Particle Swarm Optimization in Small Case Bases for Software Effort Estimation

Katharina Landeis[1,2], Gerhard Pews[1], and Mirjam Minor[2(✉)]

[1] Capgemini, Cologne, Germany
[2] Department of Business Informatics, Goethe University Frankfurt, Frankfurt, Germany
minor@cs.uni-frankfurt.de

Abstract. To facilitate research in the field of user story effort estimation (USEE), this work analyzes the applicability of case-based reasoning to effort estimation of user stories. The analysis uses real-world user stories of a software development project of the Capgemini Group. The focus of the analysis is the development of an effort estimation tool that could generate accurate effort estimates serving the purposes of project planning and control. In addition to a classical structural CBR approach, the paper applies the weights optimization method Particle Swarm Optimization to small case bases. The highly accurate effort estimates resulting from a couple of experiments on real data show that estimation by analogy presents with the aid of an automated environment an eminently practical technique. As a consequence, this approach impacts estimation accuracy and thus success of industrial software development projects.

Keywords: CBR applications · Software engineering · Few data · Particle swarm optimization

1 Introduction

Software effort estimation (SEE) is the process of approximating the amount of effort needed to complete project activities [24]. In agile software development, a project comprises several iterations, each of which delivers a set of requirements known as user stories [1]. The success or failure of an agile software project depends on accuracy in estimating the effort of user stories [16]. Underestimates bear the risk that projects reveal to exceed the budget. When efforts are overestimated, the company might mistakenly reject potentially profitable user stories or even projects.

Numerous studies on effort estimation of user stories have been published in the recent past [1,6,12,16,25]. They report on many different estimation techniques of expert-based or data-driven kind. The data-driven approaches have in common that they require a large amount of data. Malgonde and Chari's work [12] with around 500 stories are among those with smaller data sets. In practice, however, many repositories with user stories are distinctly sparser for

M. T. Keane and N. Wiratunga (Eds.): ICCBR 2022, LNAI 13405, pp. 209–223, 2022.
https://doi.org/10.1007/978-3-031-14923-8_14

privacy or further business concerns. Our research is investigating how accurate estimations can be derived from small repositories of user stories. We have developed a Case-based Reasoning (CBR) approach for estimating user stories' effort that uses particle swarm optimization. It turned out during our experiments with four small case-bases (below 60 cases each) that CBR outperforms the estimation accuracy of expert judgements achieved with the Planning Poker technique.

The remainder of the paper is organized as follows. Related work is discussed in Sect. 2. The application scenario is introduced in further detail in Sect. 3. The CBR approach is presented in Sect. 4. Next, Sect. 5 reports on the experiments with real-world data from a software project of the Capgemini Group. Finally, a conclusion is drawn in Sect. 6.

2 Related Work

SEE by analogy has already a long tradition in CBR [7,8,15,20,27,28]. These approaches have in common that a software project is represented as a case and is characterized by a set of features that are regarded as effort drivers, such as the programming languages or complexity of the project. In contrast to our approach those publications focus on entire software projects. Today, agile software development is of increasing importance. This means that the requirements whose effort can be estimated are described in the form of user stories. Those smaller, less complex units have to our knowledge not yet been considered in case-based research on SEE.

Recently, the optimization of CBR approaches in SEE has attracted considerable research attention as approaches like particle swarm optimization (PSO) and genetic algorithm (GA) have been used to optimize weights, parameters, and select features [7,8,27,28]. These research results have shown that CBR method with optimized weights derived by PSO or GA can improve estimation results. Furthermore, there exist results verifying that PSO is able to outperform GA in searching the optimized weight of CBR [14]. Again, there do not exist weights optimization approaches in the context of user story effort estimation (USEE). As the PSO method is known to outperformed GA [14], this method is chosen to identify optimal weights for effort drivers in our CBR approach.

Additionally, related approaches for optimization have gained attention in CBR research [9,22,26]. The work of Jarmulak et al. [9] optimizes the case-base index and the feature weights for similarity measure simultaneously for the application area of tablet formulation. Stahl and Gabel [22] implement a novel approach for optimizing the similarity assessment in CBR. Their work employs evolutionary algorithms for optimizing similarity measures by means of relative case utility feedback. The work of Wiratunga et al. [26] introduces learning-to-learn personalised models from few data. It applies a personalized meta-learning approach to self-management cases for patients of back-pain. These approaches provide alternative solutions to PSO in the context of CBR approaches, the latter with few available data. They have not yet been considered in our work but might inspire future work on alternative or hybrid approaches for USEE.

3 Software Effort Estimation of User Stories

SEE is an active research field since the 1970s. This dynamic field of SEE has undergone several changes in the last four decades, whereas one significant development emerged in the course of the agile movement.

The shift from traditional to agile methodologies used in the development process changed the perspective on estimation and planning in the last twenty years. In agile development settings, the project is characterized by incremental development in small iterations implying that estimations should be done progressively. Whereas in each iteration a number of user stories is implemented. Thus, the estimation in agile projects especially focuses on estimating user stories rather than whole projects [4].

A user story is a common way for agile teams to express user requirements [4] and describes functionality that will be valuable to either a user or purchaser of a software [5]. Figure 1 shows an exemplary story of this project.

Due to the importance of reliable effort estimates for effective software project management, the field of SEE was investigated actively in the last four decades, leading to the evolvement of various estimation models and techniques [23]. Nevertheless nowadays, it is common practice to apply expert-based estimation techniques to a great extent and the application of effort estimation techniques other than expert judgement is still immature [25].

Expert-based approaches require a subjective assessment of experts [17]. For this approach, different methods exist that guide the consultation process of experts. Expert-based estimation with a structured group-consensus approach is represented by the principles of Wideband Delphi [24]. A widely practiced expert-based technique in agile software development is Planning Poker (PP) which principles are in fact derived from the group-consensus methods of Wideband Delphi [24].

This gamified estimation method involves discussion among the team members [3,16]. For each user requirement, all team members make their estimate by choosing a single card from their own deck of cards. Each card from the deck shows one number which represents the estimated effort typically measured in days or story points. Story points are a relative unit of measure. All cards, this means all estimates, are revealed concurrently. If any discrepancy occurs, then discussion among team members takes place to find consensus. Finally, the agreed estimation is set down as finalized and the next user requirement is taken into consideration.

PP does not require any historical data and is applicable in any phase of software development. Typically, estimation based on group discussion contributes to a better understanding of software development problems or the identification of weaknesses in software requirements. Moreover, human experts can handle differences between past project experiences and new techniques, architectures or unique project characteristics. Besides its strengths, this group estimation method involves multiple experts with advanced expertise in the developed software which makes the estimation relatively expensive. This approach does not provide a reusable estimation model as the estimation procedure must be

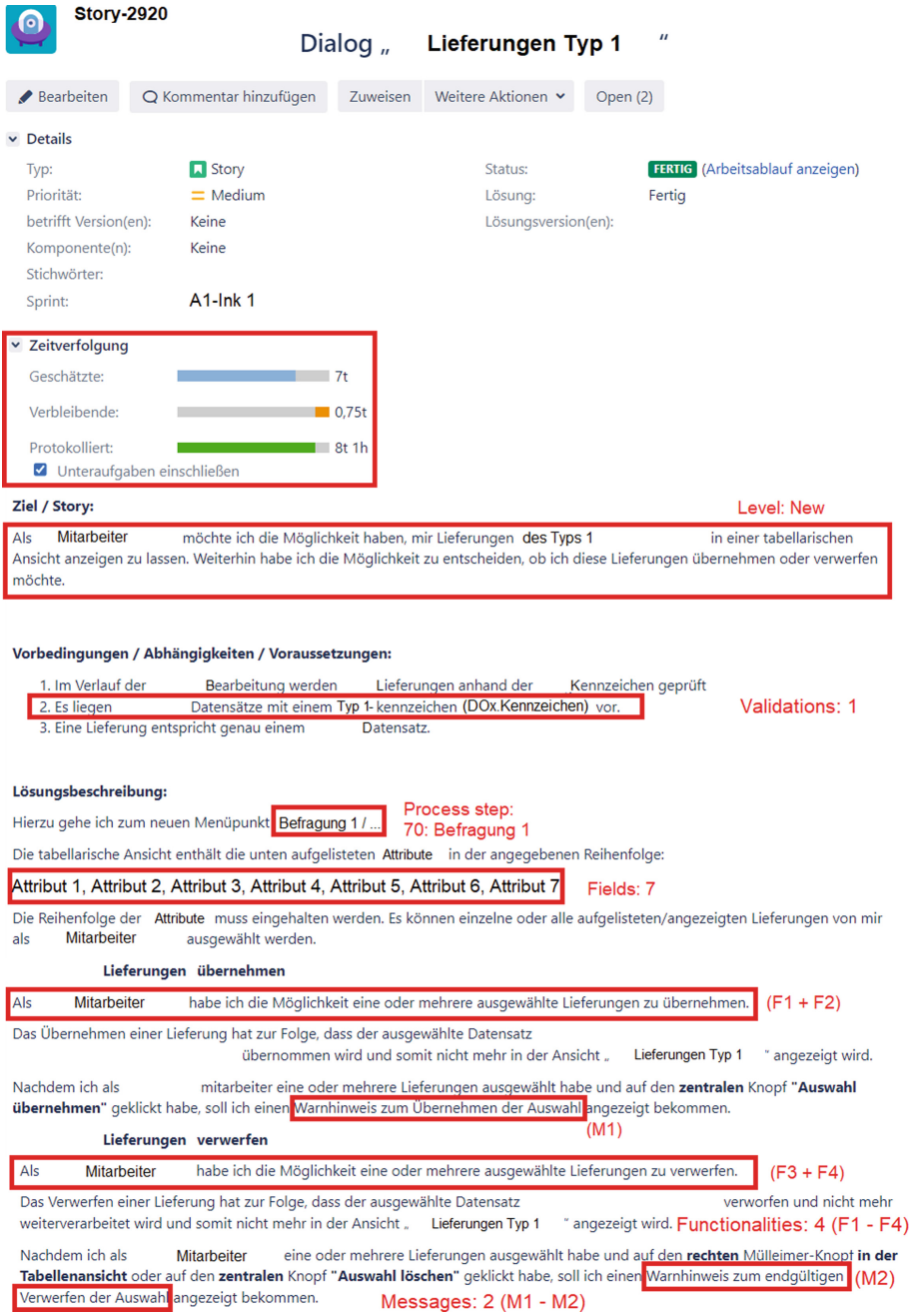

Fig. 1. Exemplary user story

repeated each time estimates are needed. Additionally, expert estimates can not provide objective and quantitative analysis of project effort dependencies and can be biased by numerous factors like individual expertise or preferences [24].

4 CBR Approach

The key to successful estimation is to identify relevant effort drivers [24]. In terms of CBR, this is achieved by defining suitable case representations for the description of user stories first. Next, the similarity functions are specified. An adaptation rule is described to derive estimation values from multiple cases. In addition to the classical CBR approach, the weights of the similarity functions are optimized in order to balance the impact of the relevant effort drivers on the system's estimation. The specific contribution of our optimization approach is that it works successfully for very small case-bases.

4.1 Case Representation

Interviews with experts revealed that effort drivers in agile software development are dependent on the project context. It is difficult to define uniform effort drivers for all user stories. Hence, the user stories have been grouped into different case-bases depending on their main purpose. For instance, stories that primarily have an effect on the user interface can be classified as stories of the category 'UI'. The UI stories can be further distinguished according to sub-categories such as 'Dialog' or 'View'. For each sub-category, an own case-base is specified. The cases record those feature-value pairs that are the most relevant effort drivers for the purpose of the case-base, i.e. the case representation implements a structural CBR approach [18]. The solution part of the cases comprises a numerical value expressing the actual effort recorded post mortem.

4.2 Similarity

The similarity functions follow the local-global principle [18]. The similarity sim for a problem q and a case $c = (p, l)$ from the case-base is calculated as follows:

$$sim(q, p) = \Phi(sim_{A_1}(q_1, p_1), \ldots, sim_{A_n}(q_n, p_n))$$

where $\Phi : \mathbb{R}^n \mapsto \mathbb{R}$ denotes an amalgamation function and $sim_{A_i} : T_i \times T_i \mapsto \mathbb{R}$ a local similarity function for the i-th feature (with range T_i) of the case representation. We have chosen to use the most prevalent amalgamation function namely the linear sum $\Phi(s_1, \ldots s_n) = \sum_{i=1\ldots n} \omega_i \cdot s_i$ where ω_i is the weight of the i-th feature.

4.3 Adaptation

The effort estimate for the new user story that is described in the query case is based on the best matching cases identified in the retrieval step. We have chosen

a multiple case adaptation approach which generates the effort estimate using the weighted mean of the actual effort values of the k most similar source cases:

$$\text{PredictedEffort} = \sum_{i=1}^{k} \frac{sim(q, p_i)}{\sum_{i=1}^{k} sim(q, p_i)} \cdot \text{ActualEffort}_i$$

This adaptation rule calculates the weight factor on the basis of the similarity measure between the user story q to be estimated and the historical user story p_i. The weight factors are only used for normalization purposes. Moreover, as this adaptation rule is based on multiple cases, for instance $k = 3$, it allows to cope with varying development productivities among software engineers which arise from different levels of experience.

4.4 Weight Optimization with PSO

As defined in Sect. 4.2 the global similarity function is deduced from local similarity functions of each feature where the weights determine the feature's influence on global similarity. It is eminent to identify the best weights in the term of improved estimation accuracy. The optimization problem of discovering appropriate weights for the features can be solved by the Particle Swarm Optimization (PSO) method [10].

PSO simulates the behaviour of fishes and birds in swarms. The position of a particle represents a potential solution of the optimization problem. The goal is that a particle meets an optimal point in the solution space. Please note, that the solution space of the optimization problem differs from the solution space of the case-base since it addresses optimal weights for the similarity function. In PSO, a population of particles representing different solution proposals is created and updated iteratively. In any iteration, the fitness of each particle is calculated by putting its information into a designated objective function. In response to the fitness value and in accordance with the movement of the other particles, each particle is updated by adjusting its position in the solution space.

In an n-dimensional problem space, the position of the j-th particle at the iteration t is described by an n-dimensional particle vector $X_j^t = (x_{j1}^t, x_{j2}^t, ..., x_{jn}^t)$. The particle's velocity is as well defined as an n-dimensional vector $V_j^t = (v_{j1}^t, v_{j2}^t, ..., v_{jn}^t)$. The position of a particle is updated using the velocity vector in accordance with Wu et al. [28]:

$$X_j^{t+1} = X_j^t + V_j^{t+1}.$$

The velocity vector is adjusted (like in [28]) by considering its previous best position p_j according to the objective function, its current position, its current velocity and the previous global best position g of all particles:

$$V_j^{t+1} = i_t \cdot V_j^t + c_1 r_1 \cdot (p_j - X_j^t) + c_2 r_2 \cdot (g - X_j^t)$$

where i_t is a time-varying inertia weight (decreasing from iteration to iteration in order to slow down the impact of the former velocity, cmp. [28]), c_1, c_2 are positive constants and r_1, r_2 random numbers between 0 and 1.

After the particle population is initialized randomly, the fitness value of each particle is calculated. Then these values are compared with the fitness values of the best positions resulting in the updates of the individual best positions and the global best position. At least, the adjusted velocities and particles' positions lead to a new particle population.

In the context of USEE, the objective is to derive weights that generate highly accurate estimates. Therefore, one needs to decide on an accuracy metric on which the optimization problem is based. As the mean magnitude of relative error (MMRE) between estimated and actual efforts is the most frequently used accuracy metric, this metric is chosen to serve as the objective function.

It is defined by the magnitude of relative error (MRE) for a case

$$\text{MRE} = \frac{|ActualEffort - PredictedEffort|}{ActualEffort},$$

averaging the MRE values for a case-base with N cases:

$$\text{MMRE} = \frac{1}{N}\sum_{i=1}^{N} MRE_i.$$

Thus, the objective is to minimize the mean prediction error of all user stories whose effort values are aimed to be estimated.

The PSO method is applied separately to each case-base. A population with the same swarmsize of particles is generated for every case-base by randomly selecting initial weights. Each particle represents a potential similarity function. More precisely, a particle is a vector of weights to aggregate the values of the (given) local similarity functions where ω_i is the weight of the i-th feature, or in other words, the weight of the i-th local similarity measure. The ω_i's are chosen within lower and upper bounds pre-defined for each feature of the particular case representation.

The defined objective function shows that there is data required that serves as case-base and data that represents the new user stories whose effort values are aimed to be predicted accurately. Consequently, the fitness of the local similarity function (a particle) is determined by applying it to a case-base and measuring the accuracy of the predicted estimation values. Dividing the data set into these two containers could lead to the problem of over-fitting. This means that a CBR cycle initialized on weights derived in this way would provide a perfect accuracy score but would fail to predict anything useful on yet-unseen data.

The problem of over-fitting is encountered through a 3-fold cross-validation. Therefore, the whole data set is divided into a training set (80%) and a test set (20%). The test set contains user stories that the model has never seen before and is used for the final evaluation of the CBR approach. The training set is used for the search of optimized weights with the aid of PSO. This data set is split up

into 3 folds. Hence, the optimal weights are derived 3 times, every time two folds are used as case-base and one fold is utilized for accuracy evaluation purposes. Even though it is common practice in cross-validation to choose the parameters of the iteration with the lowest error rate, this principle is not applicable in the context of small case bases. Since each fold contains only a very small number of cases the error values can explode due to outliers. Thus, the optimal weight vectors of all iterations are utilized by calculating their median value.

5 Experiments

The experiments have been guided by two hypotheses:

H1 The PSO optimization improves the results of the case-based estimation approach.
H2 The case-based estimation approach with PSO outperforms human experts who use Planning Poker as an estimation technique.

The experimental environment has been created in myCBR [2]. Two experimental setups have been defined and conducted on the same experimental data that is described in the following. Experiment 1 addresses hypothesis H1 as described in Sect. 5.2. Experiment 2 focuses on hypothesis H2 (see Sect. 5.3). The results of both experiments are discussed in Sect. 5.4.

5.1 Experimental Data

The experimental data comprises 127 user stories from a currently running software development project of the Capgemini Group. The project whose user stories formed the data basis of the analysis has involved around 100 employees, consisting of project manager, software developers, business analysts and software testers. The project has been executed in the context of agile software development. Concretely, the software has been developed using Scrum, a process framework managing product development in an iterative, incremental procedure [19]. Consequently, the software was planned, estimated and delivered incrementally in small iterations, whereas in each iteration a certain number of user stories was realized.

The evaluated period of 17 months offered a total of 298 user stories that were realized in 11 increments. A major problem that SEE research community encounters is the heterogeneity of data. Many studies emphasize the demand for more homogeneous datasets to achieve higher accuracy levels [13,20,21]. The fact that all stories belong to the same project and at least show a similar context, complies the requirement of homogeneity. To strengthen this requirement, a subset of 127 stories that primarily have an effect on the user interface were selected as experimental data. This data set was divided into four self-contained case-bases whose technical implementations differed from each other. The topics and size of the four experimental case-bases are depicted in Table 1. Each case-base shows effort values of similar range (cmp. Table 5).

Table 1. Distribution of user stories across the category *UI*.

Dialog	Filtering	Report	View	$\sum$
32	18	59	18	127

'Dialog' user stories implement or extend any user interface which gives the user the possibility for data entry, for record selection and deletion with the aid of check-boxes, for triggering data processing through clicking on buttons and so forth. 'Filtering' user stories implement or extend user interfaces which consist of filters. An example case describes the requirement that the filter attributes for postal addresses should include an additional attribute for name affixes. 'Report' user stories only display data and inform the user on the current status of specified variables. 'View' user stories have the central purpose to display data and are characterized by limited user interactions. To be more precise, the allowed interactions are restricted to the actions belonging to the project's standard structure of interfaces. For instance, they do permit the user to remove or select columns and navigate forth and back. Stories involving more complex user interactions are put into category 'Dialog'.

The four case-bases contain slightly different feature sets (cmp. the second column of Table 4). Table 2 exemplarily depicts the feature set of one of the case bases.

Table 2. Feature set of the *Dialog* cases.

Feature	Domain
Level	New/Extension
Process step	20/25/27/30/35/40/50/60/70/80/90
Fields	0–30
Buttons	0–5
Messages	0–8
Validations	0–8
Functionalities	0–5
Complexity	Low/Medium/High

'Level' states whether a story specifies a new user interface which meets the properties of the user story or extends an existing one. 'Process step' references the identification number of the process step in the business process model that underlies the software and describes the software's functionality at a high-level, business-oriented perspective. Stories that fulfill requirements of advanced process steps tend to require more development effort than those of earlier steps. The domain of the feature is restricted to those identification numbers that occur within the software project's business process. 'Fields' captures the number of fields needed to be implemented. 'Buttons' accounts for the number of buttons specified in the story. 'Messages' describes the number of messages needed to be implemented. 'Validations' comprise various checks for the compliance of inserted

values by the user on the permitted range of values as well as checks for data records that are allowed to be displayed on the user interface. 'Functionalities' captures the number of functionalities specified in the story. This involves all actions that can manipulate data and are triggered by the user through clicking on buttons. 'Complexity' describes the story's complexity. The assignment of complexity to a user story is either more or less subjective but factors like the difficulty of functionalities, the complexity of SQL-scripts or html-files are the most relevant indicators.

Obviously, categorical local similarity functions are specified for 'Level', 'Process step' and 'Complexity'. MyCBR's 'polynomial' similarity function is used for 'Fields', 'Buttons' and 'Messages'. 'Validations' and 'Functionalities' are compared by another of MyCBR's standard similarity functions called 'smooth-step-at' [2]. The adaptation rule considers the three best matching cases, i.e. we have specified $k = 3$.

5.2 Experiment 1

Experiment 1 investigates the impact of the optimization and aims to quantify the improvement in terms of accuracy. As already introduced in Sect 4.4 the experimental data set is split up into a training set and a test set. Each test set of the four experimental case-bases is partitioned into three folds of the same size. Consequently, the optimal weights are derived three times, whereas in each iteration two folds act as case-base and one fold is used for accuracy evaluation.

The lower and upper bounds for the feature weights are specified depending on the subjective assessment of the relevance of the effort drivers. Figure 2 shows the bounds used for 'Dialog'.

```
lb = [3.0, 3.0, 3.0, 3.0, 1.0, 1.0, 4.0, 4.0]
ub = [6.0, 6.0, 6.0, 6.0, 3.0, 3.0, 6.0, 6.0]
```

Fig. 2. Exemplary definition of lower and upper bound.

The experiment is conducted with the *pyswarm* package an optimization package for python that implements PSO [11]. The number of particles in the swarm (*swarmsize*), the particle velocity scaling factor (*omega*), the scaling factor to search away from the particle's best known position (*phip*) and the scaling factor to search away from the swarm's best known position (*phig*) are assigned to their default values [11]. The minimum stepsize of swarm's best position before the search terminates (*minstep*) is set to 1e-8 and the minimum change of swarm's best objective value before the search terminates (*minfunc*) is set to 1e-15. The maximum number of iterations for the swarm to search (*maxiter*) has the value of 200. An exemplary optimizer for the first iteration, in which the on MMRE based objective function is named *error_global_S1*, is illustrated in Fig. 3.

```
pso(error_global_S1, lb, ub, ieqcons=[], f_ieqcons=None, args=(), kwargs={},
    swarmsize=100, omega=0.5, phip=0.5, phig=0.5, maxiter=200, minstep=1e-8,
    minfunc=1e-15, debug=False)
```

Fig. 3. Exemplary call of the PSO optimizer.

Table 3. Improvement in accuracy through weights optimization

Dialog	Filtering	Report	View
+ 9.20%	+ 1.17%	+ 3.29%	+ 2.35%

Table 4. Results of weights optimization.

Case base	Feature	Optimized weights			
		Per iteration			Median
		Fold1	Fold2	Fold3	
Dialog	Level	3.00	3.00	3.00	3.00
	Process step	3.00	3.82	3.00	3.00
	Fields	5.51	5.22	5.31	5.31
	Buttons	6.00	3.00	6.00	6.00
	Messages	3.00	1.00	1.00	1.00
	Validations	3.00	3.00	1.00	3.00
	Functionalities	4.00	6.00	5.26	5.26
	Complexity	5.98	4.00	5.26	5.26
Filtering	Level	1.00	1.00	1.00	1.00
	Process step	3.00	1.00	1.00	1.00
	Filters	4.00	4.00	6.00	4.00
	Additional columns	3.00	3.00	1.00	3.00
	Dialogs	3.00	1.00	1.00	1.00
	Filterconcept	2.00	2.00	1.25	2.00
	Concept maturity	3.00	3.00	1.00	3.00
Report	Level	6.00	6.00	2.55	6.00
	Reportconcept	5.15	6.00	5.38	5.38
	Type	1.35	1.94	2.71	1.94
	Contentfields	4.82	6.00	3.75	4.82
	Export option	1.00	1.00	1.65	1.00
	Export file	3.00	3.39	3.64	3.39
	Export execution	3.00	3.00	3.00	3.00
	Entities	1.79	3.00	2.13	2.13
View	Level	1.23	1.00	1.00	1.00
	Process step	3.00	1.67	1.00	1.67
	Columns/Fields	3.00	3.00	6.00	3.00
	Buttons	3.00	3.00	3.00	3.00
	Entities	6.00	6.00	3.00	6.00
	Filteroption	1.80	2.60	1.93	1.93

The resulting, optimized weights are depicted in Table 4. The evaluation of the CBR model on the test set led to a moderate improvement of accuracy compared to the original weights as shown in Table 3.

5.3 Experiment 2

Experiment 2 addresses the evaluation of the optimized CBR model in comparison to the estimation values obtained from human experts.

Table 5 shows the results for the test sets. All effort values are shown in hours. The MRE values for the test cases are fully listed. The accumulated values for all cases (for PP even including the three training folds) are provided in the boxes below the rows with the particular test cases. In addition to MMRE, the values of two further accuracy metrics are depicted. $MdMRE$ denotes the median of the MRE's which is less sensitive to outliers than the mean MRE. $PRED(x)$ measures the accuracy by the percentage of predictions that fall within x percent of the actual value [20]. In other words, the $PRED(x)$ is the percentage of MRE which is less than or equal to value x for all stories. It is common practice to use $x = 25$ as performance indicator [6]. The CBR method outperforms the experts for all four sample case-bases with respect to the MMRE. The MdMRE and $PRED(25)$ for the 'Report' case-base achieved a slightly better accumulated value with the PP method compared to the CBR method while for the other three case-bases CBR performed better.

5.4 Discussion of Results

Both hypotheses are confirmed by the experiments. Experiment 1 succeeded well in avoiding over-fitting since the accuracy of the estimation values for the user stories is improved by the optimization (cmp. Table 3). While the amount of improvement in a single digit range might seem marginal at a first glance the impact on software projects is significant at a closer look. In terms of staff-hours as well as in terms of reputation losses the economic consequences of such an improvement are high.

The results of experiment 2 show that the CBR approach clearly outperforms the PP approach. In addition to the direct benefits from improving the accuracy of the estimation values, the support and validation of PP sessions by an automated estimation approach could provide further advantages for the software projects.

Table 5. Results from Experiment 2.

Story-No.	Actual effort	PP			CBR		
		Predicted effort	MRE	PRED(25)	Predicted effort	MRE	PRED(25)
Dialog:							
7040	12	12	0.00	1	11.86	0.01	1
4769	22	16	0.27	0	17.25	0.22	1
7870	25	40	0.60	0	26.56	0.06	1
7572	36	36	0.00	1	32.39	0.10	1
3232	47	67	0.43	0	48.38	0.03	1
2570	72	100	0.39	0	79.14	0.10	1
Accumulated:		MMRE = 0.25 MdMRE = 0.15 PRED(25) = 0.56			MMRE = 0.09 MdMRE = 0.08 PRED(25) = 1.00		
Filtering:							
7880	32	24	0.25	0	33.13	0.04	1
3233	48	48	0.00	1	48.79	0.02	1
4759	62	56	0.10	1	56.67	0.09	1
3241	76	58	0.24	1	65.66	0.14	1
Accumulated:		MMRE = 0.17 MdMRE = 0.10 PRED(25) = 0.72			MMRE = 0.07 MdMRE = 0.06 PRED(25) = 1.00		
Report:							
2740	42	40	0.05	1	41.97	0.00	1
5498	51	52	0.02	1	58.69	0.15	1
5444	64	64	0.00	1	78.66	0.23	1
5455	68	68	0.00	1	95.28	0.40	0
5471	77	77	0.00	1	71.31	0.07	1
5463	84	68	0.19	1	92.29	0.10	1
5466	92	96	0.04	1	94.87	0.03	1
8716	92	92	0.00	1	51.34	0.44	0
8827	98	106	0.08	1	96.57	0.01	1
8171	106	106	0.00	1	96.76	0.09	1
8597	120	120	0.00	1	115.17	0.04	1
8104	144	144	0.00	1	93.89	0.35	0
Accumulated:		MMRE = 0.20 MdMRE = 0.05 PRED(25) = 0.80			MMRE = 0.16 MdMRE = 0.09 PRED(25) = 0.75		
View:							
8449	27	24	0.11	1	28.05	0.04	1
4777	40	40	0.00	1	40.54	0.01	1
3036	50	32	0.36	0	43.16	0.14	1
8469	84	100	0.19	1	78.91	0.06	1
Accumulated:		MMRE = 0.16 MdMRE = 0.16 PRED(25) = 0.67			MMRE = 0.06 MdMRE = 0.05 PRED(25) = 1.00		

6 Conclusion

The aim of the paper was to analyze the applicability of a case-based reasoning estimation technique on effort estimation of user stories and to provide an automated effort estimation tool which could offer reliable support for project planning. On the basis of real-world data, the applicability was approved. The distribution of the user stories from a large software project into multiple small case-bases and the PSO based weights optimization of the local similarity measures was very successful. We think it might play a role that the user stories in the case-bases are homogeneous due to this separation. Moreover, the documentation of the stories is of an outstanding quality. Further, it seems that it is an advantage that the adaptation rule is considered by the objective function during optimization with PSO. In our future work, we are planning to conduct research to investigate this assumption. Concluding, the results show that CBR in combination with an automated environment reveals to be a suitable estimation technique for user story software estimation that can offer a reliable estimation support.

References

1. Alsaadi, B., Saeedi, K.: Data-driven effort estimation techniques of agile user stories: a systematic literature review. Artif. Intell. Rev. 1–32 (2022). https://doi.org/10.1007/s10462-021-10132-x
2. Althoff, K., Roth-Berghofer, T., Bach, K., Sauer, C.: myCBR (2015). http://www.mycbr-project.org/. Accessed 06 May 2022
3. Bilgaiyan, S., Sagnika, S., Mishra, S., Das, M.: A systematic review on software cost estimation in agile software development. J. Eng. Sci. Technol. Rev. (JESTR) **10**(4), 51–64 (2017)
4. Choetkiertikul, M., Dam, H., Trany, T., Phamy, T., Ghose, A., Menzies, T.: A deep learning model for estimating story points. IEEE Trans. Softw. Eng. **45**(7), 637–656 (2016)
5. Cohn, M.: User Stories Applied - For Agile Software Development. Addison-Wesley (2004). iSBN: 978-0-321-20568-1
6. Fernández-Diego, M., Méndez, E.R., González-Ladrón-De-Guevara, F., Abrahão, S., Insfran, E.: An update on effort estimation in agile software development: a systematic literature review. IEEE Access **8**, 166768–166800 (2020)
7. Huang, S.J., Chiu, N.H.: Optimization of analogy weights by genetic algorithm for software effort estimation. Inf. Softw. Technol. **48**, 1034–1045 (2006)
8. Huang, S.J., Chiu, N.H., Chen, L.W.: Integration of the grey relational analysis with genetic algorithm for software effort estimation. Eur. J. Oper. Res. **188**, 898–909 (2008)
9. Jarmulak, J., Craw, S., Rowe, R.: Genetic algorithms to optimise CBR retrieval. In: Blanzieri, E., Portinale, L. (eds.) EWCBR 2000. LNCS, vol. 1898, pp. 136–147. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44527-7_13
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995-International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)

11. Lee, A.: Pyswarm documentation (2014). https://pythonhosted.org/pyswarm/, Accessed 06 May 2022
12. Malgonde, O., Chari, K.: An ensemble-based model for predicting agile software development effort. Empirical Softw. Eng. **24**(2), 1017–1055 (2018). https://doi.org/10.1007/s10664-018-9647-0
13. Marco, R., Suryana, N., Ahmad, S.: A systematic literature review on methods for software effort estimation. J. Theor. Appl. Inf. Technol. **97**(2), 434–464 (2019)
14. Medeiros, J.A.C.C., Schirru, R.: Identification of nuclear power plant transients using the particle swarm optimization algorithm. Ann. Nucl. Energy **35**, 576–582 (2008)
15. Mukhopadhyay, T., Vicinanza, S., Prietula, M.: Examining the feasibility of a case-based reasoning model for software effort estimation. MIS Quart. **16**, 155–171 (1992)
16. Munialo, S.W., Muketha, G.M.: A review of agile software effort estimation methods. Int. J. Comput. Appl. Technol. Res. **5**(9), 612–618 (2016)
17. Nerkar, L.R., Yawalkar, P.M.: Software cost estimation using algorithmic model and non-algorithmic model a review. IJCA Proc. Innovations Trends Comput. Commun. Eng. ITCCE **2**, 4–7 (2014). publisher: Foundation of Computer Science (FCS)
18. Richter, M.M., Weber, R.O.: Case-Based Reasoning. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40167-1
19. Schwaber, K.: Agile Project Management with Scrum. Microsoft Press (2004). iSBN: 978-0-735-61993-7
20. Shepperd, M., Schofield, C.: Estimating software project effort using analogies. IEEE Trans. Softw. Eng. **23**(12), 736–743 (1997)
21. Shepperd, M., Schofield, C., Kitchenham, B.: Effort estimation using analogy. In: Proceedings 18th International Conference Software Engineering, pp. 170–178. IEEE CS Press (1996)
22. Stahl, A., Gabel, T.: Optimizing similarity assessment in case-based reasoning. In: Proceedings of the National Conference on Artificial Intelligence, vol. 21(2), p. 1667. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2006)
23. Suri, P., Ranjan, P.: Comparative analysis of software effort estimation techniques. Int. J. Comput. Appl. (IJCA) **48**(21), 12–19 (2012)
24. Trendowicz, A., Jeffery, R.: Software Project Effort Estimation. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-03629-8
25. Usman, M., Mendes, E., Weidt, F., Britto, R.: Effort estimation in agile software development: a systematic literature review. In: Proceedings of the 10th international conference on predictive models in software engineering, pp. 82–91 (2014)
26. Wiratunga, N., Wijekoon, A., Cooper, K.: Learning to compare with few data for personalised human activity recognition. In: Watson, I., Weber, R. (eds.) ICCBR 2020. LNCS (LNAI), vol. 12311, pp. 3–14. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58342-2_1
27. Wu, D., Li, J., Bao, C.: Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation. Soft. Comput. **22**(16), 5299–5310 (2017). https://doi.org/10.1007/s00500-017-2985-9
28. Wu, D., Li, J., Liang, Y.: Linear combination of multiple case-based reasoning with optimized weight for software effort estimation. J. Supercomput. **64**(3), 898–918 (2013). https://doi.org/10.1007/s11227-010-0525-9