are utilized. As a result, the author's expectation was that combinations would also be found where soft computing is used.

As it can be seen in table 7., the majority of the publications are categorized in the category "*machine learning models for effort estimation*" and only three publications can be found in *"support for development team"*. The major difference between the categories is that in *"support for development team"* the proposed solution is also presented from a practical view, including how the development team can use it. This means that many of the presented models based on machine learning could possibly be utilized in planning poker, but the publications do not comment on this.

However, the tools developed by Dantas et al. (2019) and Porru et al. (2016) are worth considering. Dantas et al.'s (2019) tool provides the development team with historical data during the estimation process. The tool has an user interface that can be used on mobile phones and computers. It has been tested with a development team in practice, and the process is described well in the article. Porru et al.'s (2016) classifier lacks a user interface, but it is fundamentally designed to be used in planning poker. The tool is described to work as an extra team member making its own estimations, and thus creating value for the team using soft computing.

6.3 Future work

One of the main goals of this work was to find out if tools or frameworks have been discussed in the literature, that would allow agile development teams to utilize soft computing for effort estimation. Based on the results, the research work could be continued within this theme. The following aspects are suggested in particular: data collection, data management and the user experience.

In order to follow the agile values, the team should be involved in the effort estima-

tion process. Therefore, the solution must be designed as part of a human-oriented process. In the literature, the most concrete solution was presented by Dantas et al. (2019). The suggested solution included an user interface that could be used with different devices. In addition to further developing and testing the user interface, possible integration with known task management systems should be explored. It is also necessary to design and experiment with different estimation routines around the device in order to find the best result.

With machine learning models at the center of the literature, the related limitations should be recognized. The collection of training data should be automated as far as possible, so that the project's resources do not have to be reserved for it. In order to ease this burden, it should be explored how the challenges related to data collection could be resolved using a shared database for projects on a company level. The solution may require a universal standard for writing issues so that they can be efficiently processed with machine learning. The standard could consider project and team features, of which the possibilities are presented in the study by Scott et al. (2018). The study [Porru, 2016] on estimating story points from issue reports could be taken into account in the data processing. A common standard could also create opportunities for cooperation between companies.

6.4 Threats to validity

6.4.1 Categorising and analysis

The extraction of soft computing techniques from the literature was conducted only by the author. For an improved process, a second reviewer can be used. Also, the classification of the utilization of soft computing in agile software projects was conducted by the author alone, introducing a risk for subjective bias.

6.4.2 Data collection

The data collection method included only one search string used in popular databases. Snowballing could be used to ensure that relevant papers are not ignored.

As the inclusion criteria allowed only papers discussing soft computing and effort estimation in the context of agile methodologies, relevant papers may have been excluded. The effort estimation in software development is a popular topic, and there might be publications dealing with the subject outside of agile methodologies that could be applied to this work.

As a concept, soft computing partially overlaps with both artificial intelligence and machine learning. There are no clear lines between different categories, which also leads to a situation where relevant literature may be ignored. For the concepts mentioned above and others similar, a corresponding search should be made, where the results are checked for compatible content.

7 Conclusion

This thesis has been conducted as a literature review on the utilization of soft computing techniques in the context of agile software effort estimation. The study resolved two main objectives. The first goal was to understand what is the state-of-art in utilizing soft computing for effort estimation in agile projects, and reflecting on how the situation has changed since the previous literature reviews conducted in 2016 and 2018. The second goal was to find out if soft computing can be used to support development teams in planning poker.

The key findings to the first objective is that 21 relevant publications have been published since 2016. Several new techniques have been used for the effort estimation both after 2016 and 2018. The literature was divided in the following categories: (6) machine learning models for effort estimation, (3) solutions to support the development team in effort estimation, (2) improving existing estimates, (3) soft computing used with a non-agile method or model and (4) literature reviews. Genetic algorithms and expert systems were not explicitly mentioned as techniques providing solutions in the literature.

Regarding the utilization of soft computing to support development teams in planning poker resulted in the following findings: Only three of the selected publications included solutions that had been used, or were intended to be used together with planning poker. One of the potential solutions was a tool with a user interface that provides estimations based on historical data for the team in estimation sessions. Another solution was based on a classifier that was designed to be treated as an extra team member during the planning poker session.

Based on the results, two future research topics are proposed. The estimation tools for development teams mentioned in the literature are considered promising and their further development and testing is suggested. The second topic is the development of a common issue reporting standard, which would improve the data collection between projects.

References

EPB, 2016 (2016). Dictionary britannica. https://www.britannica.com/technology/expert-system. Accessed: 2022-6-8.

dic, 2016 (2016). Merriam-webster's dictionary. `https://www.merriam-webster.com/dictionary/estimate`. Accessed: 2022-16-7.

IEE, 2021 (2021). Ieee. https://www.ieee.org/about/at-a-glance.html: :text=IEEE%20is%20the%20world's%20largest,and%20professional%20and%20educational´%20activities.

ACM, 2022 (2022). Acm digital library. `https://dl.acm.org/about`. Accessed: 2022-26-5.

Sci, 2022 (2022). Explore scientific, technical, and medical research on sciencedirect. `https://www.sciencedirect.com/`. Accessed: 2022-26-5.

IEE, 2022 (2022). Ieeexplore digital library. https://innovate.ieee.org/about-the-ieee-xplore-digital-library/: :text=The%20IEEE%20Xplore%20digital%20library,research%2C%20and%20inspire%20new%20ideas. Accessed: 2022-26-5.

Jav, 2022 (2022). javapoint.com. https://www.javatpoint.com/regression-vs-classification-in-machine-learning. Accessed: 2022-6-8.

ppC, 2022 (2022). Planning poker®. `https://www.mountaingoatsoftware.com/tools/planning-poker`. Accessed: 2022-21-7.

Sco, 2022 (2022). Scopus preview. `https://www.scopus.com/home.uri`. Accessed: 2022-26-5.

Aranda and Easterbrook, 2005 Aranda, J. and Easterbrook, S. (2005). Anchoring and adjustment in software estimation. *In Proceedings of the 2005 European Software Engineering Conference Held Jointly with the 2005 International Symposium on Foundations of Software Engineering)*, page 346–355. https://doi.org/10.1145/1095430.1081761.

Aronson, 2003 Aronson, J. (2003). Expert systems. *Encyclopedia of Information Systems*, pages 277–289. https://doi.org/10.1016/B0-12-227240-4/00067-8.

Arora, 2021 Arora, M. (2021). A state of the art regressor model's comparison for effort estimation of agile software. *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*, pages 211–215. 10.1109/ICIEM51511.2021.9445345.

Arora and Verma, 2018 Arora, M. and Verma, S. (2018). An efficient effort and cost estimation framework for scrum based projects. *Cognitive Informatics and Soft Computing*, pages 52–57. 10.14419/ijet.v7i4.12.20992.

Arora et al., 2020 Arora, M., Verma, S., and Chopra, S. (2020). A systematic literature review of machine learning estimation approaches in scrum projects. *Cognitive Informatics and Soft Computing*, pages 573–586. 10.1007/978-981-15-1451-7_59.

Assi et al., 2018 Assi, M., Halawi, B., and R., H. (2018). Genetic algorithm analysis using the graph coloring method for solving the university timetable problem. *Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem*, 126:899–906. https://doi.org/10.1016/j.procs.2018.08.024.

Auwatanamongkol, 2000 Auwatanamongkol, S. (2000). Pattern recognition using genetic algorithm. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volume 1, pages 822–828 vol.1. 10.1109/CEC.2000.870384.

Beck et al., 2001 Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., C. Martin, R., Mellor, S., Schwaber, K., Sutherland, K., and Thomas, D. (2001). Agile manifesto.

Bilgaiyan et al., 2016 Bilgaiyan, S., Mishra, S., and Das, M. (2016). Review of software cost estimation in agile software development using soft computing techniques. *2016 International Conference on Computational Intelligence and Networks*. 10.1109/CINE.2016.27.

Bilgaiyan et al., 2019 Bilgaiyan, S., Mishra, S., and Das, M. (2019). Effort estimation in agile software development using experimental validation of neural network models. *International Journal of Information Technology*, pages 569–573. https://doi.org/10.1007/s41870-018-0131-2.

Boehm, 1984 Boehm, B. W. (1984). Software engineering economics. *IEEE Transactions on Software Engineering*, SE-10(1):4–21. 10.1109/TSE.1984.5010193.

Broesch, 2009 Broesch, J. (2009). Chapter 7 - applications of dsp. pages 125–134. https://doi.org/10.1016/B978-0-7506-8976-2.00007-9.

Chen and Poo, 2003 Chen, C. and Poo, A. (2003). Engineering, artificial intelligence in. In *Encyclopedia of Information Systems*, pages 141–155. https://doi.org/10.1016/B0-12-227240-4/00058-7.

Collabnetversionone, 2018 Collabnetversionone (2018). The 12th annual state of agile.

Dantas et al., 2019 Dantas, E., Costa, A., Vinicius, M., Perkusich, M., Almeida, H., and Perkusich, A. (2019). An effort estimation support tool for agile software development: An empirical evaluation. *The 31st International Conference on Software Engineering and Knowledge Engineering*. 10.18293/SEKE2019-141.

Dragicevic et al., 2017 Dragicevic, S., Celar, S., and Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *The Journal of Systems and Software 127*, pages 109–119. https://doi.org/10.1016/j.jss.2017.01.027.

Gandomani et al., 2020 Gandomani, T., Tavakoli, Z., Zulzalil, H., and Farzani, H. (2020). The role of project manager in agile software teams: A systematic literature review. 10.1109/ACCESS.2020.3004450.

Halkjelsvik and Jørgensen, 2012 Halkjelsvik, T. and Jørgensen, M. (2012). From origami to software development: A review of studies on judgment-based predictions of performance. *Psychol. Bull.*, page 238–271. http://dx.doi.org/10.1037/a0025996.

Haugen, 2006 Haugen, N. (2006). An empirical study of using planning poker for user story estimation. *AGILE 2006 (AGILE'06)*. 10.1109/AGILE.2006.16.

Hussain et al., 2013 Hussain, I., Kosseim, L., and Ormandjieva, O. (2013). Approximation of cosmic functional size to support early effort estimation in agile. *Data Knowledge Engineering*, pages 2–14. 10.1016/j.datak.2012.06.005.

Ibrahim, 2016 Ibrahim, d. (2016). An overview of soft computing. *Procedia Computer Science 102*, page 34–38. 10.1016/j.procs.2016.09.366.

Jalil and Shahid, 2008 Jalil, Z. and Shahid, A. A. (2008). Is non technical person a better software project manager? In *2008 International Conference on Computer Science and Software Engineering*, volume 2, pages 1–5. 10.1109/CSSE.2008.1125.

Jiang et al., 2020 Jiang, T., Gradus, J., and Rosellini, A. (2020). Supervised machine learning: A brief primer. (5):675–687. 10.1016/j.beth.2020.05.002.

Jørgensen, 2004 Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70:37–60. 10.1016/S0164-1212(02)00156-5.

Jørgensen, 2014 Jørgensen, M. (2014). What we do and don't know about software development effort estimation. *IEEE Software*, 31(2):37–40. 10.1109/MS.2014.49.

Jørgensen et al., 2009 Jørgensen, M., Boehm, B., and Rifkin, S. (2009). Software development effort estimation: Formal models or expert judgment? *IEEE Software*, 26(2):14–19. 10.1109/MS.2009.47.

Kitchenham and Charters, 2007 Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. *EBSE Technical Report EBSE-2007-01.*

Kononenko and M, 2007 Kononenko, I. and M, K. (2007). Machine learning and data mining. pages 321–358. https://doi.org/10.1533/9780857099440.321.

Kotsiantis, 2007 Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. page 3–24, NLD. IOS Press.

Ludwig, 2004 Ludwig, B. (2004). Fuzzy logic modeling of energy systems. *Encyclopedia of Energy*. https://doi.org/10.1016/B0-12-176480-X/00547-7.

Matsubara et al., 2022 Matsubara, P., Gadelha, B., Steinmacher, I., and Conte, T. (2022). Sextamt: A systematic map to navigate the wide seas of factors affecting expert judgment software estimates. *The Journal of Systems Software.* https://doi.org/10.1016/j.jss.2021.111148.

Moharreri et al., 2016 Moharreri, K., Sapre, A., Ramanathan, J., and Ramnath, R. (2016). Cost-effective supervised learning models for software effort estimation in agile environments. *2016 IEEE 40th Annual Computer Software and Applications Conference*. 10.1109/COMPSAC.2016.85.

Nettleton, 2014 Nettleton, D. (2014). Commercial data mining. pages 159–170. https://doi.org/10.1016/B978-0-12-416602-8.00010-8.

Ochodek et al., 2020 Ochodek, M., Kopczynska, S., and Staron, M. (2020). Deep learning model for end-to-end approximation of cosmic functional size based on use-case names. *Information and Software Technology 123*. https://doi.org/10.1016/j.infsof.2020.106310.

Panda et al., 2015 Panda, A., Shasank, S., and Santanu, R. (2015). Empirical validation of neural network models for agile software effort estimation based on story points. *Procedia Computer Science 57 (2015)*, page 772 – 781. 10.1016/j.procs.2015.07.474.

Pandey, 2013 Pandey, P. (2013). Analysis of the techniques for software cost estimation. In *2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT)*, pages 16–19. 10.1109/ACCT.2013.13.

Perkusich, 2020 Perkusich, M. (2020). Intelligent software engineering in the context of agile software development: A systematic literature review. *Information and Software Technology 119*. https://doi.org/10.1016/j.infsof.2019.106241.

Porru, 2016 Porru, S. (2016). Estimating story points from issue reports. *PROMISE 2016: Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 1–10. https://doi.org/10.1145/2972958.2972959.

Qi et al., 2018 Qi, K., Hira, A., Venson, E., and Boehm, B. (2018). Calibrating use case points using bayesian analysis. *ESEM '18: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10. https://doi.org/10.1145/3239235.3239236.

Ramessur and Nagowah, 2021 Ramessur, A. and Nagowah, S. (2021). A predictive model to estimate effort in a sprint using machine learning techniquess. *International Journal of Information Technology*, pages 1101–1110. https://doi.org/10.1007/s41870-021-00669-z.

Rao and G., 2011 Rao, K. and G., R. (2011). *An Overview on Soft Computing Techniques*. Addison-Wesley Professional.

Raslan and Nagy, 2018 Raslan, A. and Nagy, D. (2018). An enhanced framework for effort estimation of agile projects. *International Journal of intelligent engineering systems)*, pages 205–2014. 10.22266/ijies2018.0630.22.

Ratke et al., 2019 Ratke, C., Hoffmann, H., Gaspar, T., and Floriani, P. (2019). Effort estimation using bayesian networks for agile development. *Agile Development 2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–4. 10.1109/CAIS.2019.8769455.

Ravi and S., 2021 Ravi, M. and S., M. (2021). Study on agile story point estimation techniques and challenges. *International Journal of Computer Applications*, 174.

Saini et al., 2018 Saini, A., Laxmi, A., and Sunil, K. (2018). Effort estimation of agile development using fuzzy logic. *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pages 779–783. 10.1109/ICRITO.2018.8748381.

Salamanoglu et al., 2018 Salamanoglu, M., Hacaloglu, T., and Demirors, O. (2018). Effort estimation for agile software development: comparative case studies using cosmic functional size measurement and story points. *IWSM Mensura '17: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, pages 41–49. https://doi.org/10.1145/3143434.3143450.

Samuel, 1959 Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229. 10.1147/rd.33.0210.

Satapathy and Rath, 2017 Satapathy, S. and Rath, S. (2017). Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innovations Syst Softw Eng (2017)*, pages 191–200. 10.1007/s11334-017-0288-z.

Schwaber and Sutherland, 2020 Schwaber, K. and Sutherland, J. (2020). The scrum guide.

Scott and Pfahl, 2018 Scott, E. and Pfahl, D. (2018). Using developers' features to estimate story points. *ICSSP '18: Proceedings of the 2018 International Conference on Software and System Process*, pages 106–111. https://doi.org/10.1145/3202710.3203160.

Sudarmaningtyas and Mohamed, 2021 Sudarmaningtyas, P. and Mohamed, R. (2021). A review article on software effort estimation in agile methodology. *SCIENCE TECHNOLOGY, Pertanika journal*, pages 837–861. https://doi.org/10.47836/pjst.29.2.08.

Sudarmaningtyas and Mohamed, 2020 Sudarmaningtyas, P. and Mohamed, R. B. (2020). Extended planning poker: A proposed model. In *2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pages 179–184. 10.1109/ICITACEE50144.2020.9239165.

Tanveer et al., 2018 Tanveer, A., Vollmer, A., and Braun, S. (2018). A hybrid methodology for effort estimation in agile development. *ICSSP '18: Proceedings of the 2018 International Conference on Software and System Process*, pages 21–30. https://doi.org/10.1145/3202710.3203152.

Trendowicz and Jeffery, 2014 Trendowicz, A. and Jeffery, R. (2014). *Software project effort estimation: Foundation and best practice guidelines for success.*

Usman et al., 2014 Usman, M., Mendes, E., Weidth, F., and Britto, R. (2014). Effort estimation in agile software development: a systematic literature review. *PROMISE '14: Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, page 82–91. https://doi.org/10.1145/2639490.2639503.

Wang et al., 2009 Wang, H., Ma, C., and Zhou, L. (2009). A brief review of machine learning and its application. In *2009 International Conference on Information Engineering and Computer Science*, pages 1–4. 10.1109/ICIECS.2009.5362936.

Yang, 2020 Yang, X. (2020). *Nature-Inspired Optimization Algorithms.* Elsevier. https://doi.org/10.1016/B978-0-12-416743-8.00005-1.

Zadeh, 1965 Zadeh, L. (1965). Fuzzy sets. *INFORATION AND CONTROL*, pages 338–353.

Zadeh, 1988 Zadeh, L. (1988). Fuzzy logic. *Computer*, pages 83 – 93. https://doi.org/10.1145/1095430.1081761.

Zhang, 2010 Zhang, P. (2010). Advanced industrial control technology. pages 257–305. https://doi.org/10.1016/C2009-0-20337-0.