Better Selection of Software Providers Through Trialsourcing

Magne Jørgensen
Simula Research Laboratory & University of Oslo
magnej@simula.no

Abstract: In this article we show that differences between software providers in terms of productivity and quality can be very large and that traditional means of evaluating software providers, such as CVs and evaluations by reference clients, fail to separate the competent from the incompetent ones. We argue and motivate through analyses of empirical data that software clients would benefit from using work sample-based tests of software development providers' performance (trialsourcing) to guide their selection of providers. We discuss potential challenges of trialsourcing and provide recommendations on what to consider when designing proper trialsourcing processes.

Keywords: Software development, provider selection, trialsourcing, skill assessment

Evaluation of provider competence

Selecting a competent provider for a software project is of great importance for its success. In the following, we show that traditional means of evaluating software provider competence are not always sufficient and that clients would benefit from the use of larger work sample tests, i.e., from the use of trialsourcing, to guide their selection of the most competent provider. First, we report from a case study where a company benefited from the use of trialsourcing. Then, we report from a controlled experiment documenting the large quality and productivity differences between companies which look very much the same when using traditional competence evaluation means, and from two observational studies documenting the benefits of using trialsourcing. Finally, we discuss different types of trialsourcing-based selection processes and potential challenges with such processes.

A trialsourcing case study

A telecom company invited software providers to bid for two projects similar in size and complexity. The telecom company informed the potential providers that it would select two providers, give them one project each, and use the performance on the two projects as a means to select only one provider for further development work on their software applications. The telecom company received four bids for the two projects and selected the two best bids. These two bids were evaluated to be similar with respect to price, provider competence and relevance of documented references from previous work. The project teams of both selected providers were based on a combination of local and offshore resources and involved about 20 software professionals each.

The telecom company's provider selection process based on large work sample tests – trialsourcing – turned out to be worth the extra overhead. From the first delivery, the differences in quality and productivity of the two providers were substantial. The better provider delivered the agreed functionality on time and with good quality, while the other provider had, as evaluated by the telecom company, severe problems understanding what to deliver and produced code of low quality. The poorly performing provider managed to only deliver about 70% of the functionality before the deadline. When including the required management, support and review resources, the telecom company had spent considerable extra effort in order to ensure useful deliveries. The true cost, of only 70% of the desired functionality, was estimated by the telecom company to be about 150% of the cost of the better performing provider. In addition, the code quality was noticeably lower. Following the trialsourcing-based evaluation of the two providers, the choice of one for further collaboration was quite easy. The telecom company's experience with the selected provider has for more recent work been very good.

Trialsourcing was here used successfully to reduce the risk of selecting a provider with low competence, which would not only have led to substantially higher total software development cost, but also to software quality-related problems.

A study of six providers completing the same project

To further examine to what extent commonly used competence indicators predict actual provider performance we designed a controlled experiment in which we paid six seemingly similarly competent-looking providers to develop the same small-scale software system. If similarly competent-looking providers perform very differently, this motivates the use of a trialsourcing-based selection process.

We conducted the study in the context of vWorker (now merged with *freelancer.com*), which offered a marketplace for connecting clients with software providers. The providers were typically single software developers or smaller outsourcing companies located in low-cost countries, but the marketplace did also include developers and companies from high-cost countries and larger companies.

The services offered by this marketplace included:

- Means for clients to search for and invite providers with appropriate skills to bid for software projects.
- Arrangements that ensure that the provider is paid when the work is completed and that the client does not have to pay if the quality of the work is too low.
- Processes for managing disagreements/negotiations between the clients and the providers regarding payments or quality of work (arbitration processes).
- Evaluations of providers' competence through the presentation of aggregated and project-specific information about client satisfaction, successes and failures of previous projects.

Typically, the providers and the clients never physically meet and conduct all their communication through the facilities provided by the marketplace or other internet-based communication. The failure rates and the success factors of such marketplaces seem to be about the same as in more traditional outsourcing contexts [1]. The priorities of clients when selecting providers also seems to be quite similar to that of traditional provider selection processes [2].

The bidding and selection process

In September 2012 we posted a request for bids for the development of a database-based software system at vWorker. The full specification, which is available upon request to the author, is 11 pages long. The implemented system, as completed by one of the selected companies, can be viewed at www.simula.no/BESTweb. We acted as an ordinary client, as seen from the perspective of the provider.

All bidders were requested to provide information about:

- The fixed price required to develop the specified software.
- The CVs of all the involved developers and the project leader, emphasizing relevant experience.
- The technology and high-level design proposed for the development work.
- A list of activities needed to complete the development work.
- An estimate of the total effort and the effort for each activity.
- A plan for the step-wise delivery of the functionality and the final delivery date.

We received in total 16 proposals for the development work. The proposals had effort estimates that varied from 43 to 432 work-hours and bid prices that varied from 99 to 15,660 USD. We selected six providers, all of which had high client satisfaction scores on previously completed projects (an average of at least 9.5 on a scale from 1 to 10), sufficient previous experience (completed at least 4 projects at vWorker, using relevant technology), CVs documenting relevant competence, and acceptable project proposals (acceptable development process and choice of technology). All six companies were paid their bid price. The characteristics of the six providers are displayed in Table 1.

Table 1: Characteristics of the six selected providers

Provider	Price (USD)	Mean client satisfaction rate (competence)	Number of rated projects	Estimated effort (work-hours)	Estimated price per work-hour
A	412	10.0	6	43	9.5
B	799	9.96	96	65	12.3
C	1,250	9.90	124	136	9.1
D	2,177	9.84	32	340	6.4
E	5,177	10.0	4	180	28.8
F	5,684	9.65	44	334	17.0

As can be seen from Table 1, there were only minor differences in mean client satisfaction on previous projects but quite large differences in effort estimates and price. The other competence indicators, such as quality of CVs and project proposals, were assessed to be similar for the six companies.

The results

We acted as clients for all six companies and aimed to give the same information and support to all of them. The acceptance testing was based on the same, predefined test cases and test procedures for all the providers. A delivery was accepted only if it passed all the specified tests. Table 2 shows key characteristics of the performance of the six providers.

Table 2: Performance characteristics

Provider	Actual effort[1]	LOC[2]	Acceptance test errors[3]	Cyclomatic complexity per LOC[4]	Estimate of extension effort[5]	Readability of code[6]
A	29	2,600 (6%)	20 (4)	0.09	2 (7%)	Good
B	84	2,900 (9%)	36 (13)	0.09	12 (14%)	Medium
C	187	2,800 (3%)	25 (7)	0.17	22 (12%)	Medium
D	242	5,800 (3%)	30 (7)	0.13	17 (7%)	Medium
E	527	19,000[7] (2%)	40 (16)	0.14	54 (10%)	Very poor
F	474	2,400 (4%)	27 (5)	0.16	40 (8%)	Poor

1: The effort in work-hours was reported as part of the weekly status reporting. The effort includes the correction of errors found in the acceptance tests.
2: The sum of executable lines of code written or modified rounded to the nearest 100. The amount of relevant comments, as a percentage of the lines of executable lines of code, is shown in brackets. The applications were written in a combination of php, Javascript, html/css, and SQL. An independent analysis determined that the coding style, in terms of how much a line of code included, was about the same for the six providers. All the providers used, although to different extents, frameworks such as CakePHP, Zend, and CodeIgnater.
3: The number of errors we found in the acceptance testing, with major errors in brackets. An error was categorized as “major” if the application could not be released with this error (not working at all) or only with great deviation from the specified requirements.
4: The cyclomatic complexity [3] per LOC was calculated using the php analysis tool (phpint.org). The analysis contains only the php part of the code, which for all the providers except Provider B constituted 90–98% of the written code. Provider B had a proportion of php of 64%.
5: The effort in work-hours, as estimated by the companies, to complete a specified extension to the application. This estimate’s percentage of effort of the total actual effort is given in brackets. The effort for the extension was estimated after the system development was completed and indicates the future costs of maintaining the system.
6: The assessment of the readability of the code was conducted by an external programmer.
7: Provider E seems to have modified a previously written application so that it fitted our specification. This led to functionality not specified (or needed) and many more lines of code than the other providers. However, even when not counting the lines of code related to the non-specified requirements, this provider developer/modified much more code (probably at least 10,000 lines of code) than the others. One likely reason for this is that this provider faced many coding difficulties, e.g., related to performance issues, and had to write many extra lines of code to make the existing code work for the specified requirements.

As can be seen in Table 2, Provider A had the lowest price and the best performance in terms of lowest use of effort, lowest number of errors found in the acceptance test, lowest cyclomatic complexity per line of code, lowest effort estimate for the extension, and most readable code. The fact that this provider did not look more competent than the other five providers at the time we had to select a provider illustrates the challenge of knowing when a low price is a threat and when it is an opportunity. In this case, the low price was a great opportunity for us to reduce the lifetime cost of the application substantially, but acting as a risk-averse client we would probably not have chosen a provider with a bid that much lower than the other bidders. This is a rational decision, given that, on average, the lower the bid compared to the average bid, the higher risk of project failure [4].

Clearly, the traditional competence indicators, i.e., quality of proposal, satisfaction of previous clients and CVs, were not sufficient, perhaps not even of much help, to separate the competent from the much worse performing software providers. The study also illustrates the benefit we would have had from applying the performance of the providers on this project to select a provider for future work on this or similar software applications, i.e., to use a trialsourcing-based selection process. Assume that Provider C would be the chosen provider, if selecting only one provider for our project, given its extensive experience, reasonable price and good references. Selecting Provider C instead of Provider A would in this case have implied a large increase in cost and decrease in quality. If we assume that the software development performance in Table 2 is representative of further work on this application and similar tasks, we would have to pay for six times more effort (= 189 vs 29 work-hours), and experience slower deliveries and lower quality. Six times the effort may actually be a conservative estimate, since the difference in estimates of the effort of extending the software is even more extreme (about eleven times the effort = 11 vs 2 work-hours) in favor of Provider A.

The observed large difference in productivity in this study is not unusual and reported in several previous studies [5, 6]. The problems using traditional skill indicators to predict future software development performance is analyzed in [7] with results similar to those presented here.

Previous client–provider collaboration

A client's use of its own experience with providers contracted for one or more previous projects when selecting a provider for their new project may be considered as a type of trialsource-based selection. If trialsourcing is efficient, we should consequently expect a reduction in risk of project failure when a client selects a provider with which the client has had previous, presumably good, experience, compared to when a client selects a provider based on traditional provider evaluation means only.

To examine this, we analyzed the fixed-price-based vWorker-projects with a price higher than 1,000 USD and at least two bidders. The projects smaller than 1,000 USD were excluded to avoid "toy" projects, such as students requesting support for their software development assignments. Projects with one bidder were omitted because they were not useful for a study on the effect of selecting between provider bids. The remaining data set contained 8,571 projects with a total of 84,219 bids, i.e., on average approximately 10 bids per project. The projects were conducted in the period 2001–2012. As pointed out in [1], the importance of the failure factors seem to be reasonably independent of project size within the studied size interval. This indicates that our analysis, in spite of including mainly smaller projects, may be of value for larger projects as well.

As many as 38% of the projects in the analyzed dataset failed, where we define failed as started and then cancelled or completed with a client satisfaction of 3 ("poor") or less on a scale from 1 to 10. The corresponding failure rate for the total data set of projects was 14%, which is at level with other types of projects, see [1]. The much higher failure rate of the analyzed data set is to a large extent explained by the fact that projects with only one invited bidder, probably with previous successful collaboration with the client, were excluded from the data set. We observed that a provider was 4.4 times more likely to be selected when collaborating with the software client in an earlier project.

To study the relation between the re-selection of a provider (previous collaboration) and the decrease of risk of project failure we created a binary, logistic regression model for the 8,571 projects. To isolate the effect of previous collaboration from other factors potentially explaining project failure, we added binary variables related to price (lower or higher than the mean bid price for the project), amount of experience (lower or higher than the mean number of completed projects among those bidding for the project), previous failure rate (lower or higher than the mean failure rate of previous projects completed by those bidding for the project), and client satisfaction (lower or higher than the mean client satisfaction of previous projects completed by those bidding for the project). The binary logistic regression model had as a dependent variable whether the project failed or not.

All included variables were statistically significant at $p<0.001$, and consequently affected the risk of project failure, except the amount of experience ($p=0.24$). The binary logistic regression analysis indicated that re-selecting a provider tried before reduced the risk of project failure to less than half (46%) of the risk otherwise. Including the projects with only one bidder, the result in favor of using previous experience (trials) as selection criterion became even stronger with a risk reduction to less than one fifth (19%) of that of selecting a provider not tried before. This suggests that provider selection based on previous experience with the provider, rather than the only based on traditional competence evaluation means, strongly reduces the risk of project failure. This in turn supports the benefit of using larger work samples – trialsourcing – to guide the selection of providers.

Experience with trialsourcing at vWorker

The trialsourcing process supported by vWorker is a process where all invited providers produced an initial piece of the software to be developed. The trialsourcing work, the initial piece of software, is paid work. The client then use the providers' performance on the trialsourcing work as evaluation criterion for selecting the best provider, which then completes the rest of the work. Trialsourcing had recently been introduced at vWorker when we contacted them in 2012 and only 176 projects had used it.

The project outcome data from the trialsourcing-based projects were very convincing. Among the trialsourcing-based projects there were no (0%) failures! As reported earlier, the projects not applying trialsourcing at vWorker had a failure rate of 14%. The mean client satisfaction on the trialsourcing projects was also very good with a mean value of 9.8 (maximum is 10). The mean client satisfaction for projects not applying trialsourcing was 8.4.

A potential challenge with trialsourcing projects indicated by the vWorker-data was that those projects received fewer providers competing for the project. While ordinary fixed-price projects typically received about ten bids per request, the trialsourcing projects typically received only two. Fewer bidders may be an advantage if the trialsourcing method keeps the low-productivity, low-quality providers away from producing proposals. It may, however, also be an indicator of providers hesitating to compete with other providers based on trialsourcing.

How to design a good trialsourcing process

Trialsourcing may have different formats. The three types of trialsourcing discussed in this article are:

1) Same task trialsourcing. This type, which is the one applied by vWorker, implies that a client invites several providers, for example the three or four best qualified ones, to complete the same, representative, initial piece (increment) of the software to be developed. The strength of this type of trialsourcing is that the use of the same task for all providers simplifies the comparison of performance. The disadvantage is that the cost may be higher than for the other two types, especially if the size of the trialsourcing task is large.
2) Similar task trialsourcing. This type, which is the one used in the telecom company case, implies that a client invites several providers to complete similar deliveries, all part of the software to be developed. The advantage of this type of trialsourcing is that, given that all providers are able to deliver working solutions, the extra cost may decrease compared with same task trialsourcing. The disadvantage is that it may be more difficult to compare the performance.
3) Dissimilar task trialsourcing. The analysis of the vWorker projects suggests that even when the trials consist of, to some extent, dissimilar projects completed earlier, the emphasis of the experience from these experience strongly reduces the risk of project failure. It is to be expected that this is a less reliable evaluation and selection method than the other two types of trialsourcing given the larger variation in project contexts.

Selection and evaluation of trialsourcing tasks

Interviews we have made with software providers using tests of programmers in offshoring contexts suggest that small, well-specified skill testing tasks may be useful to identify developers with insufficient programming competence, but not always sufficient to identify those with low competence in understanding more complex requirements or lack of ability to request more information when something is unclear. The trialsourcing tasks need, consequently, to be sufficiently representative, large and complex to identify the true skill of the provider.

The actual size of the trialsourcing tasks should depend on the size of the total project and the importance of selecting a highly competent provider. To enable a realistic evaluation of the providers, the work effort may need to be a minimum of 1–2 weeks and include all phases of the software development process. We suggest that the requirement specification is of a nature where the providers have to request more information and/or ask for clarification to provide a meaningful solution. This enables an evaluation of the communication skills of the providers.

The evaluation of the task performance should, we believe, be similar to the one we conducted in the comparison of the six companies. It should include evaluation of the collaboration process, the number and severity of faults in the delivered software, the quality of the code (based on code review by an independent, experienced software professional), and the productivity of the providers relative to each other.

Development process

To ensure that the providers find trialsourcing sufficiently interesting it is essential that the trialsourcing method is explained well, that the selection criteria are clear and fair, and that the trialsourcing process is constructed so that the participating providers will not make a loss when competing for such projects. There may also be a need for means to avoid providers misusing the process, such as means to avoid providers using their best developers for the trialsourcing increment and more average developers for the remaining part of the project, or being dishonest about the actual effort used for the trialsourcing work.

The trialsourcing method can easily be extended to include more than one evaluation stage. A client may for example start with four providers for the first increment, reduce this to three for the second, and then, finally, select one or two providers for the remaining project. If more than one provider is selected for the remaining project, a competitive process may be used to select which provider or developer to select for upcoming tasks. The competition may for example be based on previous performance on similar tasks within the project.

Using trialsourcing in medium- to large-scale projects, as experienced by the author, includes both agile and more traditional software development contexts. In particular the incremental nature of agile projects seems to make them suitable for trialsourcing. Trialsourcing is applicable both for fixed-price and payment-per-hour types of contracts.

Limitations

The typical concern when presenting trialsourcing for companies which have never used it before is that they believe that it will not be considered acceptable to add cost for the management of more than one provider for a trialsourcing purpose. While our studies suggest that there are many contexts where the risk-reducing effect of trialsourcing pays off, there may also be contexts where it may not be worthwhile. An example is the situation where the client has reliable and highly relevant evidence about one providers' competence, e.g., from other clients with similar types of projects. Notice, however, that experiences from reference clients selected by the providers or from clients without proper skill in evaluating software development performance and quality hardly qualify as reliable evidence.

Conclusion

Traditionally, the competence of software providers is evaluated by use of CVs, price, project proposals, reference clients and, sometimes, small-scale programming skill tests. We illustrate that such means are not sufficient to avoid selecting incompetent providers. Through a case study, an experiment and analyses of a large number of projects we motivate the selection of software providers based on their performance on a representative, substantial increment of the software to be developed (trialsourcing). This evaluation and selection method may reduce the risk of project failure, which in turn may lead to substantial cost savings, increased productivity and increased quality.

References

1. Jørgensen, M., *Failure factors of small software projects at a global outsourcing marketplace.* Journal of Systems and Software, 2014. **92**: p. 157–169.
2. Gefen, D. and E. Carmel, *Is the world really flat? A look at offshoring at an online programming marketplace.* MIS quarterly, 2008. **32**(2): p. 367–384.
3. Halstead, M.H., *Elements of Software Science.* Operating and programming systems series. 1977, New York: Elsevier.
4. Jørgensen, M., *A strong focus on low price when selecting software providers increases the likelihood of failure in software outsourcing projects*. Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE '13), Porto de Galinhas, Brazil, 14–16 April 2013, p. 220–227.
5. Prechelt, L., *The 28:1 Grant-Sackman Legend is Misleading, Or: How Large is Interpersonal Variation Really*. 1999, Karlsruhe: Fakultät für Informatik, Universität Karlsruhe.
6. Anda, B.C.D., D.I.K. Sjøberg, and A. Mockus, *Variability and reproducibility in software engineering: A study of four companies that developed the same system.* IEEE Transactions on Software Engineering, 2009. **35**(3): p. 407–429.
7. Bergersen, G.R., J.E. Hannay, D.I.K. Sjøberg, T. Dybå, and A. Karahasanović, *Inferring skill from tests of programming performance: Combining time and quality*. Proceedings of the 3rd IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '11), Banff, Canada, 22–23 September 2011, p. 305–314.

Biography: Magne Jørgensen works as a researcher at Simula Research Laboratory and a professor at the University of Oslo. Previously, he worked with software development, estimation, and process improvement in the telecom and insurance industry. He has, together with Prof. Kitchenham and Dr. Dybå, founded and promoted evidence-based software engineering and teaches this to students and software professionals. His current main research interest is effort estimation, bidding processes, outsourcing, and software development skill assessments.