Pricing Crowdsourcing-Based Software Development Tasks

Ke Mao[*†], Ye Yang[*], Mingshu Li[*]
[*]Institute of Software, Chinese Academy of Sciences
[†]University of Chinese Academy of Sciences
Beijing, China
{maoke, yangye, mingshu}@nfs.iscas.ac.cn

Mark Harman
Dept. of Computer Science,
University of College London
London, UK
mark.harman@ucl.ac.uk

***Abstract*—Many organisations have turned to crowdsource their software development projects. This raises important pricing questions, a problem that has not previously been addressed for the emerging crowdsourcing development paradigm. We address this problem by introducing 16 cost drivers for crowdsourced development activities and evaluate 12 predictive pricing models using 4 popular performance measures. We evaluate our predictive models on TopCoder, the largest current crowdsourcing platform for software development. We analyse all 5,910 software development tasks (for which partial data is available), using these to extract our proposed cost drivers. We evaluate our predictive models using the 490 completed projects (for which full details are available). Our results provide evidence to support our primary finding that useful prediction quality is achievable (Pred(30)>0.8). We also show that simple actionable advice can be extracted from our models to assist the 430,000 developers who are members of the TopCoder software development market.**

***Index Terms*—crowdsourcing, pricing, software measurement**

I. Introduction

Crowdsourcing is an emerging paradigm for accomplishing traditional tasks. It is used to outsource software development through an open call format. The current formulations widely in use were defined by Jeff Howe in 2006 [1], but the approach also finds some resonance in the earlier idealistic development models proposed by the Free Software Movement.

Microtask crowdsourcing services such as Amazon Mechanical Turk have gained a great deal of popularity and recent interest in the Software Engineering community. However, online markets for crowdsourcing larger tasks, such as software project development, still have received comparatively little attention.

Examples of such online software project development markets include TopCoder, eLance, oDesk, vWorker, Guru, TaskCity, and Taskcn. Among these platforms, TopCoder has the largest developer community. It has also been used to create software for well-known established software-intensive organisations such as Google, Microsoft, Facebook and AOL.

Platforms such as TopCoder are geared towards highly skilled workers, who undertake time-consuming, complex and demanding software development tasks. The size of projects is many orders of magnitude larger than for microtask platforms such as Amazon Mechanical Turk. Consequently, the importance of arriving at a suitable price for the work is also an important decision problem for the engineers involved [2].

For crowdsourcing-based software development tasks, project managers need to provide rewards that are sufficiently attractive to the crowd. Inappropriate price can often lead to low capital efficiency or task starvation [3]. Therefore, the primary issue that a project manager faces is how to make decisions that establish sufficient incentives within their limited budget. This decision problem of determining the appropriate price is regarded as one top the five key challenges for crowdsourced software development [4]. This paper is the first to address this pricing issue for crowdsourcing-based software development tasks.

The rest of the paper is organized as follows: Section II presents an overview of the crowdsourced development paradigm. Section III introduces a set of nine predictive modelling approaches that can be used to address the pricing problem, including neural networks, case-based reasoners, machine learners and regression based models. Section IV presents the results of an empirical study of the predictive models, while Section V considers limitations and future work in this new research agenda. Section VI concludes.

II. Software Development Tasks on TopCoder.com

TopCoder supports a global crowd of more than 430,000 developers. In this section, we briefly describe the framework of crowdsourcing-based software development employed by the TopCoder platform. Other platforms may adopt similar models however TopCoder has made a great success in its methodology and is famous for its highly skilled programmers. It is for this platform that we shall report our findings.

A. Crowdsourcing-based Software Development Framework

The crowdsourcing-based software development framework involves five phases to produce the customers' requested

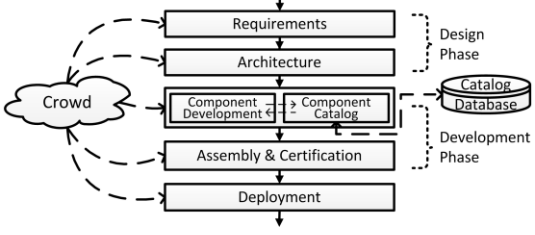


Fig. 1. Illustration of crowdsourcing-based software development process.

asset. Each phase is achieved by its global crowd of developers, in the form of competitive tasks. The submissions of each task are evaluated and scored via a peer review procedure, using a predefined scorecard. The top two contest winners are awarded the prize which is set up-front for each task. The winning submissions form the input to subsequent phases of the process.

The overall process is illustrated in Figure 1. It commences with a requirements gathering and project definition phase, in which the project manager communicates with the client company to identify the project objectives, task plan, time and budget estimates. This phase is regarded as a conceptualization and specification phase by the TopCoder platform.

The subsequent architectural phase decomposes the client's software project into a set of components and produces a full set of design documentation including, for example, UML diagrams and component specifications. The components designed by the winner of the bidding process are then implemented in subsequent development tasks.

Development may incorporate pre-built reusable components and the resulting implementations may also become a part of the reusable software catalog. The components are brought together in an assembly and certification phase, which involves the crowd competing to build the application by assembling winning components according to the architectural design. This assembly phase also includes system level testing activities.

In the final deployment phase, the fully functioning solution is deployed into the customer's quality assurance environment. After a period of user acceptance testing, all deliverables are then turned over to the client.

B. Dataset Acquisition

We implemented a crawler and used it to collect the 2,895 software design tasks and 3,015 software development tasks posted on TopCoder from Sep 29th 2003 to Sep 2nd 2012. Out of these tasks, 1,072 components belong to a reusable catalog, which provides detailed project source including code and documents to the public (protected by a subscription agreement). Each component matches 2 tasks corresponding to its design and development phases.

The 1,072 catalog components are classified into 19 categories by their application types (e.g. communication, security, web, etc.), 684 of them are in Java, the remaining 388 components are in C#. For the price field of a task, since the top two winners share the reward money (in the ratio 2:1), we define the price to be the sum of the money awarded to both the 1st and 2nd place winners.

III. Predictive Models for the Crowdsourced Development Pricing Problem

To address the pricing issue we investigate the application of predictive modelling techniques, based on historical data. We thus seek to extract the successful experience of previous decisions and outcomes to guide the current decision making. The success criterion used in this paper is defined according to TopCoder's official minimum review passing score which requires the winning submission to be scored at least 75 in the peer review process of each task.

In order to build structural as well as non-structural empirical pricing models, we propose 16 price drivers which fall into 4 categories based on analysing the characteristics of collected 5,910 tasks. These 16 price drivers are described in Table I together with descriptive and inferential statistics obtained from our analysis on the evaluation dataset. The regression coefficients in the final three columns of the figure will be discussed in detail later on. The 4 categories into which the 16 drivers fall are as follows:

1) **Development Type (*DEV*)**. (e.g. new components or updates to existing components, what programming language will be used). Price drivers from Table I: *ISUP, ISJA, ISCS.*

2) **Quality of Input (*QLY*)**. (e.g. the review score and related design statistics). Price drivers from Table I: *SCOR, WRAT, RAGI, SUBM.*

3) **Input Complexity (*CPX*)**. (e.g. implementation "difficulty" drivers). Price drivers from Table I: *TECH, DEPE, REQU, COMP, SEQU, EFRT, SUML, SIZE.*

4) **Previous Phase Decision (*PRE*)**. (e.g. the price decision of previous design phase). Price driver from Table I: *AWRD.*

TABLE I. Descriptive Statistics and Regression Coefficients of Proposed Factors

Category	Variable	Meaning	Descriptive Statistics					Regression Coefficients		
			Min	*Max*	*Mean*	*Medn*	*S.Dev*	*β*	*t*	*p*
DEV	*ISUP**	whether the task aimed at component update	0.0	1.0	0.2	0.0	0.4	-71.352	-1.998	0.046
	ISJA	whether the development language is Java	0.0	1.0	0.5	1.0	0.5	-	-	-
	ISCS	whether the development language is C#	0.0	1.0	0.5	0.0	0.5	23.986	0.808	0.419
QLY	*SCOR*	score of winner's submission of design phase	70.2	99.7	90.4	92.0	6.5	-3.032	-1.360	0.175
	WRAT	rating of the winner in design phase	481.0	3251.0	1688.2	1632.0	629.3	-0.021	-0.994	0.321
	REGI	number of registrants in design phase	1.0	38.0	9.3	8.0	5.2	2.692	0.914	0.361
	SUBM	number of submissions in design phase	1.0	26.0	2.8	2.0	2.4	-10.416	-1.601	0.110
CPX	*TECH*	number of technologies which will be used	1.0	7.0	2.1	2.0	1.3	-6.265	-0.569	0.570
	DEPE	number of component dependencies	0.0	16.0	2.9	2.0	3.0	2.555	0.606	0.545
	REQU	number of pages of requirement specification	2.0	23.0	3.7	3.0	1.8	-9.422	-1.332	0.183
	*COMP**	number of pages of component specification	3.0	56.0	11.3	10.0	5.7	8.480	3.414	0.001
	*SEQU**	number of sequence diagrams of the design	0.0	45.0	5.3	3.0	5.5	7.666	2.922	0.004
	EFRT	winner's effort(in days) in design phase	0.0	468.0	29.0	5.0	77.2	0.273	1.607	0.109
	SUML	size of UML design file, measured in KB	24.0	644.0	158.1	142.0	85.5	-0.293	-1.386	0.166
	*SIZE**	estimated size, measured in KSLOC	0.4	21.9	3.0	2.3	2.3	29.772	4.189	0.000
PRE	*AWRD**	winner's award(in $) of design phase	0.0	4200.0	920.1	900.0	457.6	0.503	17.912	0.000
-	*const**	the constant term used in regression analysis	-	-	-	-		463.194	2.437	0.015

These 16 price drivers were determined by a mixture of intuition (essentially "intelligent guesses" as to those factors that might be important), underpinned by statistical correlation analysis to determine whether the price used in the real projects was correlated to the proposed price driver. Since the analysis of this new software development paradigm is, as yet, in its infancy, we were permissive in our inclusion criteria; including any potential driver that exhibited even a weak statistical correlation. We can use the 16 price drivers we obtained as explanatory factors, deriving a multiple linear regression model, given by Equation 1:

$$PRICE = \beta_1 TECH + \beta_2 DEPE + \beta_3 REQU + \beta_4 COMP + \beta_5 SEQU + \beta_6 SCOR + \beta_7 AWRD + \beta_8 EFRT + \beta_9 SUML + \beta_{10} WRAT + \beta_{11} REGI + \beta_{12} SUBM + \beta_{13} ISUP + \beta_{14} ISJA + \beta_{15} ISCS + \beta_{16} SIZE + \beta_0 + \varepsilon \quad (1)$$

Here *SIZE* is the estimated size of the component, measured in Kilo Source Lines of Code (KSLOC). β_0 is a constant term and ε is the residual for particular observation. Of course, we can use this multiple linear regression model as a price predictor too. However, we also investigate 8 other forms of predictive models and also, as a baseline comparison, the traditional basic COCOMO'81 cost model [5], the Random guessing method which randomly assign the price of another case to the target case [6] and the Naïve model that arises from simply pricing the development task at the same rate as its previous design phase.

For completeness, we included many of the different approaches to predictive modelling used in other areas of Software Engineering. Our modellers are: three decision tree based learners (C4.5, CART, QUEST), two instance-based learners (KNN-1, KNN-k $\in$ [3,7]), one multinomial Logistic regression method (Logistic), one Neural Network learner (NNet) and one Support Vector Machines for Regression learner (SVMR). Our multiple Linear Regression model is called LReg in the following analysis.

IV. Experimental Evaluation

In order to assess the effectiveness of proposed predictive modellers, we apply a Leave-One-Out Cross-Validation (LOOCV) on the dataset introduced in Section II. After filtering out all tasks with missing values of proposed factors, there remain 980 tasks i.e. 490 projects include both design and development phases and so are sufficiently complete to support a full assessment. Our analysis of price drivers is thus derived from all 5,910 tasks available, but it is evaluated on the 490 projects for which complete information is available.

The resulting prices, $estimated_i$, are used to compute four popular performance measures that are widely used in current prediction system for software: Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), Standard Deviation Magnitude of Relative Error (StdMRE), Percentage of the estimates with relative error less than or equal to N% (Pred(N)).

We investigate three research questions:

RQ1: (Baseline Comparison): How much better are the proposed predictive models compared to the Naïve, Random and COCOMO'81 approach? If we cannot outperform these, then there is no reason to continue our research, as developers can simply use these existing approaches.

RQ2: (Performance Assessment): Which predictive model gives the best overall predictive performance as assessed by the LOOCV analysis of the four performance measures?

RQ3: (Actionable Insights): What actionable insights can we offer software engineers engaged in crowdsourced development from these emerging results?

The primary results of our empirical study are presented in Figure 2. The horizontal axis is ordered by the Pred(30) values obtained from each of the 12 predictive models (our 9 investigated models together with Naïve, Random and COCOMO'81 models as a baseline). Note that, as suggested by Shepperd et al. [6], we also tried to order these models according to the unbiased Mean Absolute Residual (MAR) measure and we found there is no significant change in the rank.

Answer to RQ1: It is immediately clear from the results that the Naïve model, the Random method and the basic COCOMO'81 model are outperformed by all 9 predictive models: All 9 models have a better predictive performance according to Pred(30) and most also have noticeably lower errors rates.

One potential explanation of the poor performance can be the size-to-effort ratio observed in TopCoder crowdsourced tasks. The basic COCOMO'81 model assumes a highly nonlinear relationship between size and effort. However, our results reveal that the crowdsourcing paradigm may be very different: We found there is no obvious association between size and effort for crowdsourced projects on TopCoder.

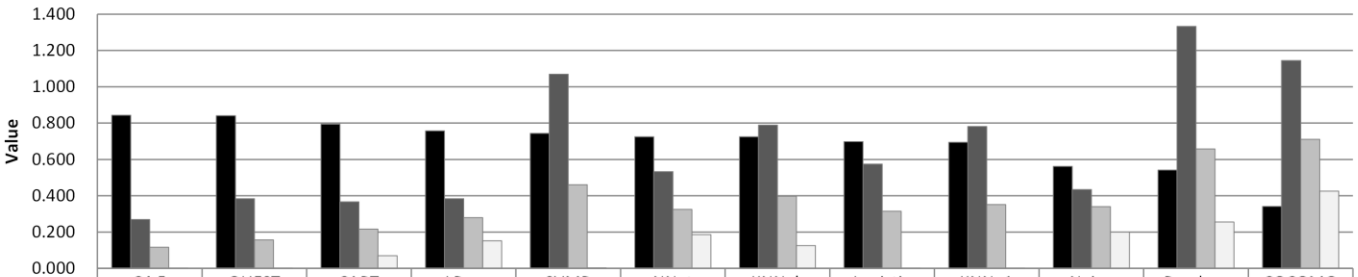
Answer to RQ2: The best performing modeller is C4.5. It has the highest Pred(30) value and also the lowest error rates according to all error measures and is, therefore, unequivocally the best performing predictor of those we studied.

Answer to RQ3: Our results indicate that price can be relatively predictable for this new paradigm. For the best performer (C4.5), more than 80% (84.3%) of estimates have an error lower than 30%. This is encouraging evidence that there may be value in predictive modelling for crowdsourced software development projects.

Though C4.5 turns out to give the best predictions, we also note that other models give strong performance. In particular, the multiple Linear Regression (LReg) gives reasonable results. This allows us to further investigate the significance of the explanatory factors of the LReg model to give additional insights to the 430,000 developers engaged in crowdsourced development using the TopCoder platform.

The regression coefficients derived from the entire calibration dataset are shown in the final three columns of Table I. The last two columns present *t*-statistics and *p*-values (since *ISJA* and *ISCS* are mutually exclusive in our dataset, the *ISJA* term is eliminated when estimating regression parameters).

Out of the 16 factors, those marked with an asterisk (*) in Table I i.e. *ISUP, COMP*, *SEQU*, *AWRD* and *SIZE,* have a statistically significant ($p < 0.05$) association with the price decision. These can be used to provide "rules of thumb" for the decision maker. For instance, the β-value of *ISUP*



	C4.5	QUEST	CART	LReg	SVMR	NNet	KNN_k	Logistic	KNN_1	Naive	Random	COCOMO
Pred30	0.843	0.841	0.794	0.757	0.743	0.724	0.724	0.698	0.694	0.561	0.542	0.341
StdMRE	0.269	0.384	0.365	0.384	1.069	0.533	0.788	0.574	0.782	0.434	1.332	1.145
MMRE	0.116	0.157	0.216	0.279	0.461	0.324	0.397	0.315	0.350	0.340	0.657	0.711
MdMRE	0.000	0.000	0.070	0.151	0.003	0.186	0.125	0.000	0.000	0.200	0.255	0.425

Fig. 2. Performance of pricing models learned by each approach. The methods are ranked by the Pred(30) measure.

suggests that upgrading projects tend to be about $70 cheaper than novel developments. For the sizing factors such as *COMP*, *SEQU* and *SIZE*, an extra 4 pages of component specification or 4 sequence diagrams or 1 KSLOC respectively tend to correspond to about $30 of the price. These rules of thumb can be used as sanity checks. When they are broken this does not mean that the price is *wrong*, but it should raise a question in the mind of the decision maker: "Why am I bucking the trend?".

V. Limitations and Future Work

Our approach is not without limitations and there remains much future work for this new development paradigm. We cannot guarantee similar performance when applied to platforms other than TopCoder. Also, our approach does not consider strategic pricing behaviour; the project manager may set a much higher price for an urgent or important task, for example. However, an estimate of a reasonable price, such as that arising from our models might, nevertheless, remain an important input to these more "political" pricing decisions.

Future work will be needed to investigate other platforms in order to propose more general factors and models for pricing software development tasks. It will also focus on the strategic analysis of pricing. Although crowdsourcing contests can be modelled as all-pay auctions [7], the previous research is based on the context of microtasks, so it may need to be revised for crowdsourcing-based software development. Future work may also consider more elaborate multi-objective prediction systems using Search Based Software Engineering to find acceptable balances between objectives [8]. Finally, TopCoder denotes a large community of global programmers. Based on platforms such as this, we intend to study the impact of social networking on software development activities.

VI. Conclusions

In order to address the problem of how to price crowdsourcing-based software development tasks we empirically analysed historical data from 5,910 TopCoder crowdsourced software development tasks to identify 16 potential price drivers. We used these price drivers to derive 9 predictive pricing models using machine learning, case-based reasoning, neural network and regression techniques. To evaluate our models we performed a leave-one-out cross-validation on all 490 projects for which full details are available on TopCoder. Our results show that high predictive quality is achievable, outperforming existing available pricing techniques and providing actionable insights for software engineers working on crowdsourced projects.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61073044, 71101138, 61003028 and National Science and Technology Major Project of China under Grant Nos. 2012ZX01039-004, 2012ZX01037-001-002.

References

[1] J. Howe, "The Rise of Crowdsourcing," *Wired*, vol. 14, pp. 1-4, 2006.

[2] Y. Singer and M. Mittal, "Pricing Tasks in Online Labor Markets", In *Proc. Human Computation*, 2011.

[3] S. Faradani, B. Hartmann, and P.G. Ipeirotis, "What's the Right Price? Pricing Tasks for Finishing on Time", In *Proc. Human Computation*, 2011.

[4] Lakhani, Karim R., David A. Garvin, and Eric Lonstein, "TopCoder (A): Developing Software through Crowdsourcing," *Harvard Business School* Case 610-032, January 2010.

[5] B.W. Boehm, *Software Engineering Economics*, Englewood Cliffs, NJ, Prentice-Hall, 1981.

[6] Shepperd, M. and S. MacDonell, Evaluating prediction systems in software project estimation. *Information and Software Technology*, 54(8): pp. 820-827, 2012.

[7] Dominic DiPalantino, and Milan Vojnovic, "Crowdsourcing and all-pay auctions," In *Proc. of the 10th ACM Conference on Electronic Commerce* , pp. 119-128, 2009.

[8] Mark Harman, "The Relationship between Search Based Software Engineering and Predictive Modeling (keynote paper)," In *Proc. 6th International Conference on Predictive Models in Software Engineering (PROMISE)*, 2010.