Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Impact of individual competence on agile effort estimation in academic setting

Luka Fürst* | Tomaž Hovelja | Marko Poženel | Damjan Vavpotič

University of Ljubljana, Faculty of Computer and Information Science, Slovenia

Correspondence
*Luka Fürst, University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, SI-1000 Ljubljana, Slovenia. Email: luka.fuerst@fri.uni-lj.si

Abstract

Effort estimation is an important activity in agile software development. The goal of the presented study was to determine the influence of individual competence on software development effort estimation. In particular, we measured both the accuracy of effort estimation and the duration of the estimation process itself, both for three different estimation methods. The subjects of our study were teams of students of a graduate-level software engineering course at the University of Ljubljana, Faculty of Computer and Information Science. Based on the grades that individual students attained in their undergraduate study, we classified each team as 'high-competence' or 'low-competence' and additionally as 'heterogeneous' or 'homogeneous' (the criterion here being the variance of the members' average grades). We found out that there was no significant difference in effort estimation accuracy neither between high-competence and low-competence teams nor between heterogeneous and homogeneous teams, regardless of which estimation method was used. However, high-competence teams spent significantly less time on effort estimation than low-competence ones. Likewise, for two of the employed estimation methods, heterogeneous teams completed effort estimation in a significantly shorter time than homogeneous teams. These results might benefit both academic and professional community.

KEYWORDS:
competence; effort estimation; agile sofware development; student project; software engineering course

1 | INTRODUCTION

Competence is undoubtedly one of the major factors in software development. The ability of a software development team's members to solve a variety of engineering problems, to work towards a common goal within the team, and to meet strict deadlines is crucial for a software project to succeed. Competence does not play a vital role only in the implementation phase of the project; it is no less important in the effort estimation phase, during which the team members try to predict the amount of work and, in turn, the length of time required to complete a given software project.

In this paper, we focus on the relationship between individual team member competence and software effort estimation. In particular, we are interested in the accuracy of effort estimation (i.e., how accurately a team predicts the time required to complete individual functionalities of the project) and the duration of the estimation process itself. We thus carried out a study to measure

both of these dependent variables as a function of competence. The study was conducted at the University of Ljubljana, Faculty of Computer and Information Science, and its subjects were teams of students of a graduate-level software engineering course.

Software effort estimation is a discipline in its own right. Numerous approaches have been proposed in both theory and practice (e.g., [1,2,3]). The approaches differ substantially in the way they are performed, in the aspects of estimation that they emphasize, and in the time they typically take to complete. The student teams that participated in our study employed three estimation methods: Planning Poker [4,5,6,7], Bucket System [8], and Affinity Estimation [9]. While Planning Poker focuses on individual user stories (each user story is estimated individually and independently from the others), Bucket System and Affinity Estimation take a more holistic approach: the team tries to estimate how difficult individual user stories are in relation to the others.

The goal of the study that we present in this paper is to obtain the answers to the following research questions:

RQ1: Does the average competence of the members of a software development team affect the accuracy of effort estimation?

RQ2: Does the variance in competence of team members affect the accuracy of effort estimation?

RQ3: Does the average competence of team members affect the time required to perform effort estimation?

RQ4: Does the variance in competence of team members affect the time required to perform effort estimation?

We will answer each of these questions three times, once for each of the above-mentioned estimation methods. For example, in the context of question RQ1, we will investigate how (if at all) the average competence of team members affects the team's estimation accuracy provided that they employ (a) Planning Poker, (b) Bucket System, or (c) Affinity Estimation.

What, precisely, is the contribution of our paper? First, while there have been (as we show in Section 2.1) studies investigating the relationship between competence and the accuracy of effort estimation, we are not aware of any research work dealing with the *duration* of the estimation process in connection with competence. Second, whereas Planning Poker is a well-established and well-known estimation method, this is far from being the case with Bucket System and Affinity Estimation. As a second contribution, our paper sheds light on these intriguing, yet barely known techniques, and shows that they are perfectly fit to be used for effort estimation.

The rest of this paper is structured as follows. In Section 2, we give an overview of related work and present the estimation methods employed in our study. Following that, we describe the structure and implementation of our study (Section 3) and show its statistically supported results (Section 4). The answers to our research questions are provided and discussed in Section 5. Section 6 brings our paper to a conclusion.

2 | BACKGROUND

In this section, we first give an overview of related work. Subsequently, we present the three estimation methods (Planning Poker, Bucket System, and Affinity Estimation) that we used in our study.

2.1 | Related work

Not all people possess the same level of competence. This claim seems so self-evident that it barely needs to be stated. However, in the area of software development, researchers only relatively recently began to address it. One of the first studies to quantify the differences in competence between the developers working on the same project was carried out by Bryan [10]. By keeping track of the performance of the programmers involved in a large project over several years, he noted that the top programmer (out of 200) resolved as many as 8% of all problems and the top 27% of programmers accomplished 78% of all tasks.

Even more time had to pass before researchers recognized developers' competence as a factor that affects not only bare productivity (i.e., the time required to deliver a certain software artifact) but also the ability to produce a (rough) *estimate* of software effort, i.e., the expected time (in terms of hours or person-hours) to produce an artifact. For example, van Genuchten [11] identified 15 reasons why software projects are often late, but developers' competence was overlooked. Incidentally, as per his study, the most important reason is underestimating the project's complexity. Lederer and Prasad [12], who also posed the question as to why deadlines in software industry are so often overstepped, pointed their fingers to project managers (poor organization). However, Lederer and Prasad did mention competence, albeit indirectly: they listed the managers' 'inability to anticipate skills of project team members' as one of the problems contributing to suboptimal organization.

Basten and Sunyaev[13] were probably among the first to explicitly identify developers' competence as a factor that influences the accuracy of software effort estimates. Lenarduzzi[14] compiled a similar list, although her survey placed 'competence level' at a rather low position (as the 11th most important factor out of 14). Kuan[15] incorporated competence into a linear regression model to predict software effort.

Psychologists had been probably posing questions regarding competence long before the advent of computing revolution, but it was not before 2000 that Kruger and Dunning in their seminal paper[16] analyzed the relationship between a person's competence and their ability to estimate their own competence (which, as we shall see later, is related to the ability to estimate the complexity of tasks in general). Kruger and Dunning discovered what was to become the *Dunning-Kruger effect*: that unskilled persons tend to overestimate their abilities, and the reason for that is what they called 'double curse' — that it is *the very lack of competence* that deprives the unskilled of recognizing their own lack of competence! Incidentally, Kruger and Dunning found out that the skilled, by contrast, tend to (somewhat) *underestimate* their abilities relative to others, but for a different reason — they found the problems that they had to solve as part of the study easy and consequently thought that they should appear easy to other participants as well.

Naturally, there have been challenges to Kruger's and Dunning's breakthrough study. For example, Krajc and Ortmann[17] raised a few questions regarding methodological soundness and offered four alternative explanations for the Dunning-Kruger effect. One of the explanations was that the unskilled find it harder to position themselves relative to others because of the lack of proper feedback (considering that in the era of grade inflation[18] a student barely ever receives negative feedback, it is difficult for him/her to determine where he/she stands). Nevertheless, the main claims stated by Kruger and Dunning remain to be valid to this day, and several issues pointed out by Krajc and Ortmann were resolved in the study of Ehrlinger *et al.*[19].

Simons[20] carried out a similar study on poker players, noting that the unskilled tend to overestimate their abilities *even if they know how good they are in relation to others*, i.e., even if they are, in psychological parlance, *calibrated*. As suggested by Simons, this is due to the will-do-better-this-time effect: many people may attribute their lack of success to bad luck or other circumstances beyond their control and thus naturally hope that these factors will sooner or later turn into their favor. Through a meta-analysis based on a thorough review of studies dealing with a person's ability to assess their own competence, Zell and Krizan[21] found out that 'people have only moderate insight into their abilities', thus confirming the Dunning-Kruger effect.

How are these psychological findings related to software effort estimation? An answer to this question was provided by Jørgensen *et al.*[22]. In their study, the participants were asked first to estimate the time required to complete each of four smaller and four larger software projects individually and then to implement the projects. By measuring, for each participant, the actual time they spent to complete each of the projects, Jørgensen *et al.* came to a conclusion that the lower-skilled do not only underestimate the complexity of larger projects (as predicted by the Dunning-Kruger effect) but also overestimate the complexity of smaller projects, which they termed 'inversed Dunning-Kruger effect'. However, while seemingly inconsistent with the findings of Kruger and Dunning, this is in fact not the case: the lower-skilled, having poor insight into their abilities, simply have trouble grasping the complexity of their task, and this also holds true when the task is in fact easier than it appears at the time it is assigned to them.

One may argue that the accuracy of software effort estimates depends on their type; for instance, that it is harder to predict the amount of work using absolute estimates (in terms of hours or person-hours) than it is with relative estimates (in terms of story points), or vice versa. However, Jørgensen and Escott[23] dispelled all doubts: they found out that relative estimates are no more accurate, no less time-consuming, and no more person-independent than absolute estimates.

To conclude this section, let us mention a few other related studies. Salamea and Farré[24] measured how developer factors (experience, among others) affect the amount of 'technical debt' (bugs, code smells, etc.) introduced into the code. Surprisingly, they found out that the quantity of technical debt is positively correlated with experience. However, they defined experience as the length of time that the developer had been working on the project, rather than as *prior* experience, as we do. Jørgensen[25] showed that the accuracy of estimates depends on how the question asking to provide an estimate is formulated. In particular, it is better to ask the project manager to determine how likely it is for the project to take more (or less) than a given amount of time than to ask them to provide an interval that, with a given percentage of confidence, includes the actual duration of the project.

2.2 | The estimation methods

In this section, we present our three estimation methods of choice in more detail. Although all of them were designed to be used in conjuction with agile software development processes, they are not limited to them; in fact, they could be used for any type of effort estimation. However, all three methods assume that a project to be estimated is divided into well-delineated

functionalities called *user stories*. In the context of software development, a user story typically takes the form 'A [*user role*] can [*a functionality*].' All three methods also assume the existence of a *product owner*, a person who represents the project and is ready to answer any questions that may arise during the estimation process.

Planning Poker[4,6]. First, a set of cards to represent the user stories has to be prepared. The cards are shuffled, arranged into a pile, and placed face-down on a table. After that, each team member receives a set of cards with grades from a Fibonacci-like sequence (e.g., 0.5, 1, 2, 3, 5, 8, 13, 20, 40 ...), where each unit represents a single *story point* (which itself represents 6 hours of uninterrupted work). Following this initial phase, the estimation process runs as follows. In each iteration, a team member picks the next user story card from the pile and turns it around so that it is visible to everyone. Each team member then independently selects a grading card from his/her hand that (in his/her opinion) best represents the difficulty of the story and places the card on the table. If all the members have picked the same grade, they assign that grade to the story and immediately move on to the next user story; otherwise, the member(s) who chose the highest grade and the one(s) who chose the lowest have to justify their decision, and the voting procedure repeats until they reach consensus. The estimation process is finished once all user stories have been graded.

Bucket System[8]. Again, the user stories are represented by cards. The team members first divide the estimation area into buckets labeled by a Fibonacci-like sequence. After that, they pick a user story that is considered neither extremely easy nor extremely hard and place it into the middle bucket. Following that, they pick two more user stories at random and place them relative to the first story. At this point, upon discussion, the team might reposition any subset of these three user stories, which will constitute the 'backbone' for the rest of the estimation process.

In the next phase, each team member receives approximately the same number of the remaining user story cards. Taking turns, the members silently, without any discussion, place their cards relative to the cards that have already been positioned.

In the last phase, any team member may initiate a discussion regarding the placement of a card. The card may be moved only if all members agree with such a step. Once everyone is satisfied with the placement of the cards, the estimation process is finished.

Affinity Estimation[9]. At the beginning, every team member receives approximately the same share of the cards that represent the user stories. The team members, taking turns, then silently place their cards on the table. The x-coordinate of a card indicates its relative difficulty: the cards to the left are considered easier than those to the right. (In contrast to Bucket System, the placement of the cards is not constrained by buckets.) If a member feels unable to place the current card, he/she may place it on a designated pile away from the voting area.

In the next phase, the members may reposition some cards if they feel so. However, before making a change, the member has to discuss it with the other members. Once the process stabilizes, the cards are placed into buckets that represent story points. The cards themselves are not moved; rather, borders are drawn around them to delineate the buckets.

In the last phase, the members may select a set of cards and discuss each of them separately. If necessary, a card may be moved to a different bucket. This phase ends once all the cards in the set have been discussed.

In summary, Planning Poker emphasizes the individual difficulties of the user stories, while Bucket System and Affinity Estimation focus predominantly on the stories' relative difficulties. In Planning Poker, the team spends most of the time estimating and discussing the effort required to complete each individual user story. In the other two methods, however, most of the time is devoted to positioning the individual user stories relative to the others; their absolute difficulties are less important.

3 | METHODOLOGY

Our study was conducted on the students of a graduate-level software engineering course at the University of Ljubljana, Faculty of Computer and Information Science. This course has been characterized by project-based learning[26,27,28] ever since its inception. Every year, the students form small teams and complete a software project using agile software development methods[29,30]. Our study did not disrupt the learning process in any way; the students worked on a software development project in the same manner as their predecessors in the years prior to our study. Since effort estimation has always been considered an indispensable part of our projects, the study placed minimal additional burden to the students' regular work. In particular, they were asked to perform effort estimation using three different methods rather than just one, to record the duration of the estimation process for

each method separately, and to keep track of the actual effort (i.e., time) required to complete individual functionalities (*user stories*) of the project.

Let us now describe our study in more detail. The students were grouped in estimation teams of three or four members. Each team had the same assignment, i.e., to develop a part of a web-based application for project management which consisted of 24 different functionalities (e.g., adding new users, adding new projects, creating a new sprint, user stories confirmation, maintaining user accounts, completing tasks, etc.). The teams followed the Scrumban development approach to monitor and direct their effort[31,32]. The teams worked on a project divided into four iterations that together lasted 14 weeks. All iterations included user story effort estimation, decomposition of user stories into tasks, work on tasks, daily stand-ups, and an iteration review. Before starting the first iteration, the teams received the descriptions of the application's functionalities in the form of user stories. To enable the students to experience different team environments, the students changed teams before the start of each iteration.

We used the average of all grades that the student obtained during his/her study on the study program to assess his/her competence level. To provide a solid overview of student proficiency in computer science, the study considered only those students who received at least five grades, i.e., 27 students. Based on these students' average grades, we calculated the average and the variance of grades of team members for each team.

At the beginning of each iteration, the teams estimated the development effort for all 24 user stories and then started to develop six selected user stories. The actual development time of each team for the six user stories was recorded to determine the estimations' accuracy using the balanced measure of relative error (BRE)[33]:

$$\mathrm{BRE} = \frac{|\text{actual effort} - \text{estimated effort}|}{\min(\text{actual effort},\ \text{estimated effort})}. \tag{1}$$

BRE evenly balances overestimation and underestimation and is used in many comparable studies[33,34,35]. Additionally, the time that each team needed to perform effort estimation (EET) was recorded.

Together, for six user stories, seven teams, and four iterations, we obtained 168 evaluations of BRE and EET for each estimation method. We divided these evaluations into two equal groups (each having 84 members) based on median average grade of teams (median = 8.242), namely low-competence teams group (LCG) and high-competence teams group (HCG). We compared the two groups to detect any significant differences in BRE and EET for the three estimation methods. Additionally, we divided BRE and EET evaluations into two different equal groups (again each having 84 members) based on median variance in average grade of teams (median = 0.335), namely homogeneous competence teams (HOCG) and heterogeneous competence teams (HECG). Again, we compared the two groups to detect any significant differences in BRE and EET for the three estimation methods.

As the kurtosis and skewness of BRE and EET indicated a non-normal distribution, we used the non-parametric Mann-Whitney test for both comparisons (low- and high-competence teams and homogeneous and heterogeneous competence teams).

4 | RESULTS

The main results are presented in Tables 1–8. Tables 1–4 show the key descriptive statistics regarding BRE and EET for low- and high-competence team groups and for homogeneous and heterogeneous competence team groups. These four tables provide descriptive statistics separately for each of the three evaluated estimation methods, namely Planning Poker (PP), Bucket System (BS), and Affinity Estimation (AE). The statistics are provided for each group by estimation method and include group size (N), mean, median, standard deviation, variance, skewness, standard error of skewness, kurtosis, standard error of kurtosis, and range.

Tables 5–8 show the results of the Mann-Whitney U-test regarding BRE and EET for low- and high-competence team groups and for homogeneous and heterogeneous competence team groups. The results are provided separately for each estimation method. The tables' columns show the following values: group size (N), mean rank (MR), sum of ranks (SR), Mann-Whitney U (U), Wilcoxon W (W), Z score (Z), and 2-tailed P-value (P). Significant differences between two specific groups are indicated by p-values of 0.05 or smaller.

The results show that, regardless of the employed estimation method, there is no significant difference in estimations' accuracy (BRE) either between low- and high-competence teams or between heterogeneous and homogeneous teams. However, there are significant differences in effort estimation time (EET) between low- and high-competence teams (for all three estimation methods) and between heterogeneous and homogeneous teams (for Bucket System and Affinity Estimation). In particular, for all

Table 1 Descriptive statistics regarding BRE — estimation accuracy as balanced measure of relative error

	LCG[a]			HCG[b]		
	PP	BS	AE	PP	BS	AE
N	84	84	84	84	84	84
Mean	1.29	1.75	1.71	1.06	1.20	1.34
Median	0.68	0.84	1.02	0.67	0.71	0.80
Std. deviation	1.70	2.22	1.95	1.26	1.65	1.52
Variance	2.88	4.91	3.80	1.58	2.72	2.30
Skewness	3.11	2.32	2.25	2.41	3.61	2.35
Std. error of skewness	0.26	0.26	0.26	0.26	0.26	0.26
Kurtosis	13.33	6.30	6.32	6.82	17.29	5.92
Std. error of kurtosis	0.52	0.52	0.52	0.52	0.52	0.52
Range	11.00	11.00	11.00	7.00	11.00	7.14

[a]Low-competence teams group (team average grade < 8.242)
[b]High-competence teams group (team average grade ≥ 8.242)

Table 2 Descriptive statistics regarding BRE — estimation accuracy as balanced measure of relative error

	HOCG[a]			HECG[b]		
	PP	BS	AE	PP	BS	AE
N	84	84	84	84	84	84
Mean	1.17	1.71	1.59	1.18	1.24	1.46
Median	0.50	0.92	0.94	0.71	0.69	0.84
Std. deviation	1.47	2.15	1.87	1.52	1.75	1.63
Variance	2.16	4.62	3.49	2.32	3.05	2.67
Skewness	2.19	2.52	2.52	3.81	3.21	2.09
Std. error of skewness	0.26	0.26	0.26	0.26	0.26	0.26
Kurtosis	5.14	7.57	8.25	20.62	13.38	4.10
Std. error of kurtosis	0.52	0.52	0.52	0.52	0.52	0.52
Range	7.00	11.00	11.00	11.00	10.98	7.58

[a]Homogeneous competence teams group (variance in team average grade < 0.335)
[b]Heterogeneous competence teams group (variance in team average grade ≥ 0.335)

three estimation methods, high-competence teams needed significantly less time to perform effort estimation (EET) than low-competence teams, while there was no significant difference in accuracy (BRE). Similarly, heterogeneous competence teams needed significantly less time to perform effort estimation (EET) than homogeneous competence teams without significant difference in accuracy (BRE), but only for Bucket System and Affinity Estimation, while there was no significant difference in EET for Planning Poker.

5 | DISCUSSION

As can be inferred from the statistical analysis presented in Section 4, the high-competence teams produced neither significantly better nor significantly worse effort estimates than the low-competence ones, regardless of which estimation technique was employed. In other words, referring to RQ1 (Section 1), competence does not affect the accuracy of effort estimation in a statistically

Table 3 Descriptive statistics regarding EET — the time (in minutes) that each team needed to perform effort estimation

	LCG[a]			HCG[b]		
	PP	BS	AE	PP	BS	AE
N	84	84	84	84	84	84
Mean	29.00	19.14	19.36	26.50	15.50	13.71
Median	27.00	17.50	16.50	21.50	16.50	13.50
Std. deviation	11.86	7.13	8.68	14.89	5.41	6.42
Variance	140.68	50.87	75.27	221.60	29.31	41.27
Skewness	1.35	0.27	1.10	2.01	0.15	0.42
Std. error of skewness	0.26	0.26	0.26	0.26	0.26	0.26
Kurtosis	2.00	−1.04	0.67	3.91	−1.23	−0.65
Std. error of kurtosis	0.52	0.52	0.52	0.52	0.52	0.52
Range	47.00	23.00	33.00	61.00	17.00	23.00

[a]Low-competence teams group (team average grade < 8.242)
[b]High-competence teams group (team average grade ≥ 8.242)

Table 4 Descriptive statistics regarding EET — the time (in minutes) that each team needed to perform effort estimation

	HOCG[a]			HECG[b]		
	PP	BS	AE	PP	BS	AE
N	84	84	84	84	84	84
Mean	28.50	18.29	19.57	27.00	16.36	13.50
Median	25.00	17.50	19.00	25.00	14.50	14.50
Std. deviation	15.26	6.07	9.29	11.46	6.94	5.27
Variance	232.88	36.79	86.27	131.42	48.23	27.72
Skewness	1.53	0.62	0.57	1.92	0.42	0.82
Std. error of skewness	0.26	0.26	0.26	0.26	0.26	0.26
Kurtosis	2.34	−0.26	0.19	3.95	−0.99	0.91
Std. error of kurtosis	0.52	0.52	0.52	0.52	0.52	0.52
Range	61.00	21.00	36.00	48.00	23.00	21.00

[a]Homogeneous competence teams group (variance in team average grade < 0.335)
[b]Heterogeneous competence teams group (variance in team average grade ≥ 0.335)

significant way. A similar conclusion can be drawn when considering variance in competence (RQ2): the homogeneous teams did not fare significantly better or worse than the heterogeneous ones, and this, again, holds for all three estimation methods.

These results indicate that all three estimation methods employed in our research, i.e., Planning Poker, Bucket System, and Affinity Estimation, are robust to deviations in competence. In this respect, all three methods are equivalent. However, they are not equivalent in terms of time demands: Bucket System and Affinity Estimation require considerably less time than Planning Poker. Based on these results, we may therefore recommend that software development teams pick either Bucket System or Affinity Estimation as their method of choice. Considering the fact that these two methods, in contrast to Planning Poker, seem to be virtually unknown (judging by the scarcity of papers and other credible resources), this conclusion came to us as a true surprise.

Naturally, robustness to (in)competence cannot be stretched indefinitely. One cannot expect that a team composed of complete novices and a team with years of experience on demanding projects will produce effort estimates of comparable accuracy. After all, the subjects of our study were first-year graduate (MSc) students, all of whom had at least some exposure to programming,

Table 5 Mann-Whitney U-test (BRE — estimation accuracy as balanced measure of relative error)

	N	MR[a]	SR[b]	U[c]	W[d]	Z	P[e]
Planning Poker				3398	6968	−0.413	0.680
LCG[f]	84	86.05	7228				
HCG[g]	84	82.95	6968				
Total	168						
Bucket System				3043	6613	−1.540	0.123
LCG	84	90.28	7584				
HCG	84	78.72	6613				
Total	168						
Affinity Estimation				3191	6761	−1.071	0.284
LCG	84	88.52	7436				
HCG	84	80.48	6761				
Total	168						

[a]Mean rank
[b]Sum of ranks
[c]Mann-Whitney U
[d]Wilcoxon W
[e]P-value (2-tailed)
[f]Low-competence teams group (team average grade < 8.242)
[g]High-competence teams group (team average grade ≥ 8.242)

many of whom had also completed a similar project as part of the undergraduate study, but hardly anyone could claim proficiency on the level of a senior developer in a major software enterprise. Nevertheless, the difference between the most competent subjects and the least competent ones was still readily discernible.

While we observed no significant differences between the high-competence and low-competence teams in terms of estimation accuracy, this was not the case with the time required to perform estimation: the high-competence teams spent significantly less time than the low-competence ones (RQ3). This may be explained by the Dunning-Kruger effect (and its inverse counterpart, as introduced by Jørgensen *et al.*[22]), i.e., that the low-competent tend to lack insight into their capabilities. As a consequence, some may overestimate themselves and consider a particular user story to be easier than it is, given their actual capabilities, while others may underestimate their own capabilities and vote for a grade that is too high. All three methods are designed in such a way that positive and negative deviations tend to level out, so a low-competence team is, ultimately, still likely to arrive at estimates that roughly match their actual capabilities. However, in the presence of such oscillations, the estimation process needs more time to stabilize. By contrast, the members of a high-competence team tend to know their capabilities better and hence reach consensus more rapidly.

Our last research question (RQ4) also refers to the duration of the estimation process, but in connection with variance in competence rather than competence itself. As shown by the statistical analysis, the teams labeled as heterogeneous spent significantly less time than those labeled as homogeneous when the user stories were estimated using Bucket System or Affinity Estimation, but no statistically significant differences were observed in the case of Planning Poker. Further research is probably needed to explain this phenomenon adequately, but some 'educated guesses' can still be made. First, let us consider how Planning Poker is structured. After each round of voting, the member(s) who picked the highest estimate and the one(s) who picked the lowest have to justify their decision. Since it is somewhat unlikely that all team members cast the same vote on the first round (and thus immediately proceed to the next story), the process of estimating a user story will typically consist of at least two rounds. However, after the 'outlying' members have had their say at the end of round 1, it is quite possible that in the second round the team will have reached consensus. The third round might sometimes also be necessary, but anything more is very unlikely. This discussion implies that Planning Poker, with its more rigid structure and a high degree of compulsory conversation, might be less sensitive to intra-team diversity than Bucket System or Affinity Estimation.

Another point to consider in our debate on homogeneous and heterogeneous teams and their estimation efficiency is the internal structure of those teams. In homogeneous teams, the members are approximately equivalent. No member is likely

Table 6 Mann-Whitney U-test (EET — the time (in minutes) that each team needed to perform effort estimation)

	N	MR[a]	SR[b]	U[c]	W[d]	Z	P[e]
Planning Poker				2826	6396	−2.233	0.026
LCG[f]	84	92.86	7800				
HCG[g]	84	76.14	6396				
Total	168						
Bucket System				2610	6180	−2.921	0.003
LCG	84	95.43	8016				
HCG	84	73.57	6180				
Total	168						
Affinity Estimation				2376	5946	−3.666	0.000
LCG	84	98.21	8250				
HCG	84	70.79	5946				
Total	168						

[a]Mean rank
[b]Sum of ranks
[c]Mann-Whitney U
[d]Wilcoxon W
[e]P-value (2-tailed)
[f]Low-competence teams group (team average grade < 8.242)
[g]High-competence teams group (team average grade ≥ 8.242)

to stand out, either in a positive or in a negative direction. However, in heterogeneous teams, such members are much more likely. It is thus reasonable to assume that a certain percentage of heterogeneous teams has a member who is distinctly more competent than the other three. Such a member may be readily perceived as the true leader by the rest of the team and his/her opinions automatically gain on importance. The rest of the team then tends to adapt to the leader's decision, resulting in a quicker consensus. Again, things are somewhat different in Planning Poker, which requires every 'outyling' member to justify his/her decision, regardless of his/her status in the team.

Is our study consistent with the findings of Jørgensen *et al.* and, in turn, Kruger and Dunning? In contrast to Jørgensen *et al.*, who observed that the low-competent underestimated larger projects and overestimated smaller ones, we did not find a significant difference in estimation accuracy between the high-competence and low-competence teams. However, we did discover that the high-competence teams performed the estimation process significantly faster than the low-competence ones. This can be at least partially attributed to the Dunning-Kruger effect: the members of the high-competence teams tend to have a better idea about the complexity of a given user story and hence their grades tend to be close to each other; as a result, the estimation process quickly converges. On the other hand, the low-competent tend to assign a much wider range of grades to a user story. Consequently, the estimation process takes a longer time to converge.

Software development projects similar to the one employed in our study are a common means to acquaint computer science students with situations that await them in their future professional life. For this reason, the results presented in this paper might benefit not only computing professionals but also teachers and students. From our study, we might infer that teams with a larger variance in competence (i.e., heterogeneous teams) are preferable over those having a smaller variance, as they perform no worse in terms of estimation accuracy but tend to be more efficient in the estimation process itself. One might also argue that being part of a heterogeneous team benefits lower-competence members, as their position in the team encourages them to catch up with their higher-competence counterparts. In addition, involvement in a heterogeneous team enhances social skills, both for low-competence and high-competence members[36].

Table 7 Mann-Whitney U-test (BRE — estimation accuracy as balanced measure of relative error)

	N	MR[a]	SR[b]	U[c]	W[d]	Z	P[e]
Planning Poker				3262	6832	−0.844	0.399
HOCG[f]	84	81.33	6832				
HECG[g]	84	87.67	7364				
Total	168						
Bucket System				2994	6564	−1.694	0.09
HOCG	84	90.86	7632				
HECG	84	78.14	6564				
Total	168						
Affinity Estimation				3487	7057	−0.132	0.895
HOCG	84	84.99	7140				
HECG	84	84.01	7057				
Total	168						

[a]Mean rank
[b]Sum of ranks
[c]Mann-Whitney U
[d]Wilcoxon W
[e]P-value (2-tailed)
[f]Homogeneous competence teams group (variance in team average grade < 0.335)
[g]Heterogeneous competence teams group (variance in team average grade ≥ 0.335)

6 | CONCLUSION

We carried out a study with the purpose of determining the influence of individual competence on the accuracy of estimating a software project and the time required to perform estimation itself. The subjects of our study were teams of three to four students attending a graduate-level software engineering course at the Faculty of Computer and Infomation Science at the University of Ljubljana. Each team independently used three methods (Planning Poker, Bucket System, and Affinity Estimation) to estimate the time required to complete individual user stories of a web application project. Based on the average grade that the team members received in ther undergraduate study, each team was classified either as a 'high-competence' team or as a 'low-competence' team. Besides that, the teams were categorized as 'homogeneous' or 'heterogeneous', depending on the variance in competence of their members.

We discovered that there was no significant difference in the accuracy of effort estimation between the high-competence and low-competence teams; both categories of teams achieved a similar quality of effort prediction, regardless of which estimation method was in effect. Likewise, there was no significant difference in estimation accuracy between the homogeneous and heterogeneous teams. In view of these results, all three estimation methods may be regarded as equivalent in terms of robustness to competence and deviations in competence. However, considering the fact that Bucket System and Affinity Estimation require substantially less time to perform than Planning Poker, one may choose to prefer the former two to the latter.

In terms of the time required by the estimation process, the high-competence teams were significantly more efficient than the low-competence ones. Furthermore, heterogeneous teams spent less time than the homogeneous ones when using Bucket System or Affinity Estimation; for Planning Poker, no significant difference was observed. The first result can be at least partially attributed to the Dunning-Kruger effect: the fact that the low-competent tend to have a worse insight into their capabilities leads to more fluctuation during the estimation process, which, in turn, takes a longer time to complete. To adequately explain the difference in estimation efficiency between the homogeneous and heterogeneous teams (for Bucket System and Affinity Estimation) requires further research, but a possible factor might be the existence of a (perceived) 'team leader' in heterogeneous teams. The rest of the team might (unconsciously) adapt to such a member, reducing the amount of communication and hence the duration of the estimation process.

In our study, the competence of a team member and, in turn, a team as a whole was determined by academic performance. In an academic environment, this factor is undoubtedly important, and its additional advantage is that it can be readily obtained from

Table 8 Mann-Whitney U-test (EET — the time (in minutes) that each team needed to perform effort estimation)

	N	MR[a]	SR[b]	U[c]	W[d]	Z	P[e]
Planning Poker				3438	7008	−0.286	0.775
HOCG[f]	84	85.57	7188				
HECG[g]	84	83.43	7008				
Total	168						
Bucket System				2898	6468	−2.004	0.045
HOCG	84	92.00	7728				
HECG	84	77.00	6468				
Total	168						
Affinity Estimation				1800	5370	−5.499	0.000
HOCG	84	105.07	8826				
HECG	84	63.93	5370				
Total	168						

[a]Mean rank
[b]Sum of ranks
[c]Mann-Whitney U
[d]Wilcoxon W
[e]P-value (2-tailed)
[f]Homogeneous competence teams group (variance in team average grade < 0.335)
[g]Heterogeneous competence teams group (variance in team average grade ≥ 0.335)

students' transcripts. However, there are other, harder-to-measure factors that may also affect effort estimation but do not necessarily correlate with academic performance[37,14,38]: social skills, domain-specific knowledge, experience with similar projects, etc. An interesting direction for future research would thus be to investigate the influence of different types of competence on effort estimation.

ACKNOWLEDGMENTS

This study was supported by the Slovenian Research Agency (Javna agencija za raziskovalno dejavnost Republike Slovenije, ARRS), Grant Number P2-0426, Program Group Digital Transformation for Smart Public Governance.

References

1. Mahmood Y, Kama N, Azmi A, Khan AS, Ali M. Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Software: Practice and Experience* 2022; 52(1): 39–65. doi: 10.1002/spe.3009

2. Karna H, Vicković L, Gotovac S. Application of data mining methods for effort estimation of software projects. *Software: Practice and experience* 2019; 49(2): 171-191. doi: 10.1002/spe.2651

3. Sudarmaningtyas P, Mohamed R. A Review Article on Software Effort Estimation in Agile Methodology. *Pertanika Journal of Science and Technology* 2021; 29(2): 837–861. doi: 10.47836/pjst.29.2.08

4. Cohn M. *Agile Estimating and Planning*. Pearson . 2005.

5. Lopez-Martinez J, Ramirez-Noriega A, Juarez-Ramirez R, Licea G, Martinez-Ramirez Y. Analysis of planning poker factors between university and enterprise. In: *5th International Conference in Software Engineering Research and Innovation (CONISOFT)*; 2017: 54–60.

6. Poženel M, Hovelja T. A comparison of the planning poker and team estimation game: a case study in software development capstone project course. *International journal of engineering education* 2019; 35(1): 195–208.

7. Raith F, Richter I, Lindermeier R, Klinker G. Identification of inaccurate effort estimates in agile software development. In: *20th Asia-Pacific Software Engineering Conference (APSEC)*; 2013: 67–72.

8. Mallidi RK, Sharma M. Study on Agile Story Point Estimation Techniques and Challenges. *International Journal of Computer Applications* 2021; 174(13): 9–14.

9. Woolf M. Why Affinity Estimation?.; 2014. accessed September 26, 2022.

10. Bryan G. Not all programmers are created equal. In: *IEEE Conference on Aerospace Applications*; 1994: 55–62.

11. Genuchten vM. Why is Software Late? An Empirical Study of Reasons For Delay in Software Development. *IEEE Transactions on Software Engineering* 1991; 17(6): 582–590. doi: 10.1109/32.87283

12. Lederer AL, Prasad J. Causes of inaccurate software development cost estimates. *Journal of Systems and Software* 1995; 31(2): 125–134. doi: 10.1016/0164-1212(94)00092-2

13. Basten D, Sunyaev A. A Systematic Mapping of Factors Affecting Accuracy of Software Development Effort Estimation. *Communications of the Association for Information Systems* 2014; 34: 4.

14. Lenarduzzi V. Could social factors influence the effort software estimation?. In: Hammouda I, Sillitti A. , eds. *7th International Workshop on Social Software Engineering, SSE 2015, Bergamo, Italy, September 1, 2015*ACM; 2015: 21–24

15. Kuan SWI. Factors on software effort estimation. *International Journal of Software Engineering & Applications* 2017; 8(1). doi: 10.5121/ijsea.2017.8103

16. Kruger J, Dunning D. Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments. *Journal of Personality and Social Psychology* 2000; 77(6): 1121–1134.

17. Krajc M, Ortmann A. Are the unskilled really that unaware? An alternative explanation. *Journal of Economic Psychology* 2008; 29: 724–738.

18. Johnson VE. *Grade inflation: A crisis in college education*. New York: Springer . 2003.

19. Ehrlinger J, Johnson K, Banner M, Dunning D, Kruger J. Why the Unskilled Are Unaware: Further Explorations of (Absent) Self-Insight Among the Incompetent. *Organizational Behavior and Human Decision Processes* 2008; 105(1): 98–121. doi: 10.1016/j.obhdp.2007.05.002

20. Simons DJ. Unskilled and optimistic: Overconfident predictions despite calibrated knowledge of relative skill. *Psychonomic Bulletin and Review* 2013; 20: 601–607.

21. Zell E, Krizan Z. Do People Have Insight Into Their Abilities? A Metasynthesis. *Perspectives on Psychological Science* 2014; 9(2): 111–125. doi: 10.1177/1745691613518075

22. Jørgensen M, Bergersen GR, Liestøl K. Relations Between Effort Estimates, Skill Indicators, and Measured Programming Skill. *IEEE Transactions on Software Engineering* 2021; 47(12): 2892–2906. doi: 10.1109/TSE.2020.2973638

23. Jørgensen M, Escott E. Relative estimates of software development effort: Are they more accurate or less time-consuming to produce than absolute estimates, and to what extent are they person-independent?. *Information and Software Technology* 2022; 143: 106782. doi: 10.1016/j.infsof.2021.106782

24. Salamea MJ, Farré C. Influence of Developer Factors on Code Quality: A Data Study. In: *19th IEEE International Conference on Software Quality, Reliability and Security Companion, QRS Companion 2019, Sofia, Bulgaria, July 22-26, 2019*IEEE; 2019: 120–125

25. Jørgensen M. Realism in Assessment of Effort Estimation Uncertainty: It Matters How You Ask. *IEEE Transactions on Software Engineering* 2004; 30(4): 209–217. doi: 10.1109/TSE.2004.1274041

26. Salankar N, Koundal D, Hu YC. Impact on the personality of engineering students based on project-based learning. *Computer Applications in Engineering Education* 2021; 29(6): 1602–1616.

27. Letouze P, Souza Júnior dJ, Silva dVM. Generating Software Engineers by Developing Web Systems: A Project-Based Learning Case Study. In: *IEEE 29th International Conference on Software Engineering Education and Training*; 2016: 194–203.

28. Gomez-Jaramillo S, Moreno-Cadavid J, Zapata-Jaramillo C. Agile project based learning applied in a software development course. In: *12th International Conference of Education, Research and Innovation*; 2019: 8089–8095.

29. Çulha D. Applying competition-based learning to agile software engineering. *Computer Applications in Engineering Education* 2016; 24(3): 382–387.

30. Martin RC. *Agile Software Development: Principles, Patterns, and Practices*. Pearson . 2002.

31. Reddy A. *The Scrumban [R]Evolution: Getting the Most Out of Agile, Scrum, and Lean Kanban*. Addison-Wesley . 2015.

32. Brezočnik L, Majer v. Comparison of agile methods: Scrum, Kanban, and Scrumban. In: *Proceedings of the 19th International Multiconference Information Society – IS 2016: Collaboration, software and services in information society*; 2016: 30–33.

33. Cao L. Estimating Efforts for Various Activities in Agile Software Development: An Empirical Study. *IEEE Access* 2022; 10: 83311–83321. doi: 10.1109/ACCESS.2022.3196923

34. Seo Y, Bae D. On the value of outlier elimination on software effort estimation research. *Empirical Software Engineering* 2013; 18(4): 659–698. doi: 10.1007/s10664-012-9207-y

35. Moløkken-Østvold K, Haugen NC, Benestad HC. Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software* 2008; 81(12): 2106–2117. doi: 10.1016/j.jss.2008.03.058

36. Antonetti M, Rufini A. Social norms, coordination and collaboration in heterogeneous teams. *Managerial and Decision Economics* 2008; 29(7): 547–554.

37. Notari M, Baumgartner A, Herzog W. Social skills as predictors of communication, performance and quality of collaboration in project-based learning. *Journal of Computer Assisted Learning* 2014; 30(2): 132–147.

38. Morgenshtern O, Raz T, Dvir D. Factors affecting duration and effort estimation errors in software development projects. *Information & Software Technology* 2007; 49(8): 827–837.