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?

*Magne Jørgensen[[1]](#footnote-1), Simula Metropolitan & Oslo Metropolitan University, Norway*

*Eban Escott, Codebots, Brisbane, Australia*

**Abstract**

***Context****: Estimates of software development effort may be given as judgments of relationships between the use of efforts on different tasks – that is, as relative estimates. The use of relative estimates has increased with the introduction of story points in agile software development contexts.*

***Objective****: This study examines to what extent relative estimates are likely to be more accurate or less time-consuming to produce than absolute software development effort estimates and to what extent relative estimates can be considered developer-independent.*

***Method****: We conducted two experiments. In the first experiment, we collected estimates from 102 professional software developers estimating the same tasks and randomly allocated to providing relative estimates in story points or absolute estimates in work-hours. In the second experiment, we collected the actual efforts of 20 professional software developers completing the same 5 programming tasks and used these to analyse the variance in relative efforts.*

***Results****: The results from the first experiment indicates that the relative estimates were less accurate than the absolute estimates, and that the time consumed completing the estimation work was higher for those using relative estimation, even when only considering developers with extensive prior experience in story point–based estimation for both tasks. The second experiment revealed that the relative effort was far from developer-independent, especially for the least productive developers. This suggests that relative estimates to a large extent are developer-dependent.*

***Conclusions****: Although there may be good reasons for the continued use of relative estimates, we interpret our results as not supporting that the use of relative estimates is connected with higher estimation accuracy or less time consumed on producing the estimates. Neither do our results support a high degree of developer-independence in relative estimates.*

**Keywords**: Software development effort estimation; expert estimates; relative estimation; story points

# Introduction

Relative estimation is the process of comparing the characteristics of one object to those of one or more other objects with the purpose of producing a judgment about a relationship between them. We may, for example, estimate the work effort required to complete Task A to be about twice as high as that required to complete Task B. This is possible without using any unit of time, such as work-hour or man-month. The use of relative estimation in software development has increased with the introduction of story points in agile software development contexts [1]. Many organisations that earlier estimated the effort of software development tasks in work-hours or some other unit of time now base their estimates on judgments of how large, in story points, one task is compared with another. How to estimate in story point and transform these relative estimates into delivery plans through productivity factors (velocity) is described in, amongst others, [2].

The research presented in this paper aims at improving our knowledge about relative (as exemplified by story points–based) estimates compared with absolute (as exemplified by work-hour–based) estimates of software development effort. We do this by testing the following three hypotheses related to the potential benefits of using relative estimation:

* Hypothesis 1: Relative estimates are more *accurate* than absolute estimates of software development effort.
* Hypothesis 2: Relative estimation is *less time-consuming* than absolute estimation of software development effort.
* Hypothesis 3: Relative effort estimates are to a large extent *developer-independent* – that is, the ratios of efforts spent on two tasks tend to be similar for different software developers.

Previous research comparing the *accuracy* of relative and absolute estimates seems to be sparse, and we were able to identify only two studies with direct comparisons of relative and absolute effort estimates [3, 4]. These studies use the correlation between the estimated and actual effort of software development tasks as their accuracy measure. Both studies report that the estimates in story points (the relative estimates) were better correlated with the actual effort than the estimates in work-hours (the absolute estimates). These results provide evidence in support of Hypothesis 1. This support is, however, not very strong, as none of the studies compared relative and absolute effort estimates for the same tasks and with the same developers. Furthermore, the correlation between the estimated and actual values is not a robust measure of estimation accuracy, as pointed out in [5].

We were unable to identify any software estimation studies comparing the *time* spent on relative and absolute estimation. Studies on relative estimation of judgment tasks other than effort estimation, such as those used in psychophysics[[2]](#footnote-2), suggest that absolute and relative estimation have many similarities in the underlying judgement processes [6]. In particular, several of these studies suggest that in reality, the majority of estimates are relative – that is, even absolute estimation includes a comparison of the target object with one or more reference objects [7]. If there is a similarity in the underlying estimation processes of software development effort and those studied in psychophysics, we may see similarity in time spent on relative and absolute software development effort estimation, and no support for Hypothesis 2.

Prior research thoroughly documents that software developers substantially vary in productivity, for example that there are large between-developer differences in effort spent to complete the same task [8, 9]. The realism of an absolute estimate of the effort needed to complete a software development task is consequently often far from person-independent. This observation motivates the statement ‘your hour is not the same as my hour’, which is sometimes used to argue against the use of absolute estimates and motivate the use of relative estimation of effort.[[3]](#footnote-3) Whereas the person-dependency of absolute effort estimates has been thoroughly documented, the same cannot be said to be the case for the person-independence of relative estimates. This means that we know little about to what extent the statement ‘your story point is not the same as my story point’ is more or less valid than the statement ‘your hour is not the same as my hour’. We were able to identify only one study on the developer-independency of relative effort estimates [10]. This study found that the relative effort substantially varied among developers.

The remaining part of the paper is organised as follows: Section 2 reports on the first experiment and aims at testing Hypotheses 1 and 2. Section 3 reports on the second experiment and examines Hypothesis 3. Section 4 discusses the results and concludes the study.

# Experiment 1: Are relative estimates more accurate and less time-consuming?

## 2.1. Study design

*Recruitment*: An invitation to participate in a study on the estimation of software development effort was sent to three software development companies in Australia, one company in UK, one international company and one company in Norway. The companies were selected based on the authors' professional networks and the participants on their willingness to participate in the estimation experiment and an assessment that they had the required set of estimation and development skills. All invited participants received a link to the tasks to be completed using the survey tool Qualtrics ([www.qualtrics.com](http://www.qualtrics.com)). The participants could start the estimation work at their convenience. They were instructed not to discuss with others about the estimation tasks, their estimates or the systems to be estimated during or after their estimation work. The estimation work included two estimation tasks, described later in this section, and requested only rough (ballpark) estimates.

*Responses*: We received 109 complete estimation responses from the around 200 invitations.[[4]](#footnote-4) Two responses were removed owing to the lack of estimation experience of the respondents, two responses were removed owing to the respondents not spending sufficient time on the responses (defined as spending in total less than 5 min on the estimation tasks), and three responses were removed owing to misunderstandings of the estimation instructions and producing absolute estimation when relative estimation was required. After this removal, we had 102 responses with meaningful estimates. Table 1 displays information about the 102 participants completing the estimation work. As can be seen in Table 1, all participants reported experience in estimating the effort of their own software development work, and most participants reported experience with relative estimation. The participants’ mean length of experience as software developer was 10.2 years. All developers were assessed, by those inviting them to participate, to have experience with software development tasks similar to those used in our experiment.

**Table 1: The participants**

|  |  |  |
| --- | --- | --- |
| **Characteristic** | **Categories** | **Proportion** |
| Role | Developers | 66% |
| Managers1 | 34% |
| Gender | Male | 79% |
| Female | 21% |
| Development skill (self-assessed) | Very good | 21% |
| Good | 51% |
| Average | 17% |
| Acceptable | 7% |
| Poor | 4% |
| Times estimating own work | More than 50 | 56% |
| 20–50 | 22% |
| 10–19 | 15% |
| 1–9 | 7% |
| Never | 0% |
| Times using relative estimation | More than 50 | 35% |
| 20–50 | 22% |
| 10–19 | 10% |
| 1–9 | 20% |
| Never | 13% |

1: Those in management roles were scrum masters, team leaders, project managers or general managers.

*The two estimation tasks:*The estimation work of the participants started with the estimation of the effort required to develop software that met the requirements specified through the description of five user stories (*Estimation task 1*). These five user stories describe the functionality of a software system with information about research studies. The developers were randomly allocated to use relative estimation in story points or absolute estimation in work-hours. The developers were instructed to base the estimates on the assumption that they did all the work themselves and used the technology they knew best. Those allocated to provide relative estimates first selected a reference user story, gave that user story ten story points, and gave the other four user stories story points representing the relative effort compared with the reference story. To enable a transfer from relative to absolute estimates, we requested from those giving relative estimates an absolute estimate of the reference user story in work-hours. This absolute estimate in work-hours of the reference story was provided after the completion of the submission of the relative estimates. Those allocated to provide absolute estimates gave for each of the user stories the estimated numbers of work-hours. The software application estimated by the participants had previously been developed by seven different software developers as part of an unrelated study [9]. Adjusted for minor differences in the scope of the specification, the median actual effort spent by these developers to complete the software development work was around 190 work-hours.

After completing the first estimation task, the participants estimated the effort required to develop two smaller software systems (*Estimation task 2*). One of the systems (*System A*) was a shoe sale support system and the other was a dinner reservation system (*System B*). Those who previously gave relative estimates in story points now gave absolute estimates of the two systems in work-hours. Those who previously gave estimates in work-hours now gave System A ten story points and estimated the number of story points for System B relative to System A. We had previously received the absolute estimates in work-hours of both systems from 423 professional software developers as part of an unrelated experiment [11]. The median estimate of System A was 90 work-hours and the median estimate of System B was 69 work-hours, indicating that System A was typically estimated to be 130% of System B. The estimation sequence of System A and System B for these developers in the previous study was random.

The complete estimation instructions and software requirement specifications for both estimation tasks are included as supplementary material.

*Data collection and analysis:* We collected the estimates given by the participants and, through the use of the time logging functionality of the survey tool (Qualtrics), the time they spent on each estimate. The time logging functionality, other than the total time spent on all estimation tasks, was not set for the first batch of participants[[5]](#footnote-5) (27 participants), leaving 75 participants with information about time spent for the estimation work.

The questions related to our test of the hypotheses, with connected analysis processes, and relevant estimation tasks are described in Table 2. The developers did not complete the tasks – that is, we did not have the actual effort that would enable us to assess the actual estimation accuracy. The first two questions in Table 2, nevertheless, indirectly address the differences in estimation accuracy by comparing the relative and absolute estimates. If these are similar in value, variation and biases, this suggests that we should not expect much difference in estimation accuracy. In addition, we compare the estimated values with the effort actually spent (for Estimation task 1) or estimated (for Estimation task 2) by other professional developers to enable an indication of expected estimation accuracy.

**Table 2: Questions and analyses**

|  |  |
| --- | --- |
| **Question** | **Analysis** |
| Are the relative and absolute estimates different? | Transform the relative estimates to absolute estimates by use of the absolute estimates of the reference object. Compare the absolute estimates and the transformed relative estimates. (Estimation task 1) |
| Are the variances in the relative and absolute estimates different? | Calculate the coefficient of variance (CoV1) of the estimates per participant and compare the CoVs of the absolute with those of the relative estimates. (Estimation task 1)  Compare the ratios of the relative and absolute estimates. (Estimation task 2) |
| Is there a difference in time spent on estimation? | Compare the time spent on the estimation tasks by those using absolute and relative estimation. For the relative estimates, we *exclude* the time spent on estimating the reference user story using an absolute scale. (Estimation tasks 1 and 2) |

1: The CoV of a developer on Estimation task 1 is the standard deviation of his/her estimates of the five user stories divided by the mean of the estimate of the same five user stories. A larger CoV indicates more variance of the estimates around the mean estimate.

When relevant, we report separate analyses on participants with extensive prior experience in relative estimation – here defined as those who have used relative effort estimation at least 20 times before – and those without this extent of experience in relative estimation. A threshold of using relative estimation at least 20 times was set to ensure that the experience in relative estimation was extensive and that there was no need to spend time on learning how to perform relative estimation in the context of the experiment.

*Limitations*: Our analyses do not intend to be a complete evaluation of the advantages and disadvantages of relative estimation or the complete estimation process with communication with the product owner and other stakeholder. We do in our study not, for example, include group-based relative estimation with feedback on actual velocity from previous work, where story points may enable better group estimation discussions and more efficient feedback-based improvement of estimation accuracy [12]. While this limits the use of our results to inform the decision on whether to use relative estimation, e.g., in a context with group-based estimation and velocity-feedback, we believe that the results on how individual effort estimation is affected by the use of relative and absolute estimation are more robust. The basis of group-based estimation is individual estimates, which means that the results are relevant in that context, as well. We acknowledge that it is possible that the results would be different in a context where relative and absolute estimation is combined with group-based estimation and velocity-feedback and believe that this should be subject to further research. Potentially, the value of relative estimation is more in how estimates are discussed and communicated, rather than in their accuracy and ease of estimation.

There are limitations regarding how we transfer from relative to absolute estimates in some of the analyses. In real-life contexts, this may be done by collecting velocity data (productivity measured as delivered story points per time unit) from previous deliveries. For practical reasons, as described above, we requested the participants to estimate the reference object using an absolute scale. Although this is a limitation in realism, we believe that it does not give an unfair advantage to relative or absolute estimation.

The analysis of time spent on relative compared with absolute estimation is limited by the fact that we have no direct access to how much time the developers spent on the estimation work and how much time was spent on understanding the estimation instructions. In particular, the instructional text for Estimation task 1 for those with relative estimation was 170 words longer than that for those with absolute estimation, and is likely to take more time to read. We believe that this limitation to some extent will be overcome by emphasising the estimates from those with extensive prior experience in relative estimation, who will need less time for reading and understanding the instructions on how to give relative estimates. Even then, however, there will be uncertainty about how much we can say about the time spent on actual estimation. Only large differences in estimation time, extending the time usage differences reasonable to assume regarding time spent on reading and understanding the instructions, should consequently count as results on true differences. Note that we excluded the time spent on enabling the transfer from relative to absolute estimates – that is, the estimation of the reference story point in work-hours – in our analyses. This may be said to give a time usage advantage to those with relative estimation, since a transfer from relative to absolute effort estimates will be needed in real-world software development contexts.

The potential difference in productivity of the developers of this study and those actually completing the software development of the three systems means our use of the actual effort, and the accuracy evaluation based on that value, is only able to give weak indications on the realism of the estimates.

The impact from relative and absolute estimation may depend on the task or project being estimated. In our study we include estimation of a number of tasks (user stories) part of one project and the estimation of two projects. While we clearly cannot generalize our results to all types of tasks and systems, we argue that similarity in results between those two contexts adds to the robustness of the results. An interesting topic, for further research, is to what extent relative estimation, compared to absolute estimation, gets better or worse as the tasks get more similar.

Although the above limitations in realism and time measurement reduce the generalisability of our results, we believe that the analysis has the potential to make useful contributions to more evidence-based practice and more informed decisions regarding the use of relative estimation [13]. The key advantages of the current research design are that it is based on the use of realistic software development estimation tasks, professional developers, comparison of relative and absolute effort estimates on the same tasks and random allocation of developers to estimation processes. The use of randomization in the allocation of treatment (estimation method) means that we, in spite of variance of expertise and competence of the developers and potential non-representativeness of the participants, can be confident in that it is the treatment, not other factors, that leads to the observed differences.

## 2.2 Results

Tables 3 and 4 display the results of the analyses, including Mann-Whitney – two-sided tests of no difference between the characteristics of the relative and absolute effort estimates. We use non-parametric, median-based tests owing to occurrence of outliers affecting the mean value and resulting in non-normal distribution of the effort estimates. We will in this and following analyses denote statistical tests with *p* < .05 as statistically significant. Although *p* < .05 is an arbitrarily chosen, common threshold, we find this threshold useful to separate differences (effect sizes) more likely to be a result of random variance and those more likely to represent systematic differences. The 95% confidence intervals of difference in median values show the interval that 95% of the times (if repeating the data collection and analysis) would include the actual median value.

**Table 3: Results from Estimation task 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Characteristic** | **Relative estimates (median)** | **Absolute estimates (median)** | **95% confidence intervals of difference in median values and *p* value of no difference (Mann-Whitney tests)** |
| Estimated total effort (sum of the effort for the five user stories) of all developers.1 | 114 work-hours (n = 51)2 | 148 work-hours (n = 50) | CI95 = (−45, 49). No diff. *p* = 1.0 |
| Estimated total effort for those with much experience in relative estimation. | 114 work-hours (n = 35) | 160 work-hours (n = 33) | CI95 = (−41, 81). No diff. *p* = .64 |
| CoV (coefficient of variance) of estimates of the five user stories of all developers. | 0.58 (n = 52) | 0.60 (n = 50) | CI95 = (−0.11, 0.13). No diff. *p* = .73 |
| CoV of estimates of the five user stories for those with much experience in relative estimation. | 0.51 (n = 36) | 0.60 (n = 33) | CI95 = (−0.01, 0.24). No diff. *p* = .08 |
| Time spent on the estimation of the five user stories for all developers. | 506 s (n = 37) | 283 s (n = 37) | CI95 = (−282, −86). No diff. *p* = .001 |
| Time spent on the estimation of the five user stories for those with much experience in relative estimation. | 508 s (n = 24) | 245 s (n = 28) | CI95 = (−319, −115). No diff. *p* = .001 |

1 The relative estimates were transferred to work-hours by use of the absolute estimate of the reference user story.  
2 One participant failed to estimate the reference user story in work-hours and is not included.

The median of the absolute effort estimates was higher than the median of the transformed relative estimates. The difference was, however, not statistically significant. The difference between the coefficients of variance was small and not statistically significant either. This lack of statistically significant differences existed for the set of all developers as well for the subset of participants with much experience in relative estimation.

A comparison of the median estimated work-hours with the median effort spent by other developers previously completing the development work (190 work-hours) indicates that the estimated values were realistic. It also indicates that the median value of the absolute estimates (148 work-hours) was somewhat closer to the actual effort than the median value of the relative estimates (114 work-hours). Overall, we interpret this as weakly suggesting that the relative estimates had lower accuracy than the absolute estimates.

A follow-up analysis of the estimates indicates the presence of *assimilation effects*[[6]](#footnote-6) for a sub-set of the developers providing relative estimates. The 19 developers (37% of those providing relative estimates) who selected the largest user story (User story 2) as their reference story estimated the total effort to be substantially higher (median total effort of 161 work-hours) than those who selected one of the smaller user stories (User stories 1, 3, 4 or 5) as their reference story (median total effort of only 109 work-hours). This is consistent with the finding from previous research that estimates tend to become similar to (are assimilated towards) the estimate given for the reference task [14]. In general, the presence of more assimilation effects seems to decrease the estimation accuracy [15].

The time spent on the estimation work by participants providing relative estimates was measured to be higher than the time spent by those providing absolute estimates. As discussed earlier, this increase in time usage may, to some extent, have been caused by the longer estimation instructions for relative estimation. Two observations in support of, nevertheless, interpreting the results as contradicting the hypothesis that relative estimation is easier and faster are the following: i) The increase in time usage for relative estimation is substantial (79%, close to 4 min, increase in time usage), and ii) the increase in time usage remains (and even increases) when including only those with extensive prior experience in relative estimation.

**Table 4: Results from the Estimation task 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Characteristic** | **Relative estimates (median)** | **Absolute estimates (median)** | **95 confidence intervals of difference in median values and *p*-value of no difference (Mann-Whitney tests)** |
| Estimated effort of System A in percent of that of System B.1 | 83% (n = 49) | 125% (n = 49)2 | CI95 = (9%, 67%).No diff. *p* = .004 |
| Estimated effort of System A in percent of that of System B for those with much experience in relative estimation. | 80% (n = 32) | 133% (n = 33) | CI95 = (16%, 83%). No diff. *p* = .003 |
| Time spent on the estimation of System A and System B. Values in seconds. | 283 (n = 36) | 226 (n = 35) | CI95 = (−162, −3.9). No diff. *p* = .042 |
| Time spent on the estimation of System A and System B for those with much experience in relative estimation. Values in seconds. | 263 (n = 23) | 197 (n = 26) | CI95 = (−173, −4.2). No diff. *p* = .040 |

1 Four of the participants failed to give estimates on this task.

2 The estimates in work-hours were transformed to relative estimates by dividing the estimated work-hours of System A by the estimated work-hours of System B.

Table 4 indicates that the estimated effort of System B compared with that of System A to a large degree depended on whether the participants gave relative or absolute estimates. Those providing relative estimates tended to believe that System A was the smaller system (median System A in estimated work-hours was 83% of median System B in work-hours), whereas those providing absolute estimates tended to believe that System A was the larger system (median System A in story points was estimated to be 125% of median System B in story points). This difference increased when only examining those with much experience in relative estimation. Whereas 63% of those estimating in story points believed that System B was larger than or required the same effort as System A, only 39% among those estimating in work-hours believed the same.

This finding is consistent with what is reported in [14] and potentially explained by the feature matching theory [16]. Those providing relative estimates, in accordance with the feature matching theory, may have had a tendency towards neglecting features (characteristics) only present in the reference object when comparing the target object (System B) with the reference object (System A). Considering the characteristics only present in System B and ignoring the characteristics only present in System A may lead developers to think that System B is larger than System A. Regardless of the explanation of our finding, the observed difference in whether System A or System B was believed to be the larger one illustrates that explicit comparisons with a reference object sometimes can result in very different effort estimates than those resulting from a sequence of absolute estimates.

As noted earlier, the median estimates of other professional developers estimating the same two systems in work-hours give that other developers typically believed that the effort to complete System A was 130% of System B. The absolute estimates in this experiment, where (median) System A was estimated to be 125% of System B, are consequently much closer to those given by other developers.

The time spent on the estimation of the two systems was, as for Estimation task 1, lower for absolute than for relative estimation, even for those with extensive experience in relative estimation. This may be an even stronger indication of at least as much estimation time spent on relative estimation as on absolute estimation compared with the results from Estimation task 1. In Estimation task 2, those estimating in work-hours had to give *two* estimates, that is, the work-hours of both System A and System B, whereas those estimating in story points only gave one estimate (the story points of System B relative to the ten story points allocated to System A). In spite of this, those using relative estimation spent more time.

In total, our analyses can only give weak results regarding what we may expect regarding estimation accuracy with relative and absolute estimation. What we have as evidence do, however, suggest that the accuracy of relative is not better, rather worse, than those of absolute estimates. Relative estimates may be quite different from absolute estimates, as found for Estimation task 2. While we cannot reject Hypothesis 1 based on strong evidence, we can at least say that our results do not support it in the contexts studied by us.

The time spent on relative effort estimates was higher than that spent on absolute estimates (for both estimation tasks). We find that our results, in spite of the analysis limitations due to different lengths of the estimation instructions, give no support to our Hypothesis 2. Perhaps the strongest evidence supporting this lack of support is the observation that even those with much previous experience in relative estimation spent more time on providing *one* relative estimate (how many story points is System B, assuming that System A has ten story points) than those providing *two* absolute estimates (first estimating System A and then estimating System B in work-hours). This of course does not exclude that there are real-world situations where relative estimation is faster than absolute estimation.

# Experiment 2: How person-independent are relative effort estimates?

## 3.1 Study Design

The large variance in developer productivity, sometimes formulated as ‘your hour is not the same as my hour’, is as pointed out earlier in this paper used as an argument against absolute estimates and in favour of using relative estimates. Implicit in this argumentation is, the assumption that relative effort to a large extent is person-independent and, as a consequence, that relative estimates can be accurate even when not knowing who will perform the task and there is a larger variation in productivity. Exemplified on a situation with two tasks, Task X and Task Y, total person-independence of relative effort estimates would require that the ratio is the same for all developers. This, in turn, is the same as stating that there is no variance in effort performance dependent on the task (no within-developer variance).

This motivates this section’s analyses of the following:

* How large is the variance of the relative effort between developers, where relative effort is measured as the ratio of effort usage of pairs of two tasks?
* What is the proportion of the variance in effort to solve a task that is explained by between-developer variance, as opposed to within-developer variance?

*Participants*: The participants were recruited via a request for consultants sent to Norwegian consulting companies. The request specified the required education and expertise of the participants. Companies replied with the curricula vitae of potential candidates, which were screened to verify that they complied with the requirements. The participants were required to at least have a bachelor’s degree in informatics (or equivalent) and familiarity with the required technology of the system on which they were supposed to complete development tasks. In total, 20 skilled software professionals (all male) were selected, with on average 9 years of experience as software professionals. The software professionals were paid close to ordinary fees for their work and asked to treat the development work as ordinary consultancy work.

*Programming tasks*: We designed five different programming tasks. All five tasks consisted of extensions of an existing software system with information about research publications. The software consisted of about 50 classes and 3,000 lines of Java code. The task requirements are outlined below:

* Task 1: Addition of functionality to save a user’s search query to persistent memory.
* Task 2: Requires that the system is extended to handle an additional piece of data from an input file (in XML format) used to update the publications.
* Task 3: Add functionality to the system that extends the manner in which publication metadata associated with each publication are dealt with, particularly the ability to add publication categories and corresponding codes.
* Task 4: Add caching logic to the system so that if statistics for all the publications in the system are requested, the cached results are used (so as to decrease the computational load on the system).
* Task 5: Add functionality such that the users can delete existing publication codes from the system.

*Data collection, research questions and analysis*:The developers completed the following steps for the five tasks:[[7]](#footnote-7)

1) Read the requirements of Task 1

2) Complete Task 1 (design, program, test and document)

3) Send in the task for acceptance testing (automated tests)

4) If passing the test, do the same for Tasks 2, 3, 4 and 5. If not passing the test, go back to Step 3.

We logged the effort spent on actual working time, including the effort spent on fixing errors found in the acceptance testing. The participants were not allowed to start the next task before the previous task had been completed and they had passed the test. The complete estimation instructions and software requirement specifications are included as supplementary material.

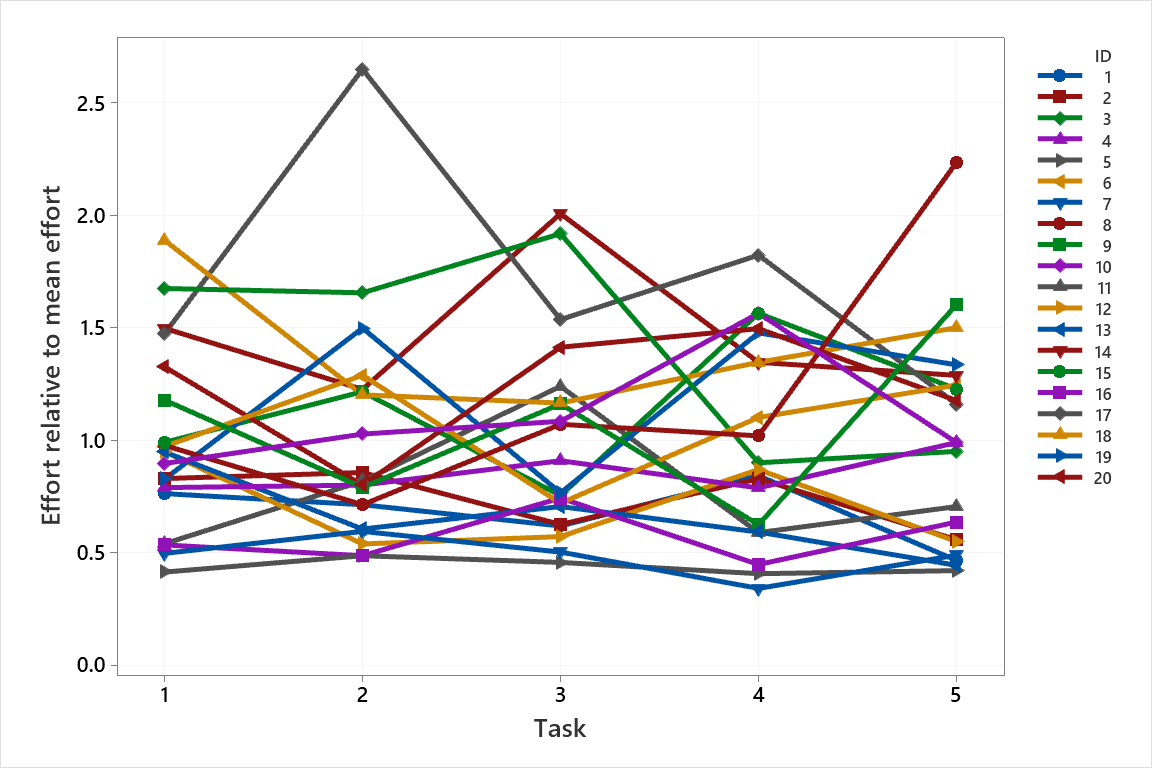
*Limitations*: The extent of person-independence of relative effort is likely to vary depending on the characteristics of the tasks and developers. This limits the generalisibility of the findings. We believe, however, that the studied context, with a rather homogenous set of skilled developers and tasks, may favour the finding of less person-dependency of relative effort compared to contexts with more specialised developer skills and more heterogeneous types of task. A developer relatively good at database-queries may for example be relatively poor at back-end development of Java code. This suggests that the proportion of within-developer variance for many real-world contexts may be much larger than what we report from this experiment. Notice that the analysis includes effort data from smaller tasks and individual work only. There may consequently be mechanisms that make the relative effort less (or more) person-dependent for larger tasks and/or for people working in development teams.

## Results

All developers completed all tasks successfully, sometimes after a few rounds of testing of their programs by us and subsequent corrections by them. The mean effort to complete the 5 tasks (with standard deviations) was for the first task 6.21 work-hours (standard deviation: 2.49), the second task 4.78 work-hours (standard deviation: 2.46), the third task 12.36 work-hours (standard deviation: 5.53), the fourth task 6.12 work-hours (standard deviation: 2.71) and the fifth task 8.49 work-hours (standard deviation: 4.04). This variation in effort usage, or productivity, among developers is similar to that found in other studies. The analysis of effort data from 61 different experiments reported in [18] finds, for example, that the median of the best and worst 25% of developers typically differ with a factor of two to three. and that the standard deviation divided by the mean is typically about 0.5. A ratio of the standard deviation to the mean of 0.5 is, as can be seen, close to what we observed for the five tasks included in our study.

Figure 1 displays the mean-centred relative effort – that is, the effort spent to solve a task by a developer divided by the mean effort of all developers on that task – of the 20 software developers for the 5 tasks.

**Figure 1: Mean-centred relative effort**



As visualised by the non-horizontal, crossing lines in Figure 1, the developers considerably varied in relative effort. As an illustration, the developer with Id 8 spent about the average effort on Task 1, almost 30% less than the average effort on Task 2, about the average effort on Tasks 3 and 4, and more than twice as much effort as the average effort on Task 5. An interesting visual observation is that the most productive (skilled) developers, that is, those with the lowest values in Figure 1, may have had less variation (fewer crossing lines) in effort spent compared with the mean effort than those who were less productive. This motivated us to include a follow-up analysis examining whether there may be differences in person-dependency of relative effort among those with higher and lower productivity (skill) levels.

The individual variance in relative effort is further illustrated in Table 5, which shows the characteristics of the effort ratios of all unique task combinations. We use the effort ratio name T1DivT2 to denote the ratio of the actual effort on Task 1 divided by the actual effort on Task 2, T1DivT3 to denote the ratio of the actual effort on Task 1 divided by the actual effort on Task 3, etc. Table 5 displays the median (Q2), the first quartile (Q1) and the third quartile (Q3) of each of the task ratios. The interval from Q1 to Q3 includes 50% of all observations. The ratio Q3/Q1, displayed in Table 5, consequently gives the lower bound on how 50% of the effort ratios vary. The rightmost column shows the proportion of effort ratios deviating more than 25% from the median effort ratio (Q2).

**Table 5: The effort ratios (relative effort) of all task combinations**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task ratio** | **Median (Q2)** | **Q1** | **Q3** | **Q3/Q1** | **Outside +/−25% of Q2** |
| T1DivT2 | 1.30 | 1.08 | 1.82 | 1.68 | 20% |
| T1DivT3 | 0.49 | 0.44 | 0.66 | 1.51 | 15% |
| T1DivT4 | 1.02 | 0.90 | 1.27 | 1.41 | 15% |
| T1DivT5 | 0.72 | 0.58 | 0.97 | 1.67 | 25% |
| T2DivT3 | 0.37 | 0.26 | 0.48 | 1.81 | 20% |
| T2DivT4 | 0.80 | 0.65 | 0.95 | 1.46 | 10% |
| T2DivT5 | 0.58 | 0.45 | 0.70 | 1.55 | 20% |
| T3DivT4 | 2.01 | 1.46 | 2.98 | 2.04 | 25% |
| T3DivT5 | 1.59 | 1.11 | 1.93 | 1.73 | 15% |
| T4DivT5 | 0.73 | 0.60 | 0.99 | 1.66 | 30% |

The ratios of Q3/Q1 vary between 1.41 and 2.04, with a median of 1.53. A Q3/Q1-ratio of 1.53 implies that 50% of the developers have a relative effort (an effort ratio) on the included tasks that differs from each other by at least 53%. The proportion of developers with effort ratios deviating by more than 25% from the median effort ratio varies between 15% and 30%, with a median of 20%. Both results suggest a high variance of effort ratios between the developers.

To examine how the variance of effort ratios between the developers may depend on the skill level, we split the developers into two similarly sized skill groups based on their effort on Task 1. Repeating the above analysis for each of the two groups, we find that the median ratio of Q3/Q1 in the high skill group (those with lower than the median effort spent to solve Task 1) was 1.23, whereas that of the low skill group (higher than the median effort spent on Task 1) was 1.71. The proportion of developers with effort ratios deviating by more than 25% from the median effort ratio had a median of 36% for those in the high skill group and a median of 62% for those in the low skill group. This supports the previous observation in Figure 1: Within a group of developers, the relative effort is less person-dependent among those with higher skills.

Our second analysis aims at analysing how much of the variance in use of effort can be attributed to *within-developer* variance in performance. To analyse this, we ran a linear mixed model with the logarithm of the actual effort as the response variable, developer id (1...20) as the random variable and the estimation task (1...5) as the fixed effect. The logarithm of the effort was used because the distributions of effort usage per task were strongly right-skewed. The log-transformation made the effort values close to normally distributed. The variance estimation was based on the restricted maximum likelihood, and the tests of fixed effects used the Kenward-Roger degrees of freedom approximation. The model produced close to normally distributed conditional residuals.

The residual variance of the random effects of the linear mixed model is the effort variance *not* explained by the differences between the developers or by task differences. This residual can be interpreted as the within-developer variance, assuming a reasonably good model fit.

**Table 6: Mixed model parameters and results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dependent variable** | ln(effort), where effort is the actual effort on a task | | | |
| **Random effects** | **Variable** | **Variance** | **Percent of total variance** | **Test** |
| Developer (1..20) | 0.13 | 64%  [Between-developer variance] | Z-value 2.77,  *p*-value .003 |
| Residual | 0.07 | 36%  [Within-developer variance] | Z-value 6.16,  *p*-value <.001 |
| **Fixed effect** | **Variable** | **Tests** | | |
| Task (1..5) | F-value 36.74, *p*-value <.001 | | |

As can be seen in Table 6, the within-developer variance is 36% and the between-developer variance is 64% of the total variance in use of effort. This suggests that the within-developer variance in use of effort is lower than the between-developer variance, but still substantial – that is, there is a strong person-dependency in relative effort.

Analogous to the previous analysis in this section, we developed separate linear mixed models for the groups of developers with lower and higher effort spent on Task 1. Consistent with the previous result on relative effort, we found the within-developer variance of the lower skilled developers (68%) to be higher than that of the higher-skilled developers (38%). This further supports that the person-dependency of relative effort decreases with higher skills.

The generality of the results in Table 6 is supported by a re-analysis of the data used in [10].[[8]](#footnote-8) That study reports the effort of 494 developers in solving the same ten programming tasks as part of their personal software process (PSP) training. Applying the same analyses as in the current paper on the data from that study, we find a Q3/Q1 ratio of 1.94 and a within-developer variance of 37% – that is, the results are not very different from what we found in our study.

Furthermore, this result corresponds with the perception of software developers, as found in a simple follow-up study on 32 software developers from two different consultancy companies. These developers were presented a context of standard software development tasks of average complexity and a team of seven developers. First, they were asked about how much the productivity typically would vary on the same task within such a team. Here they responded (median value) that the most productive developer would be 3.3 times faster than the least productive, which is a difference in productivity similar to that observed in our study[[9]](#footnote-9). Then, they were asked about how much of this variation they believed was systematic (explained as stable between-developer variance) and how much that was unsystematic (explained as variation in the individual developers’ performance). Their responses gave that they typically (median value) believed that 33% of the variance was unsystematic, i.e., they assessed the within-developer variance to be close to what we observed in our study.

As is clear from the above analyses, our results document a strong developer-dependency of relative effort, and consequently of the realism of relative effort estimates, that is, the results do not support our Hypothesis 3.

# Discussion and conclusions

The first hypothesis tested was that of relative estimation being more accurate than absolute estimation. Whereas our results are indirect and mixed, we interpret them as not giving support to this hypothesis. Instead, we find that relative and absolute estimates tend to be less accurate. We also find evidence in support of relative estimates being exposed to assimilation and feature matching biases, potentially leading to less accurate effort estimates. As pointed out in the limitation section, we can only claim this for individual estimates, not for contexts where relative estimation is combined with, for example, group-based estimation.

A possible explanation for the similarity in relative and absolute estimates is that absolute estimates are inherently based on relative estimation: there are similarities in the underlying estimation processes. Laming concludes his extensive research on relative judgments with the claim: ‘There is no absolute judgment. All judgments are comparisons of one thing with another’ [7]. If his claim is correct and all judgment-based estimates involve comparisons with other objects, then the difference between relative and absolute estimation may mainly be related to the choice of the reference object. If the implicitly chosen reference object(s) in absolute estimation are similar to the explicitly chosen reference object(s) used for relative estimation, the resulting estimates may be similar. If, however, the chosen reference objects differ, such as when an unusually large reference object is selected for relative estimation and more moderately sized reference objects are chosen for absolute estimation, the estimates may differ. The results reported in this paper are consistent with that the participants usually selected similar reference objects for relative and absolute estimation, but the results also indicate that differences in estimates occurred when participants selected an unusually large reference object for relative estimation.[[10]](#footnote-10)

Prior studies, as reported in the introduction, give evidence, although weak and limited to two studies, in support of better estimation accuracy of relative estimates, seemingly contradicting our results and supporting the first hypothesis that relative estimates are more accurate than absolute estimates. The prior studies and this study both have limitations related to their study design and measures of estimation accuracy that may explain the differences in results. The prior studies did not compare relative and absolute estimates on the same tasks or use robust measures of estimation accuracy. This study did not collect the actual effort enabling a direct measure of effort estimation accuracy but relied on comparing the actual effort with the estimates provided by other developers on the same tasks and the observation of potential estimation biases. Overall, it may be fair to conclude that we are still missing strong evidence on whether relative estimates are likely to lead to, and in which contexts, more or less accurate effort estimates.

The second hypothesis tested was that relative estimation is easier, leading to less time spent on estimation. This hypothesis was not supported by our results, and the participants spent more time on providing the relative estimates than on providing the absolute estimates. This may to some extent be explained by the longer estimation instruction texts and lack of experience with relative estimation. The longer time spent on relative estimation was, however, substantial even when only including participants with much prior experience in relative estimation. When completing Estimation task 2, those with relative estimates gave only one estimate – that is, the effort of System B relative to the effort of System A – whereas those with absolute estimates gave two estimates – that is, the effort of System A and the effort of System B. Even in this situation, those with much prior experience in relative estimation spent more time on relative than on absolute estimation. This suggests that relative estimation may involve at least as many steps and as high mental workload as that of absolute estimation.

No previous studies have tested the second hypothesis that relative estimation of effort is faster, indicating a simpler process, than absolute estimation. Our experiment, however, gave no support to this. Instead, the evidence suggests that the developers spent more time on providing relative than absolute estimates. The longer estimation instruction of those providing relative estimates may explain part of the increase but, we argue, hardly all of it.

The third hypothesis tested was that relative estimates to a large extent are person-independent. Our results do not support this hypothesis either. Although it is no doubt the case that ‘your hours are different from my hours’ (meaning that developers may vary much in effort needed to solve the same task), it is also true that ‘your story points are different from my story points´. We found that the relative effort was strongly person-dependent, as documented by the high individual variation in a developer’s use of effort relative to the mean effort of other developers on the same set of tasks and by the finding that more than half of the variance in use of effort was caused by within-developer effort variance. The degree of person-dependency of relative effort seems to depend on the skill level of the developers. In the group with the highest skilled developers, we found substantially lower person-dependency in relative effort. This means that we may expect much higher developer-independence of relative estimates in contexts with mainly highly skilled developers.

We conclude that our results provide evidence that weakens the hypotheses that relative estimates are more accurate, faster and to a large extent person-independent. Although our results should not be used to stop using relative estimation, which we believe may have advantages not examined in this paper, we hope that they may contribute to more informed choices and a more realistic expectation about the consequences of choosing relative or absolute estimates.

# References

[1] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: a survey on the state of the practice," in *Proceedings of the 19th international conference on Evaluation and Assessment in Software Engineering*, 2015, pp. 1-10.

[2] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.

[3] H. H. Arifin, J. Daengdej, and N. T. Khanh, "An Empirical Study of Effort-Size and Effort-Time in Expert-Based Estimations," in *2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, 2017: IEEE, pp. 35-40.

[4] I. Fredriksen, "Empirical research on relative and absolute effort estimation in software development projects," 2009.

[5] J. S. Armstrong, "Evaluating forecasting methods," in *Principles of forecasting*: Springer, 2001, pp. 443-472.

[6] N. Stewart, G. D. Brown, and N. Chater, "Absolute identification by relative judgment," *Psychological review,* vol. 112, no. 4, p. 881, 2005.

[7] D. Laming, *Human judgment: The eye of the beholder*. Cengage Learning EMEA, 2004.

[8] G. E. Bryan, "Not all programmers are created equal—Redux©," in *2012 IEEE Aerospace Conference*, 2012: IEEE, pp. 1-10.

[9] M. Jørgensen, "Better selection of software providers through trialsourcing," *Ieee Software,* vol. 33, no. 5, pp. 48-53, 2015.

[10] W. R. Nichols, "The End to the Myth of Individual Programmer Productivity," *Ieee Software,* vol. 36, no. 5, pp. 71-75, 2019.

[11] M. Jørgensen, "Communication of software cost estimates," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, pp. 1-5.

[12] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," *Hawthorn Woods: Renaissance Software Consulting,* vol. 3, pp. 22-23, 2002.

[13] B. A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in *Proceedings. 26th International Conference on Software Engineering*, 2004: IEEE, pp. 273-281.

[14] M. Jørgensen, "Relative estimation of software development effort: it matters with what and how you compare," *Ieee Software,* vol. 30, no. 2, pp. 74-79, 2012.

[15] T. Halkjelsvik and M. Jørgensen, *Time Predictions: Understanding and Avoiding Unrealism in Project Planning and Everyday Life*. Springer Nature, 2018.

[16] A. Tversky, "Features of similarity," *Psychological review,* vol. 84, no. 4, p. 327, 1977.

[17] M. Jørgensen and T. M. Gruschke, "The impact of lessons-learned sessions on effort estimation and uncertainty assessments," *IEEE Transactions on Software Engineering,* vol. 35, no. 3, pp. 368-383, 2009.

[18] L. Prechelt, *The 28: 1 grant-sackman legend is misleading, or: How large is interpersonal variation really*. Univ., Fak. für Informatik, Bibliothek, 1999.

1. Corresponding author. Magne Jørgensen, SimulaMet, Pilestredet 52, 0167 Oslo, Norway [↑](#footnote-ref-1)
2. Psychophysics may be described as the study of the relation between stimulus and sensations, e.g., the estimate of an object’s weights. [↑](#footnote-ref-2)
3. See for example: webgate.ltd.uk/estimating\_in\_scrum/ (accessed March 20, 2021). [↑](#footnote-ref-3)
4. The relatively low response rate and the lack of focus on a representative sample of developers limits how much we can generalise the *descriptive* results, such as the average estimates or time spent, to other populations. The identified relationships, such as the direction of the difference in time spent on relative and absolute estimation, are however believed to be more robust, given that we used a randomized controlled experiment as our study design. This is based on the assumption that the underlying mechanisms leading to the observed estimation relationships are likely to be present in other contexts. [↑](#footnote-ref-4)
5. The reason for this is that we did not consider possibility of collecting time usage in the beginning of the experiment. [↑](#footnote-ref-5)
6. Assimilation effect is a human bias with judgments towards the position of a context stimulus. (<https://en.wikipedia.org/wiki/Assimilation_and_contrast_effects>). The context stimulus (the comparison object) in our case is likely to be the chosen reference story. [↑](#footnote-ref-6)
7. The effort data were originally collected for the purpose of the study published as [17], which has a different focus than this paper. [↑](#footnote-ref-7)
8. We are grateful to William Richard Nichols who made the data available for our follow-up analysis. [↑](#footnote-ref-8)
9. As displayed in Figure 1, most developers spent effort between -50% and +100% of that of the mean developer in the experiment. This corresponded well to the median responses (-53% and +75%) in what the developers perceived was the difference in a team of seven in this follow-up study. [↑](#footnote-ref-9)
10. It is likely that the same will happen when selecting unusually a small reference object, but the number of observations of this type in our study was too small to enable meaningful analyses and robust conclusions. [↑](#footnote-ref-10)