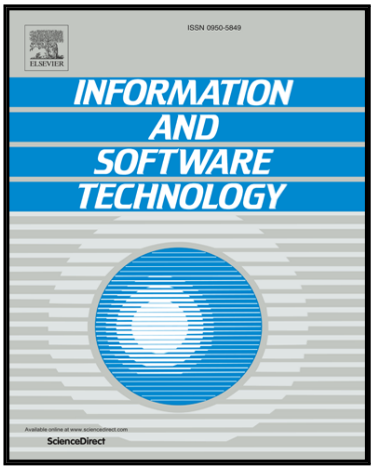
Journal Pre-proof



Intelligent Software Engineering in the Context of Agile Software Development: a Systematic Literature Review

Mirko Perkusich, Lenardo Chaves e Silva, Alexandre Costa, Felipe Ramos, Renata Saraiva, Arthur Freire, Ednaldo Dilorenzo, Emanuel Dantas, Danilo Santos, Kyller Gorgônio, Hyggo Almeida, Angelo Perkusich

PII: S0950-5849(19)30258-7
DOI: https://doi.org/10.1016/j.infsof.2019.106241
Reference: INFSOF 106241

To appear in: *Information and Software Technology*

Received date: 16 February 2019
Revised date: 21 September 2019
Accepted date: 28 November 2019

Please cite this article as: Mirko Perkusich, Lenardo Chaves e Silva, Alexandre Costa, Felipe Ramos, Renata Saraiva, Arthur Freire, Ednaldo Dilorenzo, Emanuel Dantas, Danilo Santos, Kyller Gorgônio, Hyggo Almeida, Angelo Perkusich, Intelligent Software Engineering in the Context of Agile Software Development: a Systematic Literature Review, *Information and Software Technology* (2019), doi: https://doi.org/10.1016/j.infsof.2019.106241

Intelligent Software Engineering in the Context of Agile Software Development: a Systematic Literature Review

Mirko Perkusich[a], Lenardo Chaves e Silva[a], Alexandre Costa[a], Felipe Ramos[a], Renata Saraiva[a], Arthur Freire[a], Ednaldo Dilorenzo[a], Emanuel Dantas[a], Danilo Santos[a], Kyller Gorgônio[a], Hyggo Almeida[a], Angelo Perkusich[a]

[a]*Federal University of Campina Grande, Campina Grande, PB, Brazil*

Abstract

CONTEXT: Intelligent Software Engineering (ISE) refers to the application of intelligent techniques to software engineering. We define an "intelligent technique" as a technique that explores data (from digital artifacts or domain experts) for knowledge discovery, reasoning, learning, planning, natural language processing, perception or supporting decision-making.

OBJECTIVE: The purpose of this study is to synthesize and analyze the state of the art of the field of applying intelligent techniques to Agile Software Development (ASD). Furthermore, we assess its maturity and identify adoption risks.

METHOD: Using a systematic literature review, we identified 104 primary studies, resulting in 93 unique studies.

RESULTS: We identified that there is a positive trend in the number of studies applying intelligent techniques to ASD. Also, we determined that reasoning under uncertainty (mainly, Bayesian network), search-based solutions,

Email addresses: mirko@embedded.ufcg.edu.br (Mirko Perkusich), lenardo.silva@embedded.ufcg.edu.br (Lenardo Chaves e Silva), alexandre.costa@embedded.ufcg.edu.br (Alexandre Costa), felipe.ramos@embedded.ufcg.edu.br (Felipe Ramos), renata.saraiva@embedded.ufcg.edu.br (Renata Saraiva), arthur.freire@embedded.ufcg.edu.br (Arthur Freire), ednaldo.dilorenzo@virtus.ufcg.edu.br (Ednaldo Dilorenzo), emanuel.dantas@embedded.ufcg.edu.br (Emanuel Dantas), danilo.santos@embedded.ufcg.edu.br (Danilo Santos), kyller@embedded.ufcg.edu.br (Kyller Gorgônio), hyggo@embedded.ufcg.edu.br (Hyggo Almeida), perkusic@virtus.ufcg.edu.br (Angelo Perkusich)

and machine learning are the most popular intelligent techniques in the context of ASD. In terms of purposes, the most popular ones are effort estimation, requirements prioritization, resource allocation, requirements selection, and requirements management. Furthermore, we discovered that the primary goal of applying intelligent techniques is to support decision making. As a consequence, the adoption risks in terms of the safety of the current solutions are low. Finally, we highlight the trend of using explainable intelligent techniques.

CONCLUSION: Overall, although the topic area is up-and-coming, for many areas of application, it is still in its infancy. So, this means that there is a need for more empirical studies, and there are a plethora of new opportunities for researchers.

Keywords: Intelligent Software Engineering, Agile Software Development, Search-based Software Engineering, Machine Learning, Bayesian networks, Artificial intelligence

1. Introduction

Since its inception, the primary goal of software engineering is to improve software quality and productivity. For this purpose, many initiatives have emerged, including maturity models, formal methods [1], reuse-driven software engineering [2], and value-based software engineering [3].

More recently, with the increasing amount of data generated by tools such as software versioning systems (e.g., Git[1]), build management systems (e.g., Jenkins[2]) and project management platforms (e.g., Jira[3]), an emergent field is appearing: the Intelligent Software Engineering (ISE). Xie [4, 5] defines ISE with two perspectives. First, as the application of artificial intelligence (AI) technologies to software engineering. Second, as the development of software engineering solutions for intelligent software. In this study, we only focus on

[1]https://git-scm.com/
[2]https://jenkins.io/
[3]https://www.atlassian.com/br/software/jira

Xie's first perspective. We complement it by defining the term "intelligent technique" as the exploration of data (from digital artifacts or domain experts) for knowledge discovery, reasoning, learning, planning, natural language processing, perception or supporting decision-making. Therefore, in the context of this work, data mining, fuzzy logic, machine learning, expert systems, and search algorithms (e.g., swarm intelligence, evolutionary algorithms) are examples of intelligent techniques. On the other hand, we do not consider formal methods, software process simulation modeling [6], and model-driven development [7] as intelligent techniques. In this context, examples of themes included under ISE are the application of search and optimization [8], machine learning [9], recommender systems [10], Bayesian networks [11], software analytics [12], big data analysis [13] and decision analysis in software engineering [14, 15].

The rising interest on ISE is evidenced by conferences such as the International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE), International Conference on Software Engineering and Knowledge Engineering (SEKE), the International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), which takes place with the International Conference on Software Engineering (ICSE); and the International Workshop on Intelligent Software Engineering (WISE), which took place with the IEEE/ACM International Conference on Automated Software Engineering (ASE) in 2017. This is also evidenced by the number of research groups that explored and discussed the application of AI to software engineering [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. For instance, in 2012, Harman [26] classified AI techniques applied to Software Engineering into three facets: (i) computational search and optimization techniques, (ii) fuzzy and probabilistic methods for uncertain reasoning, and (iii) classification, learning and prediction.

In this work, we focus on synthesizing the ISE field restricted to a specific software development paradigm: Agile Software Development (ASD). According to Hoda et al. [31], ASD, formally introduced in 2001 through the "Agile Manifesto", is "the mainstream development method of choice worldwide." As a consequence, the research community has witnessed this impact on the copious number of primary and secondary studies on ASD [32, 31].

ASD is a change-driven approach to develop software in the context of volatile requirements [32]. It focuses on collaboration and fast delivery of working software to, empirically, learn the needs of the customers, and deliver valuable products[4] [33]. Given this, there is a need to make decisions and optimization constantly. Examples of decisions that should be taken and optimizations to be made are:

- Which practices should we use to improve the likelihood of satisfying customers?
- Which product backlog items should we deliver this sprint/release?
- Who should be part of the team?
- What do we need to improve in our team and process?
- How fast will the team deliver a given feature?
- How to optimize regression testing without decreasing confidence in Continuous Integration?

Intelligent techniques have been applied for several purposes in ASD. For instance, Bayesian networks have been used to predict the velocity of Extreme Programming projects [34], assist in process improvement [35] and estimate effort [36]. Machine learning techniques, such as neural networks, have been used to estimate effort [37], assist in release planning [38] and process tailoring [39]. SBSE solutions have been used to prioritize requirements [40], assist in release planning [41] and allocate human resources [42]. Graph theory has been applied to optimize regression testing [43] and process improvement [44].

Based on this scenario, this article systematically surveys the area of ISE in the context of ASD to synthesize and analyze the state of the art, assess its

[4] https://agilemanifesto.org/

maturity, and identify adoption risks. Given this, we have defined the following research questions:

- RQ1: What is the current state of the art on intelligent techniques applied to ASD?
- RQ2: What is the maturity of the existing solutions applying intelligent techniques to ASD?
- RQ3: What are the risks of adopting the current intelligent techniques for ASD?

We follow the Systematic Literature Review (SLR) guideline, presented by Kitchenham et al. [45]. We believe that the relevance of this study is twofold. On the one hand, it enables practitioners to be updated with the field of ISE in the context of ASD. On the other hand, it allows researchers to identify topic areas where research is lacking or that have been extensively studied.

The rest of the article is structured as follows. Section 2 describes the systematic literature review protocol. Section 3 presents the results and analysis of the systematic literature review. Section 4 discusses the implications for research and practice. Section 5 discusses the threats to validity. Finally, Section 6 presents our final remarks.

2. Research Methodology

To identify and assess intelligent techniques applied to ASD, a systematic review was executed [45]. In what follows, we present the study's protocol.

2.1. Research questions

This research aims to synthesize and analyze the state of the art of intelligent techniques applied to ASD, assess its maturity, and identify adoption risks. For this purpose, we defined the research questions (RQs) shown in Table1.

Table 1: Research questions of the study.

Nr.	Research question	Aim
RQ 1	What is the state of the art on intelligent techniques applied to ASD?	To provide an overview on intelligent techniques applied to ASD
RQ 1.1	Which types of research contributions are provided by studies related to ISE and ASD?	To categorize available research on intelligent techniques applied to ASD according to research result type.
RQ 1.2	Which intelligent techniques are used in the context of ASD?	To deliver an overview of which intelligent techniques have been used in the context of ASD.
RQ 1.3	For which purposes are the intelligent techniques applied to ASD?	To deliver an overview of how and for which purposes has intelligent techniques been applied to ASD.
RQ 1.4	What is the publication frequency and topics?	To show a timeline of publications and trends.
RQ 2	What is the maturity of the existing solutions applying intelligent techniques to ASD?	To assess the maturity of the existing solutions applying intelligent techniques to ASD in terms of the empirical research type, research validation, and availability of datasets and tools.
RQ 3	What are the risks of adopting the current intelligent techniques for ASD?	To identify the risks (in terms of point of application, type of intelligent technique and level of automation benefits) of adopting the current intelligent techniques for ASD.

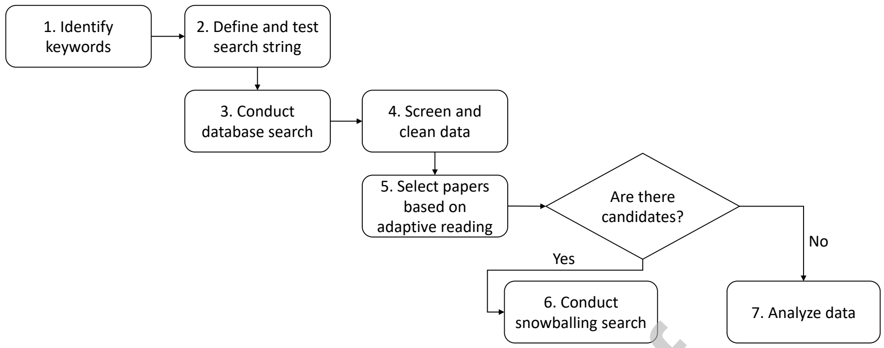


Figure 1: Overview of the review process.

2.2. Search method

Due to the inherent broad scope of the required search for this study, we executed a mixed-methods approach, in which we, first, performed a database search; and, second, snowballing (backward and forward) [46]. The resulting papers of the database search were used as the seed set for a snowballing search (backward and forward). An overview of the review process is presented in Figure 1.

2.2.1. Search terms

In **Step 1**, the goal was to identify the keywords to be used for the search string (**Step 2**) to conduct the database search (**Step 3**). The challenge is that many papers that use intelligent techniques such as "machine learning" or "artificial intelligence" do not use these terms in the paper itself. For instance, instead of using "artificial intelligence", Perkusich et al. [35] uses only "Bayesian networks". Therefore, using only generic terms is not enough. Our first step was to define the keywords from the list of terms from The 2012 ACM Computing Classification System. On the other hand, known keywords such as "ant colony optimization" is not included in the list.

Given this, as an attempt to minimize the probability of missing relevant studies, we complemented the set of keywords by applying a brute force approach. For this purpose, we manually explored highly-referenced journals and conferences in software engineering. We read the title, abstract and keywords of all articles from 2001 until 2016 of the following journals: Journal of Systems and Software, IEEE Transactions on Software Engineering, ACM Transactions on Software Engineering and Methodology, Empirical Software Engineering and Proceedings of the IEEE. Furthermore, we executed the same process for papers from ACM/IEEE International Conference on Software Engineering (ICSE).

Afterwards (**Step 2**), we classified the keywords identified in **Step 1** into two types: “intelligent techniques” and “agile software development”, resulting in the following subsets: I and A, where I is the subset of “intelligent techniques” terms and A, of “agile software development” terms. We merged I and A using the OR operation to form the search string. We are interested in $T = I \cup A$. Therefore, we used the AND operator to form T, the set of all keywords. Table 2 shows the resulting string.

Table 2: Search strings for data retrieval.

(“prediction model” OR Bayes OR BBN OR “Genetic algorithm” OR “System dynamics” OR “Case-based reasoning” OR “Rule-based reasoning” OR “Neural network” OR “Support Vector Machine” OR SVM OR “Multi-agent system” OR “Multiagent system” OR “Multiobjective learning” OR “Multi-objective learning” OR “Particle swarm optimization” OR “machine learning” OR cluster OR fuzzy OR fuzz OR “tree search” OR “rule learning” OR “causal model” OR “finite state machine” OR “artificial intelligence” OR “ant colony optimization” OR “Natural Language” OR “Link analysis” OR “Statistical analysis” OR regression OR “tabu search” OR “hill climbing” OR greedy OR ontology OR ontologies OR markov OR

"evolutionary algorithm" OR heuristic OR "theorem proving" OR "probabilistic model" OR "probabilistic network model" OR "multi-objective optimization" OR "decision tree" OR "recommendation system" OR tagging OR "random forest" OR "stochastic modelling" OR "stochastic modeling" OR "time spectra" OR "container profiling" OR "dependency graph" OR "graph theory" OR "social network analysis" OR "path profiling" OR "Tf-idf Technique" OR "Recommender system" OR "Logistic model" OR "Multilayer perceptron" OR "entropy-based algorithm" OR "statistical method" OR "statistical model" OR "Genetic granular classifier" OR "data analysis" OR "Grey relational analysis" OR "nearest neighbor" OR "nearest neighbour" OR "classification tree" OR "monte carlo simulation" OR "analogy-based estimation" OR "boltzmann machine" OR "petri net" OR "Linear Temporal Logic" OR knn OR k-nn OR "Latent semantic indexing" OR "complexity theory" OR "Model Checking" OR "Genetic programming" OR "search-based software engineering" OR "Latent Dirichlet Allocation" OR "critical incident technique" OR "n-gram statistic" OR "dictionary learning" OR "Multi-fact analysis" OR "Neurolinguistic Programming" OR "data mining" OR "Information extraction" OR "Semantic network" OR "Probabilistic reasoning" OR "Vagueness and fuzzy logic" OR "Ontology engineering" OR "Multi-agent planning" OR "Heuristic function construction" OR "Discrete space search" OR "Continuous space search" OR "Randomized search" OR "Game tree search" OR "Intelligent agent" OR Ranking OR "Supervised learning by classification" OR "Supervised learning by regression" OR "Reinforcement learning" OR "Gaussian process" OR "Inductive logic learning" OR "Statistical relational learning" OR "Maximum likelihood modelling" OR "Maximum likelihood modeling" OR "Maximum entropy modelling" OR "Maximum entropy modeling" OR "Maximum a posteriori modelling" OR "Maximum a posteriori modeling" OR "Mixture model" OR "Latent variable model" OR "matrix factorization" OR "Factorization method" OR "Instance-based learning" OR "Deep belief network" OR "Spectral method" OR "Feature selection" OR Regularization OR Cross-validation OR "Bayesian Network" OR "Markov Network"

OR “Factor Graph” OR “Decision Diagram” OR “Equational Model” OR “Causal Network” OR “Markov Chain” OR Bootstrapping OR Jackknifing OR “Regression Analysis” OR “Markov Process”) **AND** (software AND agile AND (scrum OR xp OR (crystal AND (clear OR orange OR red OR blue)) OR dsdm OR fdd OR “feature driven development” OR (lean AND development) OR refactoring OR “pair programming” OR “continuous integration” OR “continuous delivery” OR Kanban OR “user story” OR “story point” OR backlog OR “product owner” OR “test driven development” OR “extreme programming” OR “planning poker” OR devops))

To test the string, we checked if applying it to the data sources selected for the study (shown in Section 2.2.2) returned ten known relevant papers. The results are shown in Table 3. Only one paper was not returned, but we noticed that it was a limitation of the target databases (i.e., none of them indexed articles from the given journal). As shown in Section 3, the given paper was found after the first snowballing iteration. Finding the paper during the snowballing search is an evidence that it adds value to the research by minimizing the probability of missing relevant papers.

2.2.2. Data sources

After having a valid search string, we conducted the database search (**Step 3**) in the following digital libraries:

- ACM Digital Library
- Engineering Village
- ISI Web of Science
- Science Direct
- Scopus
- Springer

Table 3: Overview of the selected studies.

Paper Name	Result
A model to detect problems on Scrum-based software development projects [47]	OK
A procedure to detect problems of processes in software development projects using Bayesian networks [35]	OK
A Bayesian Network Model to Assess Agile Teams' Teamwork Quality [48]	OK
Ant colony optimization for the next release problem a comparative study [49]	OK
Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Point [50]	OK
A Bayesian based method for agile software development release planning and project health monitoring [51]	OK
Predicting project velocity in XP using a learning dynamic Bayesian network model [34]	OK
Bayesian network based xp process modelling [52]	NOK
A Lagrangian heuristic for sprint planning in agile software development [53]	OK
Multi-objective ant colony optimization for requirements selection [41]	OK

2.3. Selection criteria

To screen and clean data (**Step 4**), we defined a generic exclusion criteria:

1. A duplicate OR
2. Published in a non-peer reviewed channel (e.g., thesis) OR
3. Published in a book OR
4. Unavailable in English or Portuguese OR
5. Published before 2001.

Afterwards (**Step 5**), the remaining papers were evaluated through the following inclusion criteria:

- presents an intelligent technique applied to ASD *AND* is a primary study.

Therefore, we only included papers in which the authors explicitly stated that the proposed solution was useful for ASD. For instance, one could argue that the Quamoco approach, presented in Wagner et al. [54], could be useful for ASD. But, since the authors did not apply the solution in the given context or explicitly state it in the paper, we excluded the paper. This rule is also valid for popular agile practices such as *Refactoring*. Even though it is an Extreme Programming practice, and a popular application field of SBSE [55], *Refactoring* can be used on non-agile environments such as the maintenance phase of software projects [56, 57]. Therefore, if the paper only related to *Refactoring* or another practice and not explicitly related to ASD, it was excluded. A single researcher executed the screening and cleaning of data.

2.3.1. Selection Method

To select the papers, we used a two-stage approach. **First**, we evaluated the papers by applying the adaptive reading approach [58], which included at least two reviewers. Later, the full-text of the papers were read by two reviewers (data extractor and data checker) [59, 60]. They extracted the data from the papers and excluded them based on the same criteria used in the previous step and the quality criteria. We present details of the quality assessment and data extraction process in Sections 2.4 and 2.5, respectively.

The snowballing approach (**Step 6**) was performed with the papers identified during the database search as the seed set. For each paper in the seed set, we applied the backward and forward snowballing. Backward and forward snowballing refers to systematically analyzing the references and citations, respectively, of a given set of papers.

For the forward snowballing, we used Google Scholar and Scopus as the data sources. We applied the same approach as in **Steps 4** and **5** to select the papers and extract their data. For the backward snowballing, the papers were initially evaluated by only one reviewer. The reviewer was responsible for applying the generic exclusion criteria defined in **Step 4**, by evaluating the reference list for each paper, and, if necessary, the place of reference in the text. Later, for the included papers, we applied the same approach as in **Step 5** to select the papers and extract their data. After identifying that there were no more candidate papers, we analyzed the data regarding the Research Questions (RQs) (**Step 7**).

2.4. Quality assessment

To assess the empirical quality of the papers, we followed the criteria established by Dybå and Dingsøyr [61], which are presented as follows. To evaluate each of the criteria listed above, we used a boolean scale, in which a score of "1" means "yes" and "0", "no".

2.5. Data extraction and Classification Scheme

As follows, we present the data extracted from the papers.

(i) type of article (journal, conference),

(ii) name of the publication channel,

(iii) year of publication,

(iv) agile method (Scrum, Extreme Programming, Kanban, Crystal, Lean Software Development, Dynamic Systems Development Methodology, Adaptive Software Development, Feature Driven Development, Non-specified),

Table 4: Quality assessment criteria according to Dybå and Dingsøyr [61].

Type	Description
Research	Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)?
Aim	Is there a clear statement of the aims of the research?
Context	Is there an adequate description of the context in which the research was carried out?
Design	Was the research design appropriate to address the aims of the research?
Sampling	Was the recruitment strategy appropriate to the aims of the research?
Control	Was there a control group with which to compare treatments?
Collection	Was the data collected in a way that addressed the research issue?
Analysis	Was the data analysis sufficiently rigorous?
Reflexivity	Has the relationship between researcher and participants been considered to an adequate degree?
Findings	Is there a clear statement of findings?
Value	Is the paper of value for research or practice?

(v) research result,

(vi) empirical research type,

(vii) research validation,

(viii) name of the intelligent technique,

(ix) purpose.

(x) tool availability.

(xi) dataset availability.

(xii) point of application.

(xiii) level of automation.

For (v), we used the classification scheme presented by Shaw [62]: procedure or technique, report, qualitative or descriptive model, analytic model, tool or notation, specific solution prototype answer or judgment, or empirical model. In Table 5, we show the definition of each possible classification.

For (vi), we used the classification presented by Tonella et al. [63]: experiment, observational study, experience report, case study or systematic review. In Table 6, we show the definition of each possible classification.

For (vii), we used the classification scheme presented by Shaw [62]: analysis, evaluation, experience, example, persuasion or blatant assertion. In Table 7, we show the definition of each possible classification. We expect that more mature solutions will apply *analysis* as the validation approach.

For (viii), we extracted the names of the intelligent techniques (i.e., a paper might present multiple intelligent techniques) described in the paper. Afterward, we classified them by executing a thematic analysis approach as presented in [64], in which each paper was coded by one reviewer, and checked independently by another one. Afterward, one researcher translated the codes into themes, which were collaboratively reviewed by the other reviewers until consensus was achieved.

For each study, we only extracted intelligent techniques directly applied to solve a Software Engineering problem. For instance, in Kumar et al. [40], a *Genetic Algorithm* is used to define an ordered list of User Stories to assist in the

Table 5: Research result types according to Shaw [62].

Type	Description
Procedure or technique	New or improved way to do a task, such as design, implementation, maintenance, measurement, evaluation, selection from alternatives; includes techniques for implementation, representation, management, and analysis; a technique should be operational, not advice or guidelines, but a procedure.
Report	Interesting observations, rules of thumb, but not sufficiently general or systematic to rise to the level of a descriptive model.
Qualitative or descriptive model	Structure or taxonomy for a problem area; architectural style, framework, or design pattern; non-formal domain analysis, well-grounded checklists, well-argued informal generalizations, guidance for integrating other results, well-organized interesting observations.
Analytic model	Structural model that permits formal analysis or automatic manipulation.
Tool or notation	Implemented tool that embodies a technique; formal language to support a technique or model (should have a calculus, semantics, or other basis for computing or doing inference).
Specific solution prototype answer or judgment	Solution to application problem that shows application of SE principles, may be design, prototype, or full implementation; careful analysis of a system or its development, result of a specific analysis, evaluation, or comparison.
Empirical model	Empirical predictive model based on observed data.

Table 6: Empirical study types according to Tonella et al. [63].

Type	Description
Experiment	Controlled study to observe the outcomes and the factors involved on it. Requires observation of multiple cases.
Observational study	Gathers information to connect factors and effect variables. It is not controlled and requires observation of multiple cases. Normally is applied as a survey to collect data to be observed.
Experience report	Analysis of one case and there is no controlled context. The goal is to describe the success of the proposed/adopted technique and setup, data collection and analysis are not discussed in details.
Case study	Analysis of one case and setup, data collection and analysis are discussed in details.
Systematic review	Selects cases of the topic of interest studied in the past for evaluation and interpretation.

Table 7: Research validation types according to Shaw [62].

Type	Description
Analysis	Applies rigorous analysis based on data gathered on models and/or experiments.
Evaluation	Validation is based on a previous defined criteria or done by subject matter experts.
Experience	Real world results generated by someone other than the author can provide evidence of success.
Example	Focuses on an example to define how the real world problem can be resolved.
Persuasion	It is more about the idea than the results and normally has a lot of passion associated to it.
Blatant assertion	No results have been evaluated.

release planning of ASD projects; therefore, we extracted the technique *Genetic Algorithm* from this study. In Li and Leung [65], a *Genetic Algorithm* was used to learn the structure of a *Bayesian network*, which was used to predict the fault-proneness of object-oriented systems developed with ASD. In this case, *Genetic Algorithm* was a means to build the model (i.e., *Bayesian network*) that was used to solve the Software Engineering problem (i.e., predict the fault-proneness of object-oriented systems). Therefore, we only extracted the technique *Bayesian network* from this study. By applying the thematic analysis approach, it was classified as *Probabilistic methods for uncertain reasoning*.

An example of a study related to multiple intelligent techniques is Sobiech et al. [66]. For this study, we extracted the techniques *Analytic Hierarchy Process* (AHP) and *Knapsack problem-related heuristic*, which were classified, respectively, into *Multiple Criteria Decision Analysis* and *Search and Optimization*. In Figure 2, we show the classification scheme that we have defined for the intelligent techniques.

For (ix), we extracted the sentences of the purpose of the proposed solution and classified it according to the SWEBOK knowledge areas.

For (x), we classified each study into four possible types: open-source tool, proprietary tool, model, or none. We defined that for a study to be classified as “model”, it must present enough information to enable other researchers to replicate the proposed solution (e.g., pseudocode, algorithm or graph) at least, partially.

For (xi), we classified the studies regarding the availability of the dataset used for training the model, for data-driven solutions, or for validating the solution. We classified each study into three possible types: yes, partially, or no.

For (xii) and (xiii), we classified the studies concerning the point of application and level of automation as suggested by Feldt et al. [30]. Regarding the *point of application*, there are three possible levels: process, product, and runtime. The *process* level indicates that the intelligent technique “is applied in the development process and does not necessarily affect, directly, the source

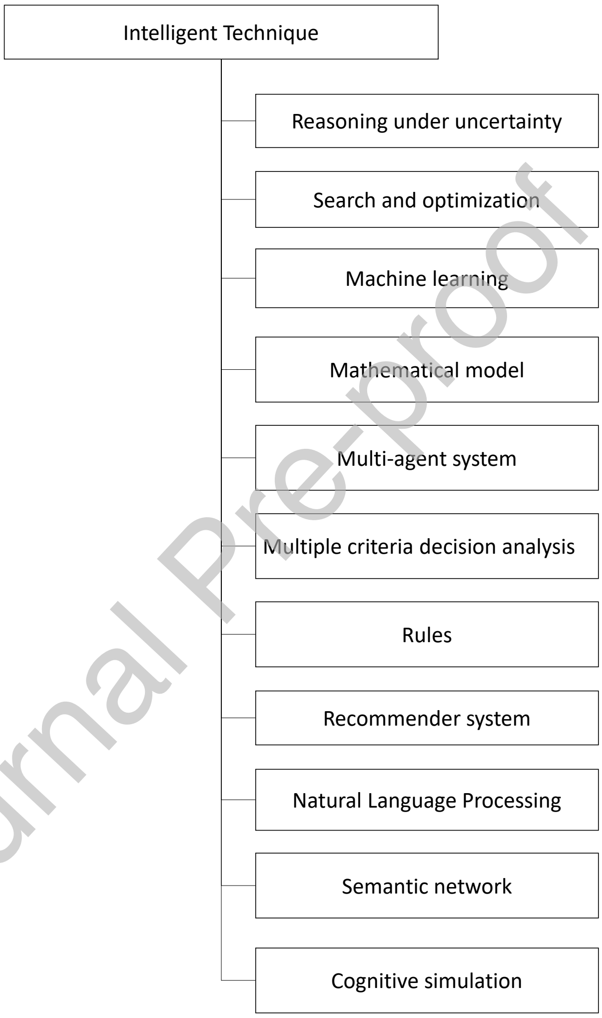


Figure 2: Intelligent techniques classification scheme.

code that will be deployed". The *product* level indicates that the intelligent technique directly affects the source code. In contrast, the *runtime* level represents intelligent techniques that "affect the deployed software during runtime" (e.g., autonomous and self-adaptive software systems). Regarding the *level of automation*, we classified each study using a scale of an integer between one and ten, as shown in Table8.

3. Results and Analysis

This section presents the results and analysis of the systematic review. In Section 3.1, we provide an overview of the results of the systematic review process. Section 3.2 presents the results of the quality assessment. In Section 3.3 we present the types of research contributions provided by studies related to ISE and ASD (RQ 1.1). Section 3.4 discusses the intelligent techniques applied to ASD (RQ 1.2). In Section 3.5, we discuss the purposes for which intelligent techniques were applied to ASD, presenting details of the most relevant studies (RQ 1.3), and in Section 3.6, we present a timeline of publications and trends (RQ 1.4). Section 3.7 discusses the maturity of the existing solutions applying intelligent techniques to ASD (RQ 2). Finally, in Section 3.8, we discuss the risks of adopting current intelligent techniques for ASD (RQ 3).

3.1. Overview of the systematic review process

In this section, we present the results of the systematic review process. In Figure 3, we present the results related to the number of studies evaluated and selected, given the different stages of the study. As shown in Figure 3, there were four iterations of snowballing. We queried the databases during the period between 5/17/2016 and 5/19/2016. The snowballing queries were executed, respectively, in the periods between: 8/29/2016 and 9/7/2016, 2/21/2017 and 3/2/2017, 4/4/2017 and 4/9/2017, and 4/16/2017 and 4/17/2017. As noticed, 24% of the selected papers were found during the snowballing phase, which shows the relevance of using snowballing to complement database search. Out

Table 8: Levels of automation according to Feldt et al. [30].

Level of automation	**Description**
10	Computer makes and implements decision if it feels it should, and informs human only if it feels this is warranted.
9	Computer makes and implements decision, and informs human only if it feels this is warranted.
8	Computer makes and implements decision, and informs human only if asked to.
7	Computer makes and implements decision, but must inform human after the fact.
6	Computer makes decision but gives human option to veto before implementation.
5	Computer offers a restricted set of alternatives and suggests one, which it will implement if human approve.
4	Computer offers a restricted set of alternatives and suggests one, but human still makes and implements final decision.
3	Computer offers a restricted set of alternatives, and human decides which to implement.
2	Computer offers a set of alternatives which human may ignore in making decision.
1	Human considers alternatives, makes and implements decision.

of the 104 selected studies, 65 (i.e., 62.5%) are papers from 51 conferences and 39 (i.e., 37.5%) articles from 32 journals. The conferences with most papers are The International Conference on Software Engineering & Knowledge Engineering (SEKE), with 5 papers; The Hawaii International Conference on System Sciences, with 3 papers; and, EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), International Conference on Product Focused Software Process Improvement (PROFES) and International Conference on Software and Data Technologies (ICSOFT), with 2 papers each. The most popular journals are Requirements Engineering, Information and Software Technology and The International Journal of Software Engineering and Its Applications, with 3 articles each; and, The Journal of Systems and Software, with 2 articles.

Given that some researchers publish the same study in multiple papers, to avoid bias on the analysis of the results, except for the quality assessment, the report is based on studies, not papers. For instance, we grouped Perkusich et al. [47] and Perkusich et al. [35] into one study. At the end of the grouping process, we had 92 unique studies.

3.2. Quality assessment

This section presents the data collected regarding the quality of the studies. In Figure 4, we show the aggregated results of the quality assessment for all papers, in which the maximum score for each quality criterion is 104 (i.e., the number of selected papers). By analyzing Figure 4, we can conclude that the overall quality of the studies is between low and medium (below 52). This result is a consequence of the lack of proper empirical studies to evaluate some of the proposed intelligent techniques, in light of the criteria presented by Dybå and Dingsøyr [61].

An extenuatory for our conclusion is the fact that 62.5% of the included papers are from conferences. Usually, they have a length constraint that might hinder the researchers from including important methodological information into the papers, negatively influencing their quality score.

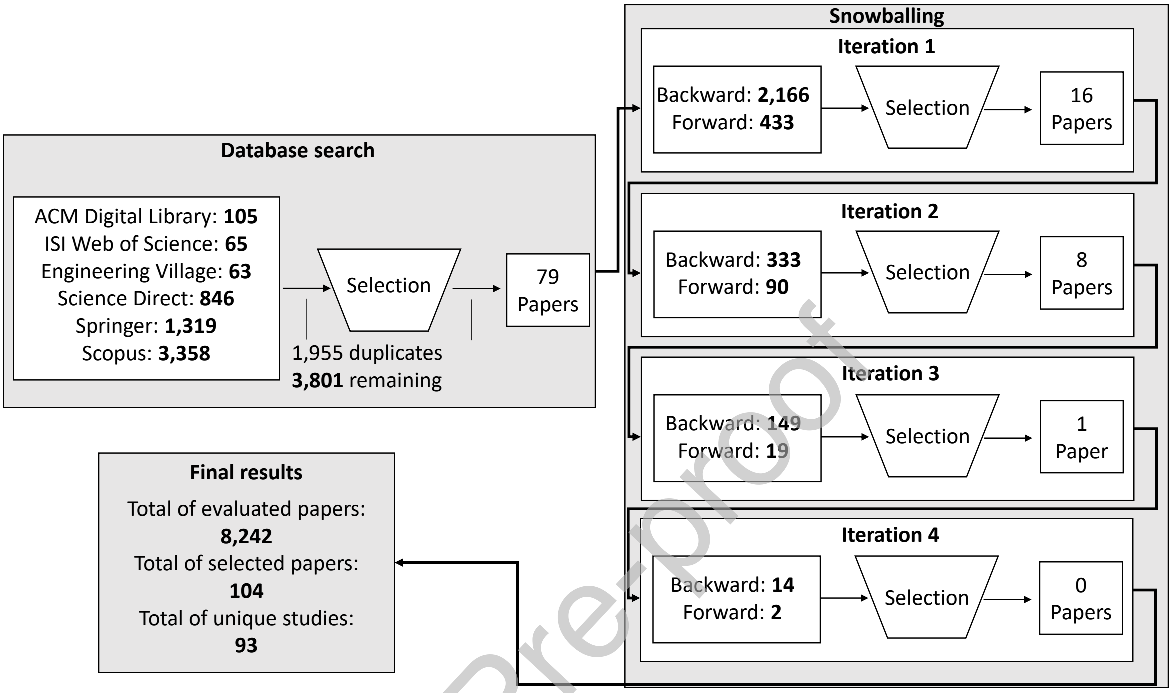


Figure 3: Number of papers in study selection.

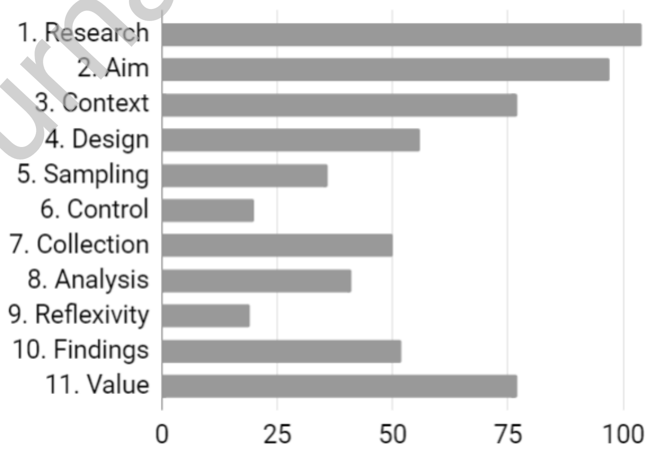


Figure 4: Quality scores of the selected studies.

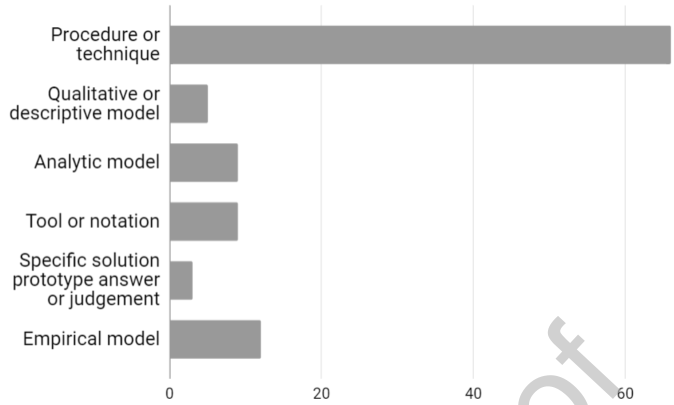


Figure 5: Publication distribution by contribution type.

3.3. Types of research contributions

This section discusses the types of research contributions provided by studies related to ISE and ASD (RQ 1.1). The primary studies were classified according to the type of research contributions, as defined in Table 5. Figure 5 shows the distribution of the research contributions. Most of the studies presented a *procedure or technique* (63.5%), followed by an *empirical model* (11.5%), *tool or notation* and *analytic model* (8.7% each), *qualitative or descriptive model* (4.8%), and a *specific solution prototype or judgment* (2.9%). The contributions include a framework to validate requirements [38], an empirical model to assist on release plan scheduling [67] and an empirical model to measure agility [68].

3.4. Intelligent techniques applied to ASD

This section discusses the identified intelligent techniques applied to ASD (RQ 1.2). The classification of the 92 studies revealed 7 main types of intelligent techniques (see Figure 6). The most popular intelligent techniques are *Reasoning under uncertainty*, *Search and optimization*, and *Machine learning* which were applied in 24.2%, 20.2% and 19.1% of the studies, respectively. The popularity of these techniques confirms the findings on Harman [26], which stated that these three techniques were the main types of artificial intelligence techniques

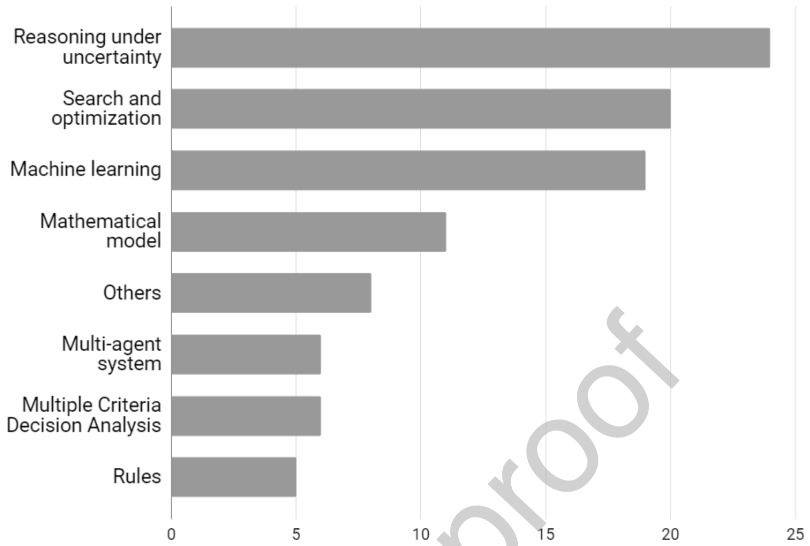


Figure 6: Publication distribution by intelligent technique.

applied for software engineering. In contrast, *Cognitive simulation* was only applied to ASD once and *Semantic network* and *Natural Language Processing*, twice.

The most popular *Reasoning under uncertainty* technique is *Bayesian networks*: it was present in 12 out of the 24 studies that applied this type of intelligent technique, followed by *fuzzy systems*, which was applied in 9. In Table 9, we show the *Reasoning under uncertainty* techniques identified and the corresponding papers.

The popularity of *Search and optimization* follows the trends showed in Harman et al. [8], in which it was shown the recent growth of SBSE research. In Table 10, we show the *Search and optimization* techniques identified and the corresponding papers.

Regarding machine learning, 68.7% of the studies used supervised learning techniques such as Regression analysis, Neural networks and Support Vector Machine; 25.0%, clustering such as K-means algorithm; and 6.3%, reinforcement

Table 9: Reasoning under uncertainty techniques identified.

Technique	Papers
Bayesian network	[65], [35], [48], [69], [51], [34], [70], [71], [72], [52], [36]
Fuzzy logic	[68], [73], [74], [75], [76], [77], [78], [79], [80]
Weighted dependency graph	[44]
Monte Carlo	[81]
Statistical model	[82]

Table 10: Search and optimization techniques identified.

Technique	Papers
Genetic algorithm	[83], [76], [84], [40]
Integer linear programming	[67]
Linear programming	[85], [86], [87], [88]
Hill climbing	[84]
Greedy algorithm	[84]
Bin-packing-related heuristic	[89]
Knapsack problem-related heuristic	[90], [91], [92], [66], [93]
Binary integer programming	[94]
Artificial bee colony	[95]
Ant colony	[41]
Constructive heuristic	[96]
Harmony search	[97]

Table 11: Machine learning techniques identified.

Technique	Papers
Plausible Justification tree	[98]
Regression analysis	[99], [100], [101], [102], [103], [104]
Neural networks	[100], [50], [105], [103]
Support Vector Machine	[100], [106]
K-means algorithm	[38]
Not specified clustering algorithm	[101], [107], [78], [108]
Not specified	[109], [110]

learning. In Table 11, we show the *Machine learning* techniques identified and the corresponding papers.

3.5. Applications of intelligent techniques on ASD

This section discusses the purposes for which intelligent techniques were applied to ASD, presenting details of the most relevant studies (RQ 1.3). The classification of the 92 studies revealed that they focused on 6 SWEBOK knowledge areas (see Figure 7). The most popular SWEBOK knowledge is *Software engineering management*, which is expected since ASD is a paradigm that greatly impacts the management of software projects, with 60.8%. It is followed by *Software requirements* (17.6%), *Software engineering process* (6.9%), *Software design* (6.9%), *Software quality* (5.9%) and *Software testing* (2.0%).

In Figure 8, we show the frequency of combination between intelligent techniques and SWEBOK knowledge areas. In what follows, we present details of the studies related to the most frequent SWEBOK knowledge areas. In Appendix 6, we present the complete list of papers classified given the intelligent technique and purpose.

3.5.1. Software engineering management

Software engineering management can be defined as the application of management activities planning, coordinating, measuring, monitoring, controlling, and reporting to ensure that software products and software engineering services are delivered efficiently, effectively, and to the benefit of stakeholders [57]. Given the expressiveness of studies on *software engineering management*, we further refined the classification of these papers given the knowledge areas from the Project Management Body of Knowledge (PMBoK) [111]. The most popular PMBoK knowledge areas identified was *Time* (35.5%), followed by *Human Resources* (17.7%), *Scope* (16.1%), *Integration* (16.1%), *Quality* (8.1%), *Risk* (3.2%) and *Stakeholders* (1.6%). In what follows, we present details of the studies related to the most frequent PMBoK knowledge areas, namely, *Time*, *Human Resource*, *Scope* and *Integration*.

Regarding studies applied to *Time*, 19 studies presented solutions to improve the estimation of effort, cost, or duration of tasks. These studies include approaches based on SBSE (e.g., [88]), machine learning [100] and probabilistic models [112]. A recent systematic review [113] presents the state-of-the-art of effort estimation in ASD, which includes the application of intelligent techniques in this context. The other 3 studies proposed solutions to improve the scheduling of the release plan and are based on SBSE ([97], [114] and [86]).

All the studies related to *Human Resources* presents solutions to assist in allocating software engineers to teams, projects or tasks (e.g., [115] and [42]), except for [116], which focuses on assessing the capability of an individual team member. The most frequent intelligent technique was SBSE, which was used on four solutions.

The studies related to *Scope* mostly focus on selecting requirements for a release or sprint (i.e., iteration). The most popular technique used for this purpose was SBSE techniques such as ant-colony optimization [41] and nested knapsack heuristic [117].

Regarding *Integration*, the most frequent intelligent technique was *Proba-*

bilistic modeling for uncertain reasoning, which is used to support project management by modeling anti-patterns [118] and for management (e.g., [71]). A popular goal is to support process tailoring (e.g., [75] and [78]), which, according to the SWEBOK, should be the first step in software project planning.

3.5.2. Software requirements

The software requirements knowledge area is concerned with elicitation, analysis, specification, and validation of software requirements [57]. The most popular intelligent techniques identified in this context are SBSE and Machine learning. SBSE has been used to support the prioritization of software requirements (e.g., [67] and [95]), and Machine learning to support the validation [38], specification [119] and elicitation of software requirements [98].

3.5.3. Software engineering process

Software engineering processes are concerned with work activities accomplished by software engineers to develop, maintain, and operate software, such as requirements, design, construction, testing, configuration management, and other software engineering processes [57]. In this context, most of the intelligent techniques have been applied to assess the process quality or assessment. The most popular intelligent technique used is Probabilistic modeling of uncertain reasoning, which has been used in [35] and [48] to evaluate the process quality and in [68] to measure the process agility.

3.5.4. Software design

Software design is the activity in which software requirements are analyzed to produce a description of the software's internal structure that will serve as the basis for its construction [57]. In this context, the most popular intelligent techniques identified were Rules (3 studies) and Machine learning (2 studies). The identified studies had a variety of aims such as dependencies identification [120], flexibility [121], Human Computer Interface design [122] and support reuse [123].

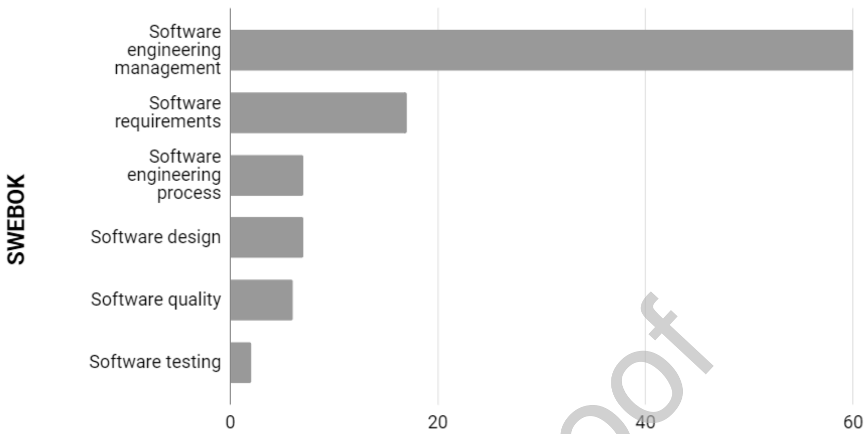


Figure 7: Publication distribution by SWEBOK knowledge areas.

3.6. Publication frequency

This section discusses the publication frequency and topics (RQ 2). In Figures 9 and 10, we show the number of selected papers per year classified by their purpose and the applied intelligent technique, respectively.

The trend of publications is positive up to 2015, with a few exceptions, showing that there is an increased interest in applying intelligent techniques to ASD. We have only found 2 relevant studies in 2016 and 1 in 2017, but these data cannot be used to interpret the trend in the research field. Since we executed the database search in May 2016, we likely missed relevant studies from 2016 and 2017. Therefore, we removed the results of 2016 and 2017 from Figures 9 and 10 to avoid misunderstandings. We could have limited the scope of the study to papers up to 2015. Still, since they are relevant to our research questions, we agreed that it would be more valuable to include them.

By analyzing Figure 9, we observe that applying intelligent techniques to software engineering management has been the most popular area of application, with an almost constant factor starting from 2009. Software requirements papers have appeared in most of the years, with more emphasis after 2013. For the

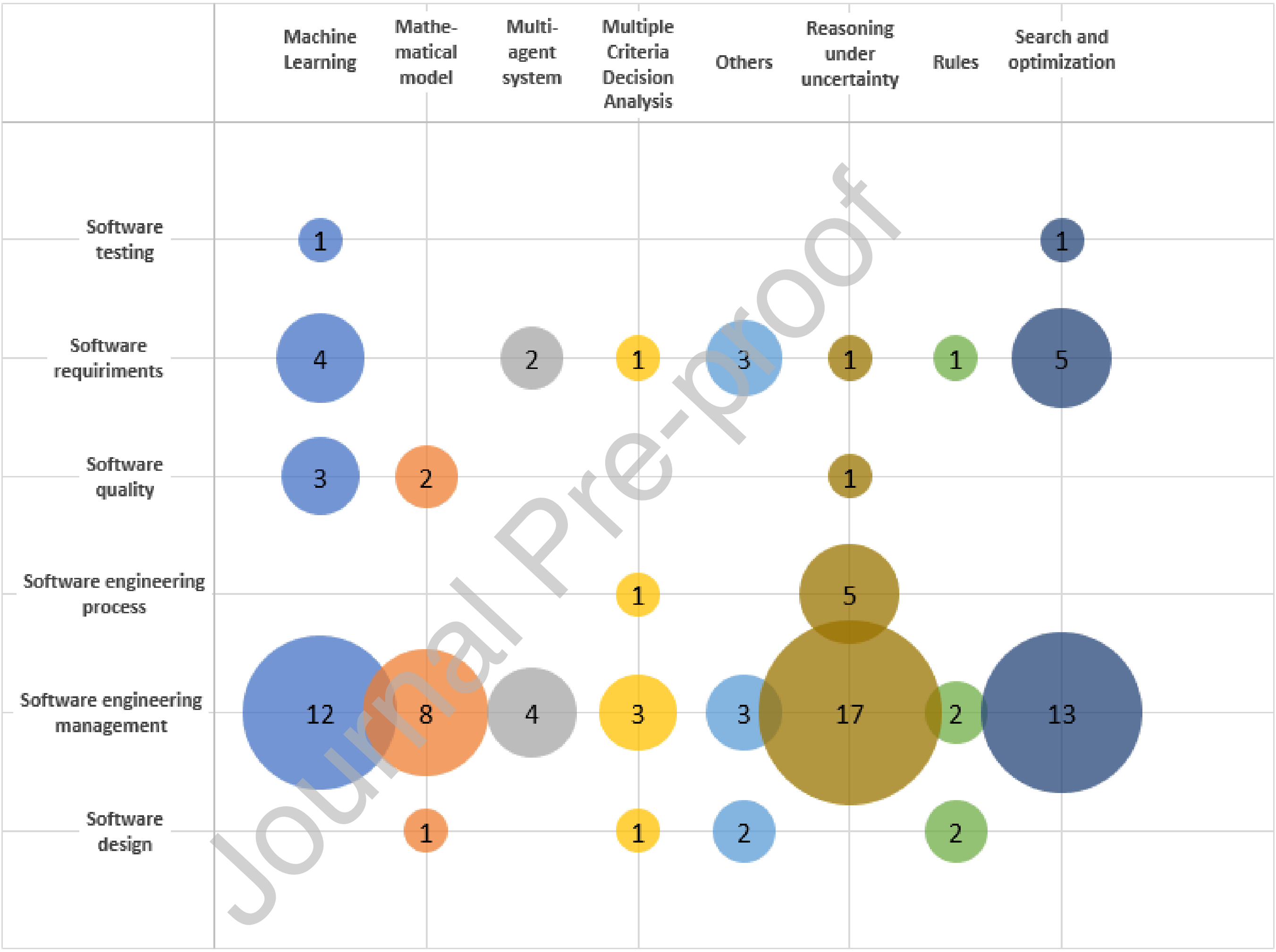


Figure 8: Frequency of combination between intelligent techniques and SWEBOK knowledge areas.

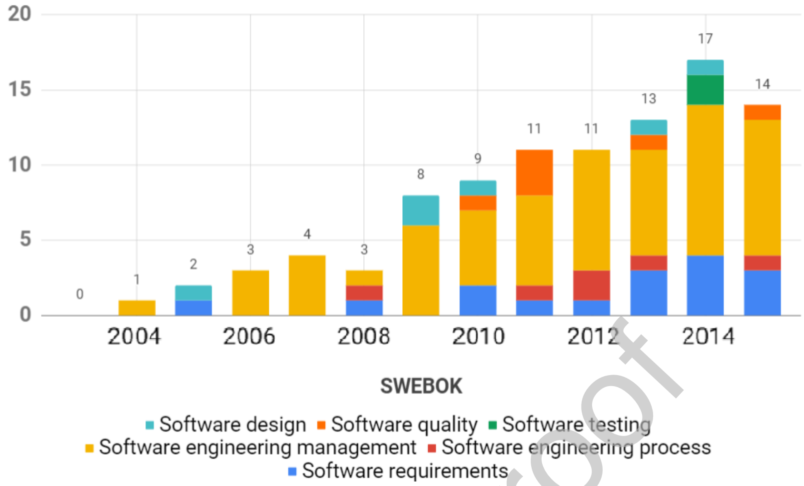


Figure 9: Cumulative publications per year classified by their purpose.

other areas, there is no trend to be identified, because researchers have been focusing effort on several areas. The results were expected because ASD has a high correlation to management techniques, and as discussed in Section 2.3, we only included studies that explicitly stated to focus on ASD.

By analyzing Figure 10, the most consistently used techniques are Machine Learning, Search and optimization, Multiple Criteria Decision Analysis and Probabilistic methods for uncertain reasoning, showing up on, respectively, 75%, 67%, 50% and 42% of the years. On the other hand, it is not possible to identify that the usage of any of the techniques has had a significant increase throughout the years.

3.7. Maturity of current intelligent techniques applied to ASD

This section discusses RQ2, in which we assess the maturity of the existing solutions applying intelligent techniques to ASD. To evaluate the maturity of the current solutions, we rely on two main factors: the impact of the contributions for practitioners and their reproducibility (i.e., the ease for another researcher to reproduce the study). For this purpose, we collected four types of data:

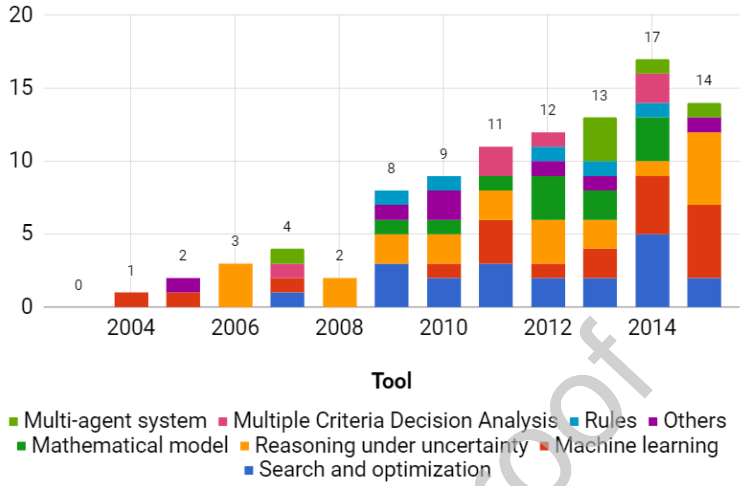


Figure 10: Cumulative publications per year classified by the applied intelligent technique.

empirical research type, research validation, availability of tools, and availability of datasets.

Regarding the empirical research type, 27.9% of the studies used a case study to evaluate their solutions, 24.0% used an experiment, 4.8% used an observational study, and the remaining papers did not perform any empirical study. Regarding the research validation strategy, 23.1% of the studies used an evaluation, 20.2%, used an analysis, and the remaining papers relied on non-empirical strategies including blatant assertion (20.2%), example (20.2%), and persuasion (10.6%). The data is in conformance to our expectations considering the quality scores presented in Figure 4.

Regarding the availability of tools, only four of the studies make the proposed solutions available as tools: XPPLAN[5] [93], SCRUMPROJECTMANAGEMENT[6] [124], ROCLET[7] [125], and AQUSA[8] [119]. The remaining studies

[5]https://github.com/gertvv/xpplan
[6]http://multiagent.fr/ScrumProjectManagement
[7]http://www.roclet.org/index.php
[8]https://github.com/gglucass/AQUSA

present pseudocode, model, or formulas associated with the proposed solutions (77 studies) and a few (11 studies) present no details regarding their solution.

Regarding the availability of the dataset used to train models or validate solutions, 14 studies make the data available by citing an open-source project [65], releasing the data in a specific repository [104], or documenting the data in the paper itself. 12 studies make only part of the data available. For instance, in [67], the authors use two sets of data to validate their solution: the small and master datasets. They present the *small dataset* in a table, but not the *master dataset*, which they collected from a company. The remaining studies or did not use data to train their models or validate their study or do not make them available.

The impact of the contributions for practitioners is meager because only four of the proposed solutions were made available as tools. Presenting the pseudocode, models, or mathematical formula can help a practitioner to implement the proposed solution, but it adds the cost and risks of implementation. Another obstacle that was only mentioned by van Valkenhoef et al. [117] is the challenge to integrate the developed solutions into tools already used by the practitioners. In their case, they have evidence that their planning model reduced the time and effort required in release planning, but entering the data from user stories (stored in the practitioner's project management tool) into their model was a challenge. Despite this, the XPPLAN and AQUSA are promising tools, and we encourage practitioners to adopt them since they are available freely, and they were empirically analyzed with positive results.

The reproducibility of the studies is hindered by the lack of sharing of the tools and datasets. A case in which the challenge of reproducibility is evidenced is in Chaves-Gonzlez et al. [95]. The authors compared the results of their multiobjective swarm intelligence evolutionary algorithm to optimize software requirements with the solution presented in del Sagrado [41]. Even though the dataset is made available by del Sagrado [41], the tool is not, which hindered Chaves-Gonzlez et al. [95] from performing an analysis of the efficiency of the algorithms in the same context. As a consequence, it is hard for researchers

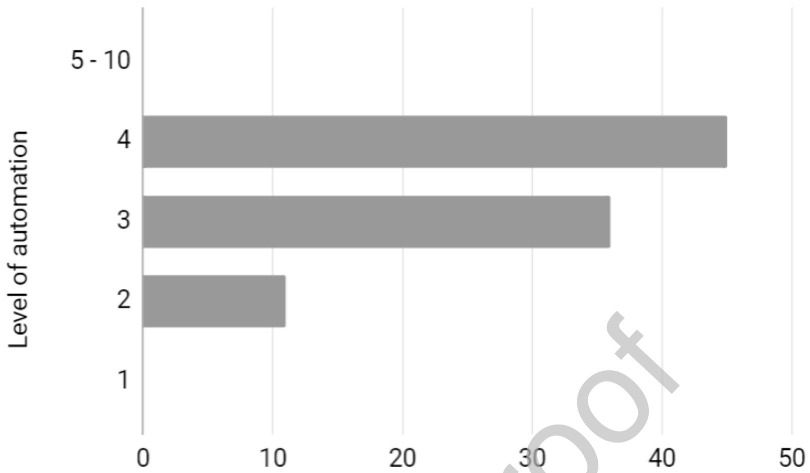


Figure 11: Frequency of levels of automation.

to conduct comparative empirical studies, which is the ideal development of a scientific field [126].

3.8. Risks on applying intelligent techniques to ASD

This section discusses RQ3, in which we explore the risks, benefits, and obstacles in applying intelligent techniques to ASD based on the selected studies. To assess the risks of the identified proposed solutions, we relied on the factors proposed by Feldt et al. [30] and presented in Section 2.5 (i.e., points of application, levels of automation, and intelligent technique). Regarding the points of application, all the identified studies are of the type *process*. We are confident that this follows directly from a limitation of the scope of our study (i.e., including only studies that explicitly focused on ASD), and does not reflect the current solutions that have the potential to be applied to ASD. Despite this, we only focus our analysis on the studies that we have identified during the systematic review.

Figure 11 shows the frequency of studies per level of automation. As shown in Figure 11, all studies have a level of automation of up to four, which are related to supporting decision-making. According to Feldt et al. [30], these are indicators that the current approaches on applying intelligent techniques to ASD have low risk. Their main argument is that, since the machine does not directly modify the software, the human (e.g., a project manager) has the opportunity to review the machine's output and decide to implement it or not.

Regarding the type of intelligent technique, a significant property is its interpretability. For instance, Abrahamsson et al. [103] and Dragicevic et al. [36] present data-based solutions for predicting development effort. In Abrahamsson et al. [103], a neural network was used for this purpose, while Dragicevic et al. [36] constructed a Bayesian network. In general, an advantage of using Bayesian networks over neural networks is because it is easier to interpret and understand the knowledge encoded in its representation, since the underlying graph has recognizable distinctions and causal relationships [127, 128, 129, 130]. Therefore, for this purpose, suppose that the effort estimation (in time) for a given feature, as interpreted from the models, is 6 days. If the manager is using the Bayesian network proposed by Dragicevic et al. [36], he can diagnose the reasons for the model to calculate 6 days for a given task and analyze the correctness given his knowledge (i.e., does the model consider all the relevant factors?) [131]. Given this, using a Bayesian network instead of a neural network, for this purpose, could reduce the risk of using tasks with incorrect effort estimations for project planning.

Another perspective to analyze the risk of using an intelligent technique is from the viewpoint of its acceptability by the software engineering practitioners, which significantly influences its adoption. According to Minku et al. [132], the lack of involvement of humans in developing the solutions and their transparency might hinder its adoption. For instance, in the case of Dragicevic et al. [36], the structure of the Bayesian network was constructed with the assistance of humans, and they only relied on data for building the node probability tables. Given this and the properties of a Bayesian network, according to the criteria presented by Minku et al. [132], we can conclude that a Bayesian network's chance of being adopted is higher than a neural network for a complex and important task such as effort estimation, which impacts the project's schedule and sets the client's expectations.

One of the main findings of our study is that, given our classification scheme (see Figure 2), all the types of the identified intelligent techniques in this study are easy to understand because they are based on simple rules and humans can verify their reasoning process, except for, potentially, *Machine learning*, which appeared in nineteen studies. For instance, this is the case for *Search and optimization*, *Rules*, and *Mathematical models*, which are mostly based on simple rules [30], and for *Reasoning under uncertainty*, represented by Bayesian networks and fuzzy systems, in which the reasoning is readable by humans. Regarding *Machine learning*, there are also explainable techniques, such as *Case-based reasoning* [133], which appeared in three studies, in which the user can check the similar cases that explain a given output. Furthermore, it is valuable to notice that there are currently many initiatives on building explainable deep neural network techniques [134, 135, 136, 137].

4. Implications for research and practice

This systematic review has several implications for research and practice. For research, even though there are high-quality studies for several purposes ranging from effort estimation to requirements selection and process management, the review shows a clear need for more empirical studies on applying intelligent techniques to ASD, given that less than 50% of the studies relied on empirical studies to evaluate their proposed solutions. In our opinion, there are clear potential benefits of applying intelligent techniques to ASD, but there is a need for more studies exploring the factors that affect the adoption of the tools by practitioners, as discussed in Section 3.8 which would support researchers in determining the most appropriate technique to tackle a problem. Additionally, the study shows that intelligent techniques have been proposed to support decision-making in ASD. Understanding the factors that affect the adoption of intelligent tools would help researchers to formulate processes for the practitioners to use the tool for decision-support purposes.

This review shows that there are problems that have been frequently studied, such as effort estimation, human resources allocation, and requirements selection. It is not part of our scope to tackle and discuss each problem itself, which we believe deserves a dedicated study (e.g., see [113] for effort estimation in ASD), but the evidence shows that there is a lack of comparison between the proposed solutions. For instance, a popular aim of the selected studies is to estimate effort (duration or cost) of tasks, with a total of 19 studies. Several different intelligent techniques were used, such as SBSE, machine learning, and *Bayesian networks*. In 2011, [100] presented an approach to estimate effort in agile environments. Later, other studies present similar studies, but they do not compare their solutions with the one presented in [100].

As discussed in Section 3.7, this might directly follow from the lack of availability of datasets and tools by the researchers. Therefore, we believe that researchers should share their data and tools. The sharing of tools leads to the open science initiative. Recently, data science [138], machine learning, big data, and advanced analytics have emerged as data-oriented approaches. Also, smart-data, a method that supports data engineering and knowledge engineering to the process of model development [139] has been recently proposed.

Furthermore, transfer learning [140, 141, 142, 143] techniques have been developed to reuse data and apply it to different, but related problem (e.g., use data from collected from one company to build models for other companies). All these approaches rely on data to build more accurate models. Therefore, if researchers share their collected data, better models can be constructed. Furthermore, sharing data leads to more understanding of the developed research.

Even though there are studies in which promising solutions have been proposed (see Section 3.5), we believe that the lack of availability of tools (only four studies made their solutions available as tools), hinders the usefulness of current research in the ISE field to ASD practitioners.

To increase the usefulness of the research to industry, we think that there is a need for researchers in the field to collaborate to determine a common research agenda. It is beyond the scope of this article to suggest such an agenda, but we believe that there is a need for researchers from different communities such as machine learning, search and optimization, reasoning under uncertainty, knowledge management, data analytics, software metrics, and decision-support systems to define the basis from which the field can evolve. Hopefully, the synthesis presented herein may provide an initial inspiration to create one.

For practitioners, this review shows that many promising studies applying intelligent techniques to ASD have been reported. Our study can be a starting point for them to analyze the available tools and kick-start an adoption process by reusing available tools, partnering with researchers with promising results, but have not published a tool, or even to implement ideas (e.g., algorithms) presented in the studies that they see having utility in their context.

5. Threats to validity

Wohlin et al. [144] suggest a classification schema that distinguish four validity aspects of an empirical study: *construct validity*, *internal validity*, *external validity*, and *conclusion validity*. In what follows, we present the threats to validity of this study following the classification schema proposed by Wohlin et al. [144].

- *Construct validity*: the classification of the intelligent techniques followed a thematic analysis approach [64], in which each paper had its intelligent technique extracted by one researcher, and checked by a second one. One researcher classified the collected data into themes, and, collaboratively, there the themes were defined until a consensus was reached. Despite this, there remains a risk that the classification is not representative of the area due to subjective bias. Additionally, given the broadness of the topic area of this systematic review, we used the general-purpose criteria defined in Dybå and Dingsøyr [61], which is used by several researchers [145]. Finally, we did not exclude from our study, primary studies with low quality, because our goal was to have an overview of the research area from a broad perspective. More than 62.5% of the included studies are from conferences, which usually have a constraint regarding the length of the paper. For instance, SEKE restricts papers to six pages, which hinders the researchers from including important methodological information to evidence the quality of the study. Following this, a direct threat is that our conclusions related to the need for more empirical studies would be heavily influenced if we had discarded the low-quality studies or conference papers. Despite this, our conclusions regarding the contribution to practice and reproducibility would not.

- *Internal validity*: we used a mixed-methods approach as our search strategy, including a database search followed by snowballing. Our approach minimized the threat of missing relevant studies during our search for primary papers. To avoid the authors' bias regarding the inclusion of papers, we only included papers in which an intelligent technique was explicitly applied to ASD. As a consequence, this limited the scope of our study, because there are potentially many solutions that could be applied to ASD that were not included such as related to refactoring, automatic defect correction, and self-adaptive systems.

- *External validity*: we report the protocol used to execute this systematic review following the guidelines [45], so that this study can be replicated or extended by other researchers.

- *Conclusion validity*: we operationalized the protocol using spreadsheets, including all the information necessary to select the primary papers and extract data. One researcher managed the study selection process by randomly allocating a set of primary papers to the reviewers, and each of them performed the selection independently. This strategy reduced the risk of researcher bias. For the data extraction, all the papers were checked by two reviewers.

6. Conclusion and Future Work

This article has provided a survey and review of the area of ISE in the context of ASD. We identified 104 unique studies, which we mapped into 92 unique studies. Less than 50% of the studies performed an empirical approach to evaluate the proposed solutions.

The key findings of the systematic review indicate that (i) there is an increase in the number of studies applying intelligent techniques to ASD, (ii) reasoning under uncertainty, search-based solutions, and machine learning are the most used intelligent techniques in the context of ASD; (iii) the predominant purposes are software engineering management, more specifically, effort estimation, requirements prioritization, resource allocation, and requirements selection for a release or sprint; and requirements management; (iv) the adoption risks in terms of safety of the current solutions are low, and (v) there is a need for an increase in the validation and evaluation methods for the proposed solutions.

References

[1] E. M. Clarke, J. M. Wing, Formal methods: State of the art and future directions, ACM Computing Surveys (CSUR) 28 (4) (1996) 626–643.

[2] I. Jacobson, M. Griss, P. Jonsson, Software reuse: architecture process and organization for business success, Vol. 285, acm Press New York, 1997.

[3] B. Boehm, Value-based software engineering, ACM SIGSOFT Software Engineering Notes 28 (2) (2003) 4.

[4] T. Xie, Intelligent software engineering: Synergy between ai and software engineering, in: Proceedings of the 11th Innovations in Software Engineering Conference, ACM, 2018, p. 1.

[5] T. Xie, Intelligent software engineering: Synergy between ai and software engineering, in: X. Feng, M. Müller-Olm, Z. Yang (Eds.), Dependable Software Engineering. Theories, Tools, and Applications, Springer International Publishing, Cham, 2018, pp. 3–7.

[6] H. Zhang, B. Kitchenham, D. Pfahl, Software process simulation modeling: an extended systematic review, in: International Conference on Software Process, Springer, 2010, pp. 309–320.

[7] B. Selic, The pragmatics of model-driven development, IEEE software 20 (5) (2003) 19–25.

[8] M. Harman, S. A. Mansouri, Y. Zhang, Search-based software engineering: Trends, techniques and applications, ACM Comput. Surv. 45 (1) (2012) 11:1–11:61. doi:10.1145/2379776.2379787.
URL http://doi.acm.org/10.1145/2379776.2379787

[9] D. Zhang, J. J. Tsai, Machine learning and software engineering, Software Quality Journal 11 (2) (2003) 87–119. doi:10.1023/A:1023760326768.
URL https://doi.org/10.1023/A:1023760326768

[10] M. Gasparic, A. Janes, What recommendation systems for software engineering recommend: A systematic literature review, Journal of Systems and Software 113 (Supplement C) (2016) 101 – 113. doi:https://doi.org/10.1016/j.jss.2015.11.036.
URL http://www.sciencedirect.com/science/article/pii/S0164121215002605

[11] A. T. Misirli, A. B. Bener, Bayesian networks for evidence-based decision-making in software engineering, IEEE Transactions on Software Engineering 40 (6) (2014) 533–554. doi:10.1109/TSE.2014.2321179.

[12] T. Menzies, T. Zimmermann, Software analytics: so what?, IEEE Software (4) (2013) 31–37.

[13] M. Allamanis, E. T. Barr, P. Devanbu, C. Sutton, A survey of machine learning for big code and naturalness, ACM Computing Surveys (CSUR) 51 (4) (2018) 81.

[14] E. Colombo, C. Francalanci, Selecting crm packages based on architectural, functional, and cost requirements: Empirical validation of a hierarchical ranking model, Requirements engineering 9 (3) (2004) 186–203.

[15] L. Zhu, A. Aurum, I. Gorton, R. Jeffery, Tradeoff and sensitivity analysis in software architecture evaluation using analytic hierarchy process, Software Quality Journal 13 (4) (2005) 357–375.

[16] J. Mostow, Foreword what is ai? and what does it have to do with software engineering?, IEEE Transactions on Software Engineering (11) (1985) 1253–1256.

[17] D. Partridge, Artificial intelligence: applications in the future of software engineering, Halsted Press, 1986.

[18] L. Ford, Artificial intelligence and software engineering: a tutorial introduction to their relationship, Artificial intelligence review 1 (4) (1987) 255–273.

[19] D. Partridge, Artificial intelligence and software engineering: a survey of possibilities, Information and Software Technology 30 (3) (1988) 146–152.

[20] C. Bowerman, S. Stobart, An intelligent software-engineering environment: Ise, in: Systems Engineering of Computer Based Systems, 1995., Proceedings of the 1995 International Symposium and Workshop on, IEEE, 1995, pp. 429–433.

[21] S. Liu, H. Asaminami-ku, Formal methods and intelligent software engineering environments, Information: An International Journal 1 (1995) 83–102.

[22] F. Meziane, S. Vadera, Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects: New Prospects, IGI Global, 2009.

[23] F. N. Raza, Artificial intelligence techniques in software engineering (aitse), in: International MultiConference of Engineers and Computer Scientists (IMECS 2009), Vol. 1, 2009.

[24] P. Jain, Interaction between software engineering and artificial intelligence-a review, International Journal on Computer Science and Engineering 3 (12) (2011) 3774.

[25] H. H. Ammar, W. Abdelmoez, M. S. Hamdi, Software engineering using artificial intelligence techniques: Current state and open problems, in: Proceedings of the First Taibah University International Conference on Computing and Information Technology (ICCIT 2012), Al-Madinah Al-Munawwarah, Saudi Arabia, 2012, p. 52.

[26] M. Harman, The role of artificial intelligence in software engineering, in: Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering, IEEE Press, 2012, pp. 1–6.

[27] K. H. Shankari, D. R. Thirumalaiselvi, A survey on using artificial intelligence techniques in the software development process, IJERA, ISSN (2014) 2248–9622.

[28] C. Rich, R. C. Waters, Readings in artificial intelligence and software engineering, Morgan Kaufmann, 1986.

[29] B. W. Sorte, P. Joshi, V. Jagtap, Use of artificial intelligence in software development life cycle: A state of the art review, International Journal of Advanced Engineering and Global Technology (2015) 398–403.

[30] R. Feldt, F. G. Neto, R. Torkar, Ways of applying artificial intelligence in software engineering, 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (2018) 16–19.

[31] R. Hoda, N. Salleh, J. Grundy, The rise and evolution of agile software development, IEEE Software 35 (5) (2018) 58–63.

[32] R. Hoda, N. Salleh, J. Grundy, H. M. Tee, Systematic literature reviews in agile software development: A tertiary study, Information and Software Technology 85 (2017) 60–70.

[33] B. Ramesh, L. Cao, R. Baskerville, Agile requirements engineering practices and challenges: an empirical study, Information Systems Journal 20 (5) (2010) 449–480.

[34] P. Hearty, N. Fenton, D. Marquez, M. Neil, Predicting project velocity in xp using a learning dynamic bayesian network model, IEEE Transactions on Software Engineering 35 (1) (2009) 124–137. doi:10.1109/TSE.2008.76.

[35] M. Perkusich, G. Soares, H. Almeida, A. Perkusich, A procedure to detect problems of processes in software development projects using bayesian networks, Expert Systems with Applications 42 (1) (2015) 437 – 450. doi:https://doi.org/10.1016/j.eswa.2014.08.015.
URL http://www.sciencedirect.com/science/article/pii/S0957417414004916

[36] S. Dragicevic, S. Celar, M. Turic, Bayesian network model for task effort estimation in agile software development, Journal of Systems and Software 127 (Supplement C) (2017) 109 – 119. doi:https://doi.org/10.1016/j.jss.2017.01.027.
URL http://www.sciencedirect.com/science/article/pii/S0164121217300171

[37] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, G. Succi, Effort prediction in iterative software development processes – incremental versus global prediction models, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), 2007, pp. 344–353. doi:10.1109/ESEM.2007.16.

[38] P. Belsis, A. Koutoumanos, C. Sgouropoulou, Pburc: a patterns-based,

unsupervised requirements clustering framework for distributed agile software development, Requirements Engineering 19 (2) (2014) 213–225. doi:10.1007/s00766-013-0172-9.
URL https://doi.org/10.1007/s00766-013-0172-9

[39] R. Dwivedi, D. Gupta, Applying machine learning for configuring agile methods, International Journal of Software Engineering and Its Applications 9 (3) (2015) 29–40. doi:10.14257/ijseia.2015.9.3.04.

[40] A. Kumar, R. Nagar, A. S. Baghel, A genetic algorithm approach to release planning in agile environment, in: 2014 International Conference on Information Systems and Computer Networks (ISCON), 2014, pp. 118–122. doi:10.1109/ICISCON.2014.6965230.

[41] J. del Sagrado, I. M. del Águila, F. J. Orellana, Multi-objective ant colony optimization for requirements selection, Empirical Software Engineering 20 (3) (2015) 577–610. doi:10.1007/s10664-013-9287-3.
URL https://doi.org/10.1007/s10664-013-9287-3

[42] R. Shankarmani, S. S. Mantha, V. Babu, D. Mehta, K. Khatri, P. Kaushil, A decision support system utilizing a semantic agent, in: 2010 IEEE International Conference on Software Engineering and Service Sciences, 2010, pp. 442–447. doi:10.1109/ICSESS.2010.5552340.

[43] Anita, N. Chauhan, A regression test selection technique by optimizing user stories in an agile environment, in: 2014 IEEE International Advance Computing Conference (IACC), 2014, pp. 1454–1458. doi:10.1109/IAdCC.2014.6779540.

[44] T. Birkholzer, C. Dickmann, J. Vaupel, A framework for systematic evaluation of process improvement priorities, in: 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications, 2011, pp. 294–301. doi:10.1109/SEAA.2011.52.

[45] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Tech. Rep. EBSE-2007-01, Schoolf of Computer Science and Mathematics, Keele University (2007).

[46] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, ACM, New York, NY, USA, 2014, pp. 38:1–38:10. doi:10.1145/2601248.2601268.
URL http://doi.acm.org/10.1145/2601248.2601268

[47] M. Perkusich, H. O. de Almeida, A. Perkusich, A model to detect problems on scrum-based software development projects, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, ACM, New York, NY, USA, 2013, pp. 1037–1042. doi:10.1145/2480362.2480560.
URL http://doi.acm.org/10.1145/2480362.2480560

[48] A. Silva Freire, R. Da Silva, M. Perkusich, H. Almeida, A. Perkusich, A bayesian network model to assess agile teams' teamwork quality, in: Software Engineering (SBES), 2015 29th Brazilian Symposium on, 2015, pp. 191–196. doi:10.1109/SBES.2015.29.

[49] J. del Sagrado, I. M. del Aguila, F. J. Orellana, Ant colony optimization for the next release problem: A comparative study, in: 2nd International Symposium on Search Based Software Engineering, 2010, pp. 67–76. doi:10.1109/SSBSE.2010.18.

[50] A. Panda, S. M. Satapathy, S. K. Rath, Empirical validation of neural network models for agile software effort estimation based on story points, Procedia Computer Science 57 (Supplement C) (2015) 772 – 781, 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015). doi:https://doi.org/10.1016/j.procs.2015.07.474.
URL http://www.sciencedirect.com/science/article/pii/S1877050915020037

[51] A. Nagy, M. Njima, L. Mkrtchyan, A bayesian based method for agile software development release planning and project health monitoring, in: 2010 International Conference on Intelligent Networking and Collaborative Systems, 2010, pp. 192–199. doi:10.1109/INCOS.2010.99.

[52] M. Abouelela, L. Benedicenti, Bayesian network based xp process modelling, International Journal of Software Engineering and Applications 1 (3) (2010) 1–15.

[53] M. A. Boschetti, M. Golfarelli, S. Rizzi, E. Turricchia, A lagrangian heuristic for sprint planning in agile software development, Computers Operations Research 43 (Supplement C) (2014) 116 – 128. doi:https://doi.org/10.1016/j.cor.2013.09.007.
URL http://www.sciencedirect.com/science/article/pii/S0305054813002657

[54] S. Wagner, A. Goeb, L. Heinemann, M. Kls, C. Lampasona, K. Lochmann, A. Mayr, R. Plsch, A. Seidl, J. Streit, A. Trendowicz, Operationalised product quality models and assessment: The quamoco approach, Information and Software Technology 62 (Supplement C) (2015) 101 – 123. doi:https://doi.org/10.1016/j.infsof.2015.02.009.
URL http://www.sciencedirect.com/science/article/pii/S0950584915000452

[55] T. Mariani, S. R. Vergilio, A systematic review on search-based refactoring, Information and Software Technology 83 (2017) 14 – 34. doi:https://doi.org/10.1016/j.infsof.2016.11.009.
URL http://www.sciencedirect.com/science/article/pii/S0950584916303779

[56] M. OKeeffe, M. . Cinnide, Search-based refactoring for software maintenance, Journal of Systems and Software 81 (4) (2008) 502 – 516, selected papers from the 10th Conference on Software Maintenance and Reengineering (CSMR 2006).

doi:https://doi.org/10.1016/j.jss.2007.06.003.
URL http://www.sciencedirect.com/science/article/pii/S0164121207001409

[57] P. Bourque, R. E. Fairley, et al., Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0, IEEE Computer Society Press, 2014.

[58] N. B. Ali, K. Petersen, C. Wohlin, A systematic literature review on the industrial use of software process simulation, J. Syst. Softw. 97 (C) (2014) 65–85. doi:10.1016/j.jss.2014.06.059.
URL http://dx.doi.org/10.1016/j.jss.2014.06.059

[59] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain, Journal of Systems and Software 80 (4) (2007) 571 – 583. doi:http://dx.doi.org/10.1016/j.jss.2006.07.009.
URL //www.sciencedirect.com/science/article/pii/S016412120600197X

[60] M. Staples, M. Niazi, Experiences using systematic review guidelines, J. Syst. Softw. 80 (9) (2007) 1425–1437. doi:10.1016/j.jss.2006.09.046.
URL http://dx.doi.org/10.1016/j.jss.2006.09.046

[61] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Information and Software Technology 50 (910) (2008) 833 – 859. doi:http://dx.doi.org/10.1016/j.infsof.2008.01.006.
URL http://www.sciencedirect.com/science/article/pii/S0950584908000256

[62] M. Shaw, Writing good software engineering research papers, in: 25th International Conference on Software Engineering, 2003. Proceedings., 2003, pp. 726–736. doi:10.1109/ICSE.2003.1201262.

[63] P. Tonella, M. Torchiano, B. Du Bois, T. Systä, Empirical studies in reverse engineering: state of the art and future trends, Empirical Software Engineering 12 (5) (2007) 551–571. doi:10.1007/s10664-007-9037-5. URL http://dx.doi.org/10.1007/s10664-007-9037-5

[64] D. S. Cruzes, T. Dyba, Recommended steps for thematic synthesis in software engineering, in: 2011 International Symposium on Empirical Software Engineering and Measurement, IEEE, 2011, pp. 275–284.

[65] L. Li, H. Leung, Predicting fault-proneness of object-oriented system developed with agile process using learned bayesian network, in: 15th International Conference on Enterprise Information Systems (ICEIS 2013), 2013.

[66] F. Sobiech, B. Eilermann, A. Rausch, A heuristic approach to solve the elementary sprint optimization problem for non-cross-functional teams in scrum, SIGAPP Appl. Comput. Rev. 14 (4) (2015) 19–26. doi:10.1145/2724928.2724930.
URL http://doi.acm.org/10.1145/2724928.2724930

[67] C. Li, M. van den Akker, S. Brinkkemper, G. Diepen, An integrated approach for requirement selection and scheduling in software release planning, Requirements Engineering 15 (4) (2010) 375–396. doi:10.1007/s00766-010-0104-x.
URL https://doi.org/10.1007/s00766-010-0104-x

[68] P. K. Medappa, S. Bhattacharya, Towards a framework for assessing agility, in: 2012 45th Hawaii International Conference on System Sciences, 2012, pp. 5329–5338. doi:10.1109/HICSS.2012.599.

[69] D. Settas, S. Bibi, P. Sfetsos, I. Stamelos, V. Gerogiannis, Using bayesian belief networks to model software project management antipatterns, in: Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), IEEE, 2006, pp. 117–124.

[70] M. Perkusich, A. Medeiros, K. C. Gorgônio, H. O. de Almeida, A. Perkusich, et al., A bayesian network approach to assist on the interpretation of software metrics, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM, 2015, pp. 1498–1503.

[71] R. Willamy, J. Nunes, M. Perkusich, A. S. Freire, R. M. Saraiva, H. O. Almeida, A. Perkusich, A method to build bayesian networks based on artifacts and metrics to assess agile projects., in: SEKE, 2016, pp. 81–86.

[72] I. Ancveire, I. Gailite, M. Gailite, J. Grabis, Software delivery risk management: Application of bayesian networks in agile software development, Information Technology and Management Science 18 (1) (2015) 62–69.

[73] S. K. Bansal, A. Jolly, An encyclopedic approach for realization of security activities with agile methodologies, in: 2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence), IEEE, 2014, pp. 767–772.

[74] E. Gul, A fuzzy system model for task implementation in extreme programming process, in: Third International Conference on Software, Services and Semantic Technologies S3T 2011, Springer, 2011, pp. 111–117.

[75] S. M. El-Said, M. Hana, A. S. Eldin, Agile tailoring tool (att): A project specific agile method, in: 2009 IEEE International Advance Computing Conference, IEEE, 2009, pp. 1659–1663.

[76] R. Britto, P. S. Neto, R. Rabelo, W. Ayala, T. Soares, A hybrid approach to solve the agile team allocation problem, in: 2012 IEEE Congress on Evolutionary Computation, 2012, pp. 1–8. doi:10.1109/CEC.2012.6252999.

[77] F. Mafakheri, F. Nasiri, M. Mousavi, Project agility assessment: an integrated decision analysis approach, Production Planning and Control 19 (6) (2008) 567–576.

[78] R. Dwivedi, D. Gupta, Applying machine learning for configuring agile methods, International Journal of Software Engineering and its applications 9 (3) (2015) 29–40.

[79] W. Pedrycz, Quantitative logic-based framework for agile methodologies, Journal of Systems Architecture 52 (11) (2006) 700–707.

[80] H. Sedehi, G. Martano, Metrics to evaluate & monitor agile based software development projects-a fuzzy logic approach, in: 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement, IEEE, 2012, pp. 99–105.

[81] K. Logue, K. McDaid, Agile release planning: Dealing with uncertainty in development time and business value, in: 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008), IEEE, 2008, pp. 437–442.

[82] K. McDaid, D. Greer, F. Keenan, P. Prior, G. Coleman, P. S. Taylor, Managing uncertainty in agile release planning., in: SEKE, 2006, pp. 138–143.

[83] R. Dwivedi, D. Gupta, Customizing agile methods using genetic algorithms, in: Proc. of Int. Conf. on Advances in Communication, Network, and Computing, CNC, CNC '14, 2014.

[84] X. Dong, Q. S. Yang, Q. Wang, J. Zhai, G. Ruhe, Value-risk trade-off analysis for iteration planning in extreme programming, in: 2011 18th Asia-Pacific Software Engineering Conference, 2011, pp. 397–404. doi:10.1109/APSEC.2011.11.

[85] S. Jansi, K. C. Rajeswari, A greedy heuristic approach for sprint planning in agile software development, International Journal For Trends In Engineering Technology 3 (1) (2015) 18 – 21.

[86] V. C. Gerogiannis, P. G. Ipsilandis, Multi objective analysis for timeboxing models of software development, in: Proceedings of the Second International Conference on Software and Data Technologies, ICSOFT'2007, 2007.

[87] S. Menachem, Y. Reich, Improving effectiveness of agile development, in: Proceedings of ICED 09, the 17th International Conference on Engineering Design, 2009, pp. 311 – 322.

[88] V. S. Nepomuceno, M. E. Fontana, Decision support system to project software management, in: 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013, pp. 964–969. doi:10.1109/SMC.2013.170.

[89] Á. Szőke, Agile release planning through optimization, 2009, pp. 149–160.

[90] M. Golfarelli, S. Rizzi, E. Turricchia, Sprint planning optimization in agile data warehouse design, in: Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery, DaWaK'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 30–41. doi:10.1007/978-3-642-32584-7_3.
URL http://dx.doi.org/10.1007/978-3-642-32584-7_3

[91] Á. Szőke, Conceptual scheduling model and optimized release scheduling for agile environments, Information and Software Technology 53 (6) (2011) 574 – 591, special Section: Best papers from the APSEC. doi:https://doi.org/10.1016/j.infsof.2011.01.008.
URL http://www.sciencedirect.com/science/article/pii/S095058491100019X

[92] F. Sobiech, B. Eilermann, A. Rauch, On iteration optimization for non-cross-functional teams in scrum, in: Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems, RACS '14, ACM, New York, NY, USA, 2014, pp. 266–271. doi:10.1145/2663761.2664198.
URL http://doi.acm.org/10.1145/2663761.2664198

[93] G. van Valkenhoef, T. Tervonen, B. de Brock, D. Postmus, Quantitative release planning in extreme programming, Information and Software Technology 53 (11) (2011) 1227 – 1235, aMOST 2010. doi:https://doi.org/10.1016/j.infsof.2011.05.007. URL http://www.sciencedirect.com/science/article/pii/S0950584911001340

[94] Á. Szőke, Optimized Feature Distribution in Distributed Agile Environments, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 62–76. doi:10.1007/978-3-642-13792-1_7. URL https://doi.org/10.1007/978-3-642-13792-1_7

[95] J. M. Chaves-Gonzlez, M. A. Prez-Toledano, A. Navasa, Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm, Knowledge-Based Systems 83 (Supplement C) (2015) 105 – 115. doi:https://doi.org/10.1016/j.knosys.2015.03.012. URL http://www.sciencedirect.com/science/article/pii/S0950705115001069

[96] Á. Szőke, Decision Support for Iteration Scheduling in Agile Environments, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 156–170. doi:10.1007/978-3-642-02152-7_13. URL https://doi.org/10.1007/978-3-642-02152-7_13

[97] C. S. Kumar, A. A. Kumari, R. S. Perumal, An optimized agile estimation plan using harmony search algorithm, International Journal of Engineering and Technology 6 (5) (2014) 1994 – 2001.

[98] R. Ankori, Automatic requirements elicitation in agile processes, in: Software-Science, Technology and Engineering, 2005. Proceedings. IEEE International Conference on, IEEE, 2005, pp. 101–109.

[99] O. Alshareet, An empirical study to develop a decision support system (dss) for measuring the impact of quality measurements over agile software development (asd), Indian Journal of Science and Technology 8 (15).

[100] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, W. Pedrycz, Predicting development effort from user stories, in: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, IEEE, 2011, pp. 400–403.

[101] R. Kevin Sungkur, M. Ramasawmy, Knowledge4scrum, a novel knowledge management tool for agile distributed teams, VINE 44 (3) (2014) 394–419.

[102] I. Fronza, A. Sillitti, G. Succi, J. Vlasenko, Toward a non invasive control of applications: A biomedical approach to failure prediction, in: International Conference on Enterprise Information Systems (ICEIS), 2011, pp. 1–10.

[103] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, G. Succi, Effort prediction in iterative software development processes–incremental versus global prediction models, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), IEEE, 2007, pp. 344–353.

[104] F. A. Batarseh, A. J. Gonzalez, Predicting failures in agile software development through data analytics, Software Quality Journal 26 (1) (2018) 49–66.

[105] A. Panda, S. M. Satapathy, S. K. Rath, Neural network models for agile software effort estimation based on story points, in: Proceedings of the International Conference on Advances in Computing, Control and Networking, 2015, pp. 26–30.

[106] S. M. Satapathy, A. Panda, S. K. Rath, Story point approach based agile software effort estimation using various svr kernel methods (2014) 304–307.

[107] Á. Szőke, Optimized feature distribution in distributed agile environments, in: International Conference on Product Focused Software Process Improvement, Springer, 2010, pp. 62–76.

[108] N. Hanakawa, K. Okura, A project management support tool using communication for agile software development, in: Software engineering conference, 2004. 11th asia-pacific, IEEE, 2004, pp. 316–323.

[109] K. Ghane, A model and system for applying lean six sigma to agile software development using hybrid simulation, in: Technology Management Conference (ITMC), 2014 IEEE International, IEEE, 2014, pp. 1–4.

[110] A. Yasin, S. Gaber, M. Omar, H. Mohd, F. Baharom, M. M. Din, Designing story card in extreme programming using machine learning technique, in: Knowledge Management International Conference (KMICe), 2012.

[111] E. W. Larson, C. F. Gray, A guide to the project management body of knowledge: Pmbok (®) guide, Project Management Institute, 2015.

[112] P. Hearty, N. Fenton, D. Marquez, M. Neil, Predicting project velocity in xp using a learning dynamic bayesian network model, IEEE Transactions on Software Engineering 35 (1) (2009) 124–137.

[113] E. Dantas, M. Perkusich, E. Dilorenzo, D. F. Santos, H. Almeida, A. Perkusich, Effort estimation in agile software development: An updated review, International Journal of Software Engineering and Knowledge Engineering 28 (11n12) (2018) 1811–1831.

[114] S. Jansi, A greedy heuristic approach for sprint planning in agile software planning, International Journal for Trends in Engineering & Technology 3 (1) (2015) 18–21.

[115] R. Colomo-Palacios, I. González-Carrasco, J. L. López-Cuadrado, Á. García-Crespo, Resyster: A hybrid recommender system for scrum team roles based on fuzzy and rough sets, International Journal of Applied Mathematics and Computer Science 22 (4) (2012) 801–816.

[116] S. Čelar, M. Turić, L. Vicković, Method for personal capability assessment in agile teams using personal points, in: Telecommunications Forum Telfor (TELFOR), 2014 22nd, IEEE, 2014, pp. 1134–1137.

[117] G. van Valkenhoef, T. Tervonen, B. de Brock, D. Postmus, Quantitative release planning in extreme programming, Information and software technology 53 (11) (2011) 1227–1235.

[118] I. Stamelos, Software project management anti-patterns, Journal of Systems and Software 83 (1) (2010) 52–59.

[119] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, S. Brinkkemper, Improving agile requirements: the quality user story framework and tool, Requirements Engineering 21 (3) (2016) 383–403.

[120] M. Staron, W. Meding, C. Höglund, P. Eriksson, J. Nilsson, J. Hansson, Identifying implicit architectural dependencies using measures of source code change waves, in: Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on, IEEE, 2013, pp. 325–332.

[121] C. Fernández-Sánchez, J. Díaz, J. Pérez, J. Garbajosa, Guiding flexibility investment in agile architecting, in: System Sciences (HICSS), 2014 47th Hawaii International Conference on, IEEE, 2014, pp. 4807–4816.

[122] C. S. A. Peixoto, A. E. A. da Silva, A conceptual knowledge base representation for agile design of human-computer interface, in: Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on, Vol. 1, IEEE, 2009, pp. 156–160.

[123] F. McCarey, M. O. Cinnéide, N. Kushmerick, An eclipse plugin to support agile reuse, in: International Conference on Extreme Programming and Agile Processes in Software Engineering, Springer, 2005, pp. 162–170.

[124] Y. Lin, P. Descamps, N. Gaud, V. Hilaire, A. Koukam, Multi-agent system for intelligent scrum project management, Integrated Computer-Aided Engineering 22 (3) (2015) 281–296.

[125] M. Štolc, I. Polášek, A visual based framework for the model refactoring techniques, in: 2010 IEEE 8th International Symposium on Applied Machine Intelligence and Informatics (SAMI), IEEE, 2010, pp. 72–82.

[126] B. Kitchenham, L. Pickard, S. L. Pfleeger, Case studies for method and tool evaluation, IEEE software 12 (4) (1995) 52–62.

[127] D. Heckerman, A tutorial on learning with bayesian networks, in: Innovations in Bayesian networks, Springer, 2008, pp. 33–82.

[128] K. B. Korb, A. E. Nicholson, Bayesian artificial intelligence, CRC press, 2010.

[129] K. Verbert, R. Babuška, B. De Schutter, Bayesian and dempster–shafer reasoning for knowledge-based fault diagnosis–a comparative study, Engineering Applications of Artificial Intelligence 60 (2017) 136–150.

[130] Y. Hu, X. Mo, X. Zhang, Y. Zeng, J. Du, K. Xie, Intelligent analysis model for outsourced software project risk using constraint-based bayesian network, Journal of software 7 (2) (2012) 440–449.

[131] B. Yet, Z. Perkins, N. Fenton, N. Tai, W. Marsh, Not just data: A method for improving prediction with knowledge, Journal of biomedical informatics 48 (2014) 28–37.

[132] L. L. Minku, E. Mendes, B. Turhan, Data mining for software engineering and humans in the loop, Progress in Artificial Intelligence 5 (4) (2016) 307–314.

[133] J.-B. Lamy, B. Sekar, G. Guezennec, J. Bouaud, B. Séroussi, Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach, Artificial intelligence in medicine 94 (2019) 42–53.

[134] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explanations: An overview of interpretability of machine learning,

in: 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA), IEEE, 2018, pp. 80–89.

[135] D. Gunning, Explainable artificial intelligence (xai), Defense Advanced Research Projects Agency (DARPA), nd Web 2.

[136] R. Goebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, A. Holzinger, Explainable ai: the new 42?, in: International Cross-Domain Conference for Machine Learning and Knowledge Extraction, Springer, 2018, pp. 295–303.

[137] Y. Dong, H. Su, J. Zhu, B. Zhang, Improving interpretability of deep neural networks with semantic information, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4306–4314.

[138] L. Cao, Data science: a comprehensive overview, ACM Computing Surveys (CSUR) 50 (3) (2017) 43.

[139] A. Constantinou, N. Fenton, Towards smart-data: Improving predictive accuracy in long-term football team performance, Knowledge-Based Systems 124 (2017) 93 – 104. doi:https://doi.org/10.1016/j.knosys.2017.03.005.
URL http://www.sciencedirect.com/science/article/pii/S0950705117301223

[140] L. C. Briand, W. L. Melo, J. Wüst, Assessing the applicability of fault-proneness models across object-oriented software projects, IEEE transactions on Software Engineering (7) (2002) 706–720.

[141] B. Turhan, T. Menzies, A. B. Bener, J. Di Stefano, On the relative value of cross-company and within-company data for defect prediction, Empirical Software Engineering 14 (5) (2009) 540–578.

[142] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, B. Murphy, Cross-project defect prediction: a large scale experiment on data vs. domain

vs. process, in: Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ACM, 2009, pp. 91–100.

[143] Y. Zhou, T. M. Hospedales, N. Fenton, When and where to transfer for bayesian network parameter learning, Expert systems with applications 55 (2016) 361–373.

[144] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer Science & Business Media, 2012.

[145] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Information and software technology 55 (12) (2013) 2049–2075.

Complete list of selected papers

Table A.12: Complete list of selected papers.

Study ID	Paper ID	Title	ISE technique	Purpose
1	1	A genetic algorithm approach to release planning in agile environment	Search and optimization	Software requirements
2	2	An empirical study to develop a Decision Support System (DSS) for measuring the impact of quality measurements over agile Software Development (ASD)	Machine learning	Software engineering process
3	3	A decision support system utilizing a semantic agent	Semantic matching	Software engineering management
4	4	PBURC: A patterns-based, unsupervised requirements clustering framework for distributed agile software development	Machine learning	Software requirements
26	5	A model to detect problems on scrum-based software development projects	Probabilistic methods for uncertain reasoning	Probabilistic methods for uncertain reasoning

5	6	Decision support system to project software management	Search and optimization	Software engineering management
6	7	Incorporating artificial intelligence technique into DSDM	Case-based Reasoning	Software testing
7	8	An integrated approach for requirement selection and scheduling in software release planning	Search and optimization	Software requirements
8	9	Towards a framework for assessing agility	Multiple Criteria Decision Analysis and Probabilistic methods for uncertain reasoning	Software engineering process
9	10	Predicting fault-proneness of object-oriented system developed with agile process using learned Bayesian network	Search and optimization	Software quality
10	11	Value-risk trade-off analysis for iteration planning in eXtreme Programming	Search and optimization	Software requirements

11	12	Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points	Machine learning	Software engineering management
12	13	A regression test selection technique by optimizing user stories in an Agile environment	Graph Theory	Software testing
13	14	Dealing the selection of project management through hybrid model of verbal decision analysis	Multiple Criteria Decision Analysis	Software engineering management
14	15	An encyclopedic approach for realization of security activities with agile methodologies	Fuzzy logic	Software engineering management
15	16	Predicting development effort from user stories	Machine learning	Software engineering management
16	17	NextMove: A framework for distributed task coordination	Multiple Criteria Decision Analysis	Software engineering management
17	18	Agile release planning through optimization	Search and optimization	Software engineering management

18	19	Improving effectiveness of Agile development	Search and optimization	Software engineering management
19	20	A Fuzzy System Model for Task Implementation in Extreme Programming Process	Fuzzy logic	Software quality
20	21	A Heuristic Approach to Solve the Elementary Sprint Optimization Problem for Non-cross-functional Teams in Scrum	Multiple Criteria Decision Analysis and Search and optimization	Software requirements
21	22	Quantitative release planning in extreme programming	Search and optimization	Software engineering management
22	23	A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation	Mathematical model	Software engineering management
23	24	Story Points Based Effort Estimation Model for Software Maintenance	Mathematical model	Software quality
20	25	On Iteration Optimization for Non-cross-functional Teams in Scrum		Software engineering management

24	26	Approximation of COSMIC functional size to support early effort estimation in Agile	Natural Language Processing	Software engineering management
25	27	A procedure to detect problems of processes in software development projects using Bayesian networks	Probabilistic methods for uncertain reasoning	Software engineering process
26	28	Agile tailoring tool (ATT):A project specific agile method	Fuzzy logic	Software engineering management
27	29	Identifying implicit architectural dependencies using measures of source code change waves	Mathematical model	Software design
28	30	A Bayesian Network Model to Assess Agile Teams' Teamwork Quality	Probabilistic methods for uncertain reasoning	Software engineering process
29	31	A hybrid approach to solve the agile team allocation problem	Fuzzy logic-Search and optimization	Software engineering management
73	32	Ant colony optimization for the next release problem a comparative study	Search and optimization	Software engineering management
30	33	Applying case based reasoning in agile software development	Case-based Reasoning	Software requirements

31	34	Multi-Agent System for intelligent Scrum project management	Multi-agent system	Software engineering management
32	35	Supplier ranking by multi-alternative proposal analysis for agile projects	Mathematical model	Software engineering management
33	36	Resyster: A hybrid recommender system for scrum team roles based on fuzzy and rough sets	Recommender System	Software engineering management
34	37	A model and system for applying Lean Six sigma to agile software development using hybrid simulation	Machine learning	Software engineering management
35	38	Designing Story Card in Extreme Programming Using Machine Learning Technique	Machine learning	Software requirements
13	39	Towards a Verbal Decision Analysis on the Selecting Practices of Framework SCRUM	Multiple Criteria Decision Analysis	Software engineering management
36	40	Optimized Feature Distribution in Distributed Agile Environments	Machine learning / Search and optimization	Software engineering management

37	41	Incremental effort prediction models in agile development using radial basis functions	Machine learning	Software engineering management
38	42	Story point approach based agile software effort estimation using various SVR kernel methods	Machine learning	Software engineering management
39	43	Project agility assessment: An integrated decision analysis approach	Fuzzy logic	Software engineering process
40	44	A framework for systematic evaluation of process improvement priorities	Graph Theory	Software engineering process
41	45	Multi objective analysis for timeboxing models of software development	Search and optimization	Software engineering management
42	46	Partial selection of agile software requirements	Recommender System	Software requirements
43	47	Optimal refactoring policy for agile information systems maintenance: A control theoretic approach	Mathematical model	Software quality
44	48	Knowledge4Scrum, a novel knowledge management tool for agile distributed teams	Machine learning	Software engineering management

45	49	Handling development time uncertainty in agile release planning	Statistical method	Software engineering management
46	50	A visual based framework for the model refactoring techniques	Rules	Software design
47	51	Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm	Search and optimization	Software requirements
48	52	Guiding flexibility investment in agile architecting	Multiple Criteria Decision Analysis	Software design
49	53	Toward a non invasive control of applications: A biomedical approach to failure prediction	Machine learning	Software quality
50	54	A conceptual knowledge base representation for agile design of human-computer interface	Semantic network	Software design
51	55	Conceptual scheduling model and optimized release scheduling for agile environments	Search and optimization	Software engineering management

52	56	Lightweight risk management in agile projects	Rules	Software engineering management
53	57	Simulations of agile software processes for healthcare information systems development based on machine learning methods	Multi-agent system	Software engineering management
54	58	Applying machine learning for configuring agile methods	Fuzzy logic and Machine learning	Software engineering management
55	59	Intelligent agent (IA) systems to generate user stories for a positive user experience	Multi-agent system	Software requirements
56	60	Using Bayesian belief networks to model software project management antipatterns	Probabilistic methods for uncertain reasoning	Software engineering management
57	61	A hybrid model for agile practices using case based reasoning	Case-based Reasoning	Software engineering management
58	62	Human-centered software development methodology in mobile computing environment: Agent-supported agile approach	Multi-agent system	Software requirements

59	63	A Bayesian based method for agile software development release planning and project health monitoring	Probabilistic methods for uncertain reasoning	Software engineering management
60	64	A method of analysis to uncover artefact-communication relationships	Communication Theory	Software requirements
61	65	Improving agile requirements: the Quality User Story framework and tool	Natural Language Processing	Software requirements
62	66	A Lagrangian heuristic for sprint planning in agile software development	Search and optimization	Software engineering management
63	67	Automatic requirements elicitation in agile processes	Machine learning	Software requirements
64	68	Change-impact driven agile architecting	Rules	Software requirements
65	69	Quantitative logic-based framework for agile methodologies	Rules	Software engineering management
66	70	Metrics to evaluate & monitor agile based software development projects - A fuzzy logic approach	Fuzzy logic / Rules	Software engineering management

67	71	Human-computer interface expert system for agile methods	Rules	Software design
68	72	An Eclipse plugin to support Agile Reuse	Recommender System	Software design
62	73	Multi-sprint planning and smooth replanning: An optimization model	Machine learning	Software engineering management
69	74	Effort prediction in iterative software development processes - incremental versus global prediction models	Machine learning	Software engineering management
45	75	Handling uncertainty in agile requirement prioritization and scheduling using statistical simulation	Statistical method	Software requirements and Software engineering management
70	76	Predicting project velocity in XP using a learning dynamic Bayesian network model	Probabilistic methods for uncertain reasoning	Software engineering management
71	77	Phase Wise Effort Estimation for Software Maintenance: An Extended SMEEM Model	Mathematical model	Software engineering management
73	78	Multi-objective ant colony optimization for requirements selection	Search and optimization	Software engineering management

72	79	Predicting failures in agile software development through data analytics	Machine learning	Software quality
74	80	A Bayesian network approach to assist on the interpretation of software metrics	Probabilistic methods for uncertain reasoning	Software engineering management
75	81	A Method to Build Bayesian Networks based on Artifacts and Metrics to Assess Agile Projects	Probabilistic methods for uncertain reasoning	Software engineering management
76	82	Software Delivery Risk Management: Application of Bayesian Networks in Agile Software Development	Probabilistic methods for uncertain reasoning	Software engineering management
77	83	Effort Estimation in Agile Software Projects using Fuzzy Logic and Story Points	Fuzzy logic	Software engineering management
77	84	Towards a Fuzzy based Framework for Effort Estimation in Agile Software Development	Fuzzy logic	Software engineering management
11	85	Neural Network Models for Agile Software Effort Estimation based on Story Points	Machine learning	Software engineering management

78	86	A decision model for agile software release	Multiple Criteria Decision Analysis	Software engineering management
62	87	A Greedy Heuristic Approach for Sprint Planning in Agile Software Development	Search and optimization	Software engineering management
13	88	Applying Verbal Decision Analysis in the Selecting Practices of Framework SCRUM	Multiple Criteria Decision Analysis	Software engineering management
79	89	Bayesian network based xp process modelling	Probabilistic methods for uncertain reasoning	Software engineering management
80	90	Customizing Agile Methods using Genetic Algorithms	Search and optimization	Software engineering management
81	91	Sprint planning optimization in agile data warehouse design	Search and optimization	Software engineering management
82	92	Design of a multi-agent system architecture for the scrum methodology	Multi-agent system	Software engineering management
83	93	Managing uncertainty in agile release planning	Statistical method	Software engineering management

84	94	Decision support for iteration scheduling in agile environments	Search and optimization	Software engineering management
85	95	A project management support tool using communication for agile software development	Machine learning	Software engineering management
13	96	An Applicability in Verbal Decision Analysis for Selecting Approaches from Framework Scrum	Multiple Criteria Decision Analysis	Software engineering management
86	97	Bayesian network model for task effort estimation in agile software development	Probabilistic methods for uncertain reasoning	Software engineering management
87	98	Project Reliability Growth Model Based on Curves of Accumulated Communication Topics for Software Development	Mathematical model	Software engineering management
88	99	Enhancing Quality in Scrum Software Projects	Mathematical model	Software engineering management
89	100	Scaling Agile Estimation Methods with a Parametric Cost Model	Mathematical model	Software engineering management
90	101	An Efficient Approach for Agile Web Based Project Estimation: AgileMOW	Mathematical model	Software engineering management

91	102	An Optimized Agile Estimation Plan Using Harmony Search Algorithm	Search and optimization	Software engineering management
92	103	Improving Resource Leveling in Agile Software Development Projects through Agent-Based Approach	Multi-agent system	Software engineering management
93	104	Method for personal capability assessment in agile teams using personal points	Graph Theory	Software engineering management