The Use of Microtasks in Crowdsourcing Software Development

William Simão de Deus
Federal University of Technology - Paraná (UTFPR), Brazil
william_s94@hotmail.com

José Augusto Fabri
Federal University of Technology - Paraná (UTFPR), Brazil
fabri@utfpr.edu.br

Alexandre L'Erario
Federal University of Technology - Paraná (UTFPR), Brazil
alerario@utfpr.edu.br

Abstract — Crowdsourcing (CS) is a distributed software de- velopment model in which small activities (or microtasks) of design, development, and tests are developed by employees by an online platform. Due to the possibility of aggregating specialists, reducing time and costs, companies are applying CS in their software development projects. However, there are still gaps in the literature about microtasks, there are few studies focused on the theme and the lack of taxonomy, generating several management challenges. In this form, it was conducted an analysis of 14485 microtasks recorded on a CS platform to investigate microtasks in the development of CS software. As results demonstrated a set of considerations about the use of microtasks, the risks and the impacts of the microtasks in CS projects. Finally, the benefits and percentage of successful completion of each microtask were also verified.

Keywords – Microtasks; Crowdsourcing; Software Development

I. INTRODUCTION

Crowdsourcing (CS) is a model of task development and problem-solving based on the contribution of a large group of people [1]. Several areas have adopted the CS to optimize results, in this form, CS quickly becomes a model for the realization of services, obtaining ideas and, innovation source of for contents [2]. Recently, CS has been applied to the software development using design, development or testing tasks for people who are globally dispersed [3].

CS software development has become a well-established approach and is currently finding strong support due to the creation of CS platforms dedicated to the industrial and productive software sector [4]. In this way, the literature has been constantly updated with publications to understand the benefits, challenges, and configuration of CS software development [3], [5], [6]. Because of this scenery, L'Erario et al. [7] presented the results of CS teaching and learning among undergraduate students. According to the authors, the distributed development paradigm is being updated for CS and future professionals in the area of software development need to learn this knowledge.

Despite the encouraging scenario on the industrial trend and the academic interest of CS software development, there is still a gap related to CS activities (popularly known as microtasks). Microtasks are simple CS activities with a short duration time that can be paralleled [8]. Thus microtasks employ reduced time and effort, in addition to a relatively low development cost [9]. However, although they are simple, there is a paucity of concentrated studies on the subject of microtasks in software development CS [10].

Krieger et al. [11] cite that microtasks represent a portion of the complexity in developing CS software projects. Because microtasks concentrate small portions of work, CS projects tend to have a lot of microtasks, and thus may present difficulties for the development and coordination of work [6]. In addition, there is difficulty in identifying what effectively is a microtasks and what its influence on software projects [5]. The lack of focused studies on the subject of microtask poses several challenges for the application of CS. According to Tranquilini et al. [12], online platforms end up accepting different types of microtasks and can not establish a degree or factor that classifies the difficulty of executing a microtask. In addition, the literature itself concentrates many terminologies on microtask, usually, there are works that adopt the term "micro-task" [13] or "micro task" [14]. In addition, there are authors who treat only as "CS activities" or "tasks" [15], and even the term "macro-task" has already been adopted [16]. Although all works focus on the same object of study, a common taxonomy of the term is lacking. In order to avoid parameterization disorders, the term "microtask" it was adopted in this study because it represents the most widely used term in the academic world.

Based on the previously explicit context, the objective of this work it was to analyze how microtasks are being applied in the development of CS software, empirically verifying its use. For this, we conducted a statistical analysis on a data set consisting of 14485 microtasks extracted from a CS software development platform. In the analysis, it was investigated the way of application in software projects, the main challenges, and the relation that the configuration of the CS implies on the microtask

To achieve the goal outlined, this study it was organized in six sections. The first section contextualized the theme, evidenced its importance and the gaps of the analyzed area. In the second section, a review is presented on the CS, microtasks and the platform used for extracting the data set. The third section presents a reflection on the work synergistic to this study. To elucidate the purpose of this study, the fourth section demonstrates the questions investigated. The data extraction process and the performed analysis are found in the fifth section. The final considerations of the study and the limitations of the selected approach are presented in the sixth section.

II. CROWDSOURCING

The term CS it was coined in 2006 by Howe [1] after analyzing how various organizations were developing their products. According to the author, the organizations were outsourcing activities to their own audience through an open call. Therefore, many areas have benefited from this approach, and software development has concentrated benefits due to the use of specialists, cost reduction and parallelization of activities [2]

According to Mao et al. [3], CS has become a new approach to software development that empowers online developers to develop short, independent activities proposed by a leader. Hosseini et al. [17] presented that the CS configuration uses four bases:

- **Crowdsourcer:** Leader of the project, your responsibilities are to manage, organize and coordinate project collaborators and deliverables. It is your work to keep the project intact.
- **Crowd:** Set consisting of anonymous participants. The crowd represents a heterogeneous crowd that is scattered globally. The crowd's responsibility is to develop the tasks proposed by the crowdsourcer.
- **Platform:** It is an online system that allows crowdsourcer to coordinate the activities and crowd of the project. The platform is responsible for providing information about the project, providing communication between the crowd and crowdsourcer, and identifying the contributions of each collaborator of the crowd.
- **Microtask:** It is an independent and atomized portion of work. In CS, a microtask represents a software design, development, or testing activity. A microtask is registered by the crowdsourcer on a CS platform and should be developed by the crowd.

A. Microtasks

The microtasks represent the most important link in CS, they are responsible for uniting the crowd, the platform, and the leader. The great innovation of the use of microtasks in software development resides in the atomization of the work in portions that employ the minimum of effort and time in its conclusion [8]. Microtasks can be represented by a similar concept of division of labor into small tasks

Microtasks represent a promising area of research because of the gaps that are perceived in the literature, this is reflected by the absence of a taxonomy on the microtask (microtask, micro-task, micro-task, etc.) [6], [18], [19]. Although all the authors cited above address the same theme, none explicitly define the term in common mode.

B. Platform

To conduct this work an extensive analysis it was performed on a set of microtasks. The set of microtasks used it was extracted from an online platform dedicated to the development of CS software. Due to strategic factors, the platform did not authorize its identification, in this way it will be presented in this study as platform X.

The platform X works in the CS software development area and has a large community of designers, developers, and enthusiasts in the area. The platform daily registers almost 20 microtasks. The value of the completion of each microtask is fluctuating, and unit microtasks are found worth pennies of dollars to projects with thousands of microtasks estimated at prices in excess of ten thousand dollars. This value is decided solely by crowdsourced, however, in order to minimize risks the platform X has complex mechanisms to block values/activities diverging from reality. Due to the portability of the microtasks, multinational companies are recently allocating their projects on the X platform to achieve economic and temporal benefits.

The platform X has a programmable interface that makes it possible to access the registered microtasks. The interface provided pure text based responses that were tabulated in electronic sheets. The data thus captured were treated using three methods of analysis: Manual, Computational, and Statistics.

III. SIMILAR WORKS

The literature has a set of works with results close to those obtained in this research. Thus, the most current and relevant studies for an analysis were selected. The first work it was developed by LaToza and Hoek [18] (SW01) and presented a view on CS development, with emphasis on the application of microtasks. The authors conducted an analysis and highlighted the major gaps in the area that require further studies. One of the limitations perceived in the work is the theoretical approach, there being no quantification of data or consultation of specialists.

The second work it was developed by Mao et al. [20] (SW02) and highlighted a model of recommendations for the development of CS activities. This work it was validated through experimentation and demonstrated practical results of its approach. However, the work focused on larger CS development activities known as tasks or macro tasks and may offer serious limitations when applied to the concept of microtasks.

TABLE I: COMPARISON OF RELATED WORK

Work	Approach	Limitations
(SW01)	Theoretical	Limited validation process
(SW02)	Practice	Analysis of tasks

As shown in Table 1, the first work (TR01) did not validate its conjectures with a group of experts or data quantification. While in the second study (TR02) the authors did not use the microtask-focused approach. Finally, the limitation of this study is fixed by using only one platform for data extraction.

IV. RESEARCH QUESTIONS

A. Crowdsourcing Software Development and Microtasks

The literature presents gaps on the effective way in which microtasks are applied in software development [18]. Several studies highlight the need to understand how the development of CS occurs through the microtasks [6], [21]. Based on this scenario, the first question investigated by this study refers to "How are microtasks being applied in the development of CS software?". To analyze this, it was verify the structuring, application steps in software projects and technologies employed in the microtasks.

B. Success and Failure

According to Naik [22], software projects in a traditional and distributed way tend to be complex due to several factors. However, when dealing with CS development a gap persists about which are the main reasons that make the development of a microtask unworkable. Based on this context, we postulate the second research question seeking "What are the main reasons for failure found in microtasks that make a CS project unfeasible or complex?" To answer this question we check the index of success and failure found in the microtask and analyze the types of detected failures.

C. Crowdsourcing and Microtasks

The last issue of this study is to advance the state of the art on the relationship between CS and microtasks. For this, we select two incipient aspects of the literature: i) "how the CS configuration affects the microtasks?". In this way, we investigate how the size of the crowd impacts the microtasks and the coordination process [5], [6]. And it was analyzed the "what is the complexity of CS projects?", helping to investigate how total microtasks influence the CS project [18].

V. ANALYSES

To conduct the analyses of this paper, a data set based on 14485 microtasks extracted from the platform X it was used. The microtasks were registered between 12/31/2011 and 12/15/2016 (maximum period allowed for data extraction from the programmable interface of the platform X). The data were tabulated in electronic sheets and from there they received three modes: manuals (grouping and ordering), statistics (use of trend equations) and computational (reading and analyzing from algorithms).

A. How microtasks are being applied in the development of CS software?

CS can have two modes: collaborative and competitive. In the collaborative mode, the crowd submits solutions to a particular problem and the crowdsourcer selects which solutions will be accepted. Already in the competitive mode, is considered champion the first submission made and approved. The percentage of microtasks according to CS mode is shown in Figure 1.

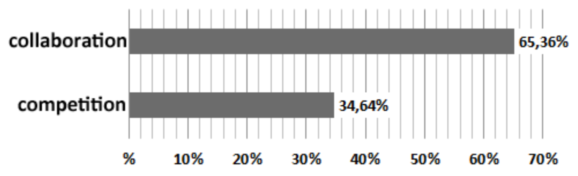

Figure 1: Crowdsourcing Mode in Microtasks

The collaborative CS demonstrates a greater use of microtasks. This value is amplified by the way microtasks are categorized. Platform X provides a set of 14 categories of microtasks: A1 competion (unit test), A3 Competition (desktop), codification, correction, creation, definition, development, diagram, first to finish, planning, requirements especification, test, test scenarios, and U2 competition (prototype).

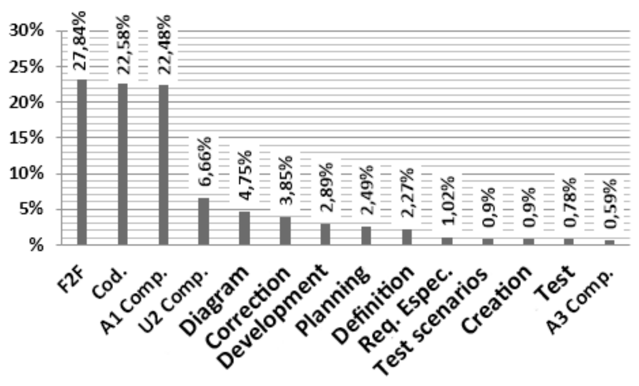

Figure 2: Categories of Microtasks

The categories of microtasks it was presented in Figure 2, the three main categories occupy more than 70% of the set of microtasks analyzed. The first to finish category has almost 30 %. The categories codification and A1 competition have, respectively, 22.58% and 22.48%. The other categories did not exceed 7% of incidence. To deepen the analysis on each category, it was identified in which stages of a project the different categories are being adopted. For this, the approach proposed by Dwarakanath et al. [4] that defined five steps in a CS:

- **Analysis:** requirements of a CS project
- **Desing:** prototypes, design and layout
- **Development:** coding routines, and/or bugs
- **Test:** test execution and error correction
- **Integration:** union of microtasks

The microtask categories of Figure 2 were also analyzed and generated in relation to the steps set forth above. The complete relationship between step x category is shown in Figure 3 by means of an area chart.

It was noticed that the integration stage is still little explored by the microtasks, being used only in the category first to finish. Microtasks facing the requirements stage are maturing and have been explored in three categories first to finish, diagram, and requirements specification. The development stage it was

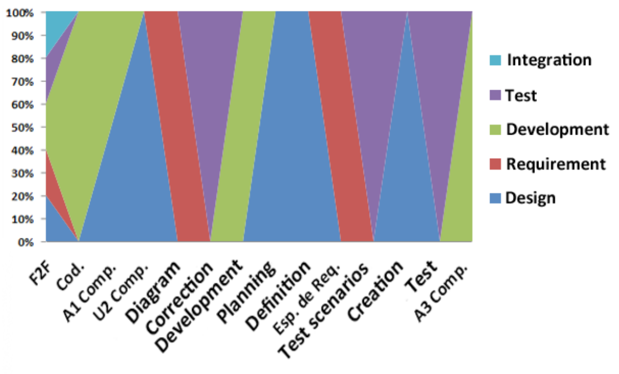

Figure 3: Microtasks and Steps in Crowdsourcing Software Development

represented in five categories first to finish, codification, A1 competition, development, and A3 competition. The the test step first to finish, correction, A1 competition, tests, and test scenarios. Finally, it was identified the design activities of a project first to finish, U2 competition, planning, definition, and creation.

After these analyzes, it was also identified which technologies were used in microtasks, in total, 8 technologies were identified and classified in:

- Service: microtasks for existing projects.
- Mobile: microtasks that employ mobile technology.
- Web: microtasks for new web projects.
- Operating System: microtasks applied at startup hardware.
- Cloud: storage microtasks in the cloud.
- Social Network: microtasks about development or testing on social networks.
- API: microtasks that offer request and responses based on JSON or XML format.
- Other: microtasks targeted technologies that do not fit into any of the previous categories.

The graph on the technologies used in the microtasks is presented in Figure 4:

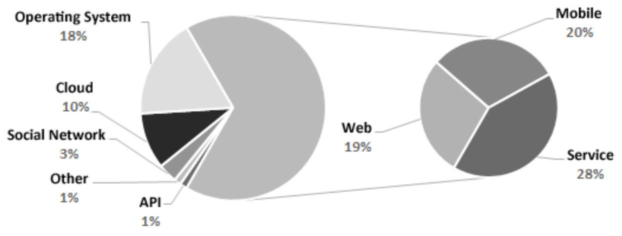

Figure 4: Microtask technologies

About 28% of microtasks are service-based, while mobile and web technologies represent respectively 20% and 19%. There is also an application of microtasks for Operating Systems with 18% and Cloud Storage 10%, enclosing the five most commonly used technologies. The detailing of such tech- nologies has had a great influence on microtasks using mobile technologies (Android and iOS), browser (JavaScript, CSS, Angular.js, Node. Js, Jquery, HTML, etc.) and programming languages (Java, C#, and .Net). The complete graph with all the detailed technologies is presented in Figure 5

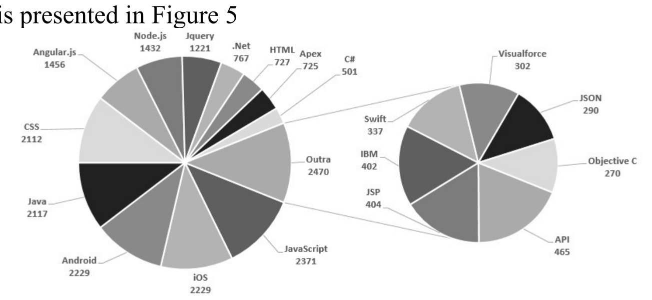

Figure 5: Details of the Microtasks Technologies

B. What are the main reasons for failure found in microtasks that make a CS project unfeasible or complex?

To initiate the analysis of the main reasons for failure, a similar approach it was applied to Mao et al. [20]. In this approach, the microtasks were added in a computational vector. This vector it was traversed several times by means of a search algorithm. As a result, at the end of processing, the algorithm exhibited the patterns most commonly used in microtask descriptions. Thus, the three main forms of support for the development of microtasks were identified: links to access complementary documentation, project code repository, and asynchronous e-mail communication. However, there is a large discrepancy in the use of each resource. Just over half the microtasks, about 56.55% dedicated the declaration of some external access link to assist the crowd in the development of the microtasks. While only 0.37% of the microtasks have some data repository and only 0.3% of the microtasks had some contact email address. One of the reasons given for failures can be drawn from the significant 32.26% share of microtasks that do not have any of the above-described types of aid. All support variations are found in Figure 6:

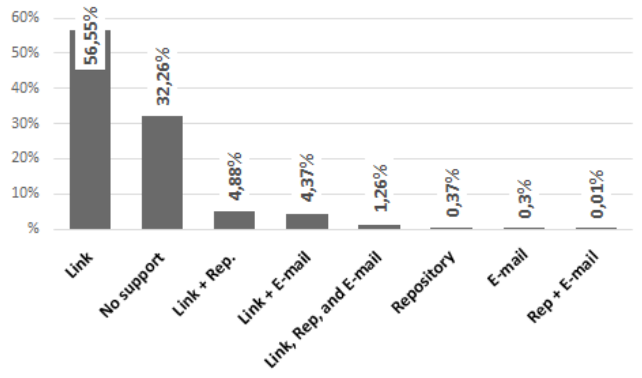

Figure 6: Microtasks Description

Supported by the chaotic scenario in which almost 1/3 of the microtasks do not have any type of support detected, the relation of success and failures of execution it was explored. Screening revealed the following classifications:

- **No submissions:** microtask without any crowd submis- sions;
- **Canceled by client:** microtasks canceled by crowd- sourcer;
- **Canceled on review:** microtasks that have been requested to be canceled;
- **Impracticable:** microtasks not feasible due to their com- plexity and time available;
- **Winner Indifferent:** Contributors who have accepted submission and have not applied for the award;
- **No records:** microtasks without any participant records.

Despite the complexity of using CS and the lack of support in microtask descriptions (showed in Figure 6), 86.07% of the microtasks analyzed were successful in development. The main reason for the cancellation it was that there were no submissions (6.99%), followed by the cancellation of client (3.27%). The other reasons do not exceed 2% incidence. The percentage of all classifications on failure reasons can be viewed by means of Figure 7.

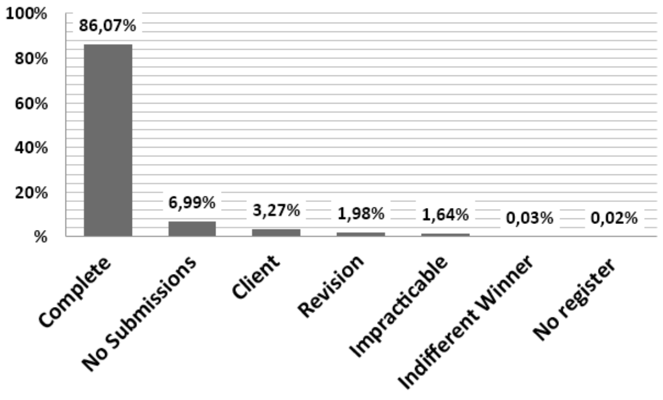

Figure 7: Success x Failure

C. Crowdsourcing and Microtasks

1) How the CS configuration affects the microtask: To investigate the connection between CS and microtasks, a step it was begun analyzing the relationship between total submissions and crowd size. To investigate this cause-and- effect relationship, a mathematical equation it was used to generate a linear trend graph based on the following formula:

$$f(x) = \sum_{i\neq 0, j=x}^{S} f(y) = \sum_{j=x}^{R} \qquad (1)$$

In which S represents the sum of microtask submissions, given in the order of i (not canceled) and j (grouped by total records). R represents the sum of the size of the crowd given by the i grouping. Therefore, the final ratio of the equation groups and determines the average trend of submissions by the size of the crowd.

Initially, it was noticed that CS is a homogeneous trend: as the total number of participants increases, the total number of

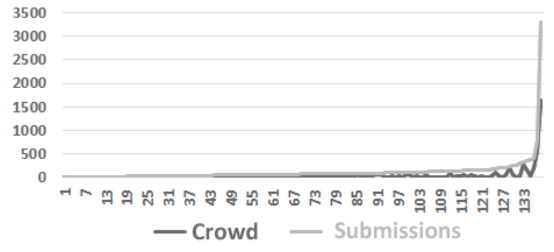

Figure 8: The relationship between the size of the crowd and the total of submission

submissions increases (see the first graph of Figure 8). How- ever, crowds with more than 100 participants demonstrated the opposite, exhibiting a heterogeneous trend. This led to the conduction of a new analysis by applying as a third variable the ratio of participation percentage. In this way the results provided by equation (1) were submitted to a new analysis, using a constant N referring to the percentage:

$$f(p) = f(y)/f(x) * N \qquad (2)$$

Where f (p) represents the division of the total submission by the size of the crowd multiplied by N = 100. Thus, it was identified that crowd with fewer participants concentrate a higher rate of participation. This trend decreases as the crowd increases to a maximum size of 50 participants. Microtasks that were developed by a crowd with more than 50 participants represent a complex and heterogeneous tendency. For, despite the increase in labor power, it is not reflected in participation. The complete graphical representation of crowd size and total submission is found in the lower graph of Figure 9.

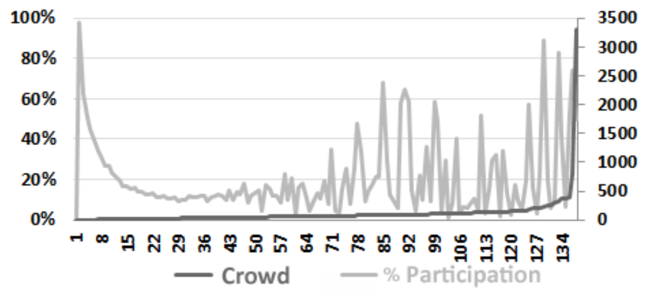

Figure 9: The detail of the percentage of participation accord- ing to the size of the crowd and the total of submissions

2) *What is the complexity of CS projects:* To analyze this question, a personalization of the Saremi and Yang [23] approach it was applied to parameterize the complexity of CS projects. In the original approach, the projects are classified into simple (<100 microtasks) and complex (>100 microtasks). To facilitate the understanding and data range, chose to classify as simple the projects with up to 50 microtasks and to establish a new interval, between 50 and 100, for projects with medium complexity.

The figure 10 demonstrates the complexity of the projects by microtasks, as can be seen, the projects considered simple have an average of 35 microtasks. However, the median is set at 10 microtasks. Medium projects, however, have a more heterogeneous dispersion, and tend to retain about 63 micro- tasks. However, the complex projects have a heterogeneous variation, grouping between 100 and 285 microtasks. Few projects have dissonant values, above 300 microtasks. Most complex projects, however, have about 110 microtasks.

To complement the answer on the complexity of the CS projects, the time spent to complete the microtasks it was analyzed. Few microtasks were completed in less than 24 hours, not reaching even 1% incidence. Meanwhile, micro- tasks completed in one day totaled only 2.11%. Most of the microtasks were completed in two days of work, totaling 41.22%. The rest of the microtasks were completed between 3 or more days and did not exceed 10% incidence. However, the microtasks completed in 30 days showed a high incidence. In the authors' perception, this fact is because the platform X provided a deadline for the conclusion of the microtasks, and with this, some crowdsourcers may have randomly defined and/or platform X has done it automatically. The graph 11 shows the complete list of days used for the conclusion of microtasks.

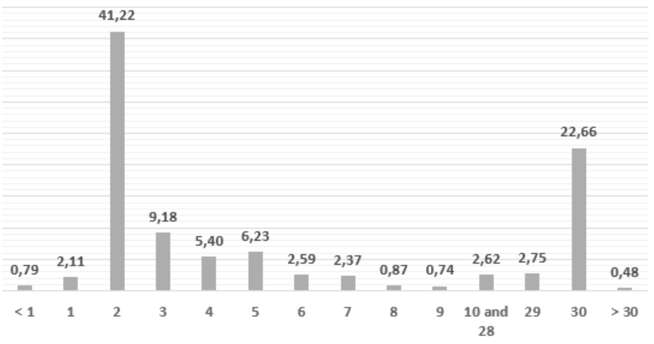

Figure 11: Microtasks Complexity in Days

VI. CONCLUSIONS

In this paper, were analyzed a set of data formed by 14485 microtasks. This data set it was based on the maximum number of microtasks provided by the X platform programmable interface. The analysis conducted in this work demonstrated that the collaborative microtasks of CS represent a greater incidence when compared to competitive microtasks. However, both modalities are being adopted in all stages of a CS software project. In addition, the three most widely used technologies in microtasks are the web, mobile, and services, and the three main reasons for microtask failures are the lack of aid in the description, lack of submissions and cancellations by the customer. It has also been identified that crowd size configuration influences submissions, and it has been found that crowds with up to 50 participants are most active in sub- missions. Already the larger crowds present a heterogeneous trend line of submissions. It was found that simple projects have an average of 35 microtasks, while medium and complex projects have, on average, 63 and 105 microtasks, respectively. Finally, most microtasks were completed in two days of work.

The original study it was developed by [24] and presented a set of considerations on the use of microtasks and the development of CS software. The original study is in the Portuguese language. This version represents an update of the original study, because of that, the literature consulted it was revised, several sentences and paragraphs were rewritten and the references updated for new works published, finally, all its content it was translated into English. In addition, this extensive version demonstrated an important graph on the detailing of the

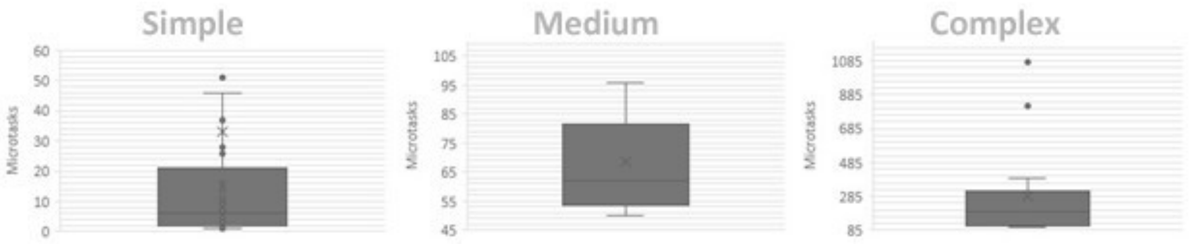

Figure 10: Microtasks Complexity

technologies present in microtasks and another graph about the days of work in microtasks.

The development of a study always presents limitations, it is emphasized that only a CS platform it was used to conduct the analysis. In addition, the data set it was limited to the maximum size allowed by the X platform programmable interface.

As future work, there is interest in conducting analysis to verify and deepen the challenges of microtasks. One possible contribution is to treat the flow of microtasks and establish metrics to assist in their management.

REFERENCES

[1] J. Howe, "The rise of crowdsourcing," Wired magazine, vol. 14, no. 6, pp. 1–4, 2006.

[2] A. Benedek, G. Molnar, and Z. Szuts, "Practices of crowdsourcing in relation to big data analysis and education methods," in Intelligent Sys- tems and Informatics (SISY), 2015 IEEE 13th International Symposium on, Sept 2015, pp. 167–172.

[3] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," Journal of Systems and Software, vol. 126, pp. 57 – 84, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121216301832

[4] A. Dwarakanath, U. Chintala, N. C. Shrikanth, G. Virdi, A. Kass, A. Chandran, S. Sengupta, and S. Paul, "Crowd build: A methodology for enterprise software development using crowdsourcing," in 2015 IEEE/ACM 2nd International Workshop on CrowdSourcing in Software Engineering, May 2015, pp. 8–14.

[5] M. Hosseini, A. Shahri, K. Phalp, J. Taylor, R. Ali, and F. Dalpiaz, "Configuring crowdsourcing for requirements elicitation," in 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS), May 2015, pp. 133–138.

[6] W. S. Deus, R. M. Barros, and A. L'Erario, "Um modelo para o gerenciamento do crowdsourcing em projetos de software," in 2016 I Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (I WASHES), Oct 2016, pp. 1–10.

[7] A. L'Erario, J. A. Fabri, R. H. C. Palácios, W. Godoy, and W. S. de Deus, "Ensino de desenvolvimento crowdsourcing em cursos de graduação: Um estudo comparativo," Iberian Conference on Information Systems and Technologies (CISTI), vol. 1, no. 12, pp. 685 – 690, 2017.

[8] T. D. LaToza, W. B. Towne, C. M. Adriano, and A. van der Hoek, "Microtask programming: Building software with a crowd," in Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, ser. UIST '14. New York, NY, USA: ACM, 2014, pp. 43–54. [Online]. Available: http://doi.acm.org/10.1145/2642918.2647349

[9] T. D. LaToza and A. V. D. Hoek, "Crowdsourcing in software engi- neering: Models, motivations, and challenges," IEEE Software, vol. 33, no. 1, pp. 74–80, Jan 2016.

[10] K.-J. Stol and B. Fitzgerald, "Researching crowdsourcing software development: Perspectives and concerns," in Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering, ser. CSI-SE 2014. New York, NY, USA: ACM, 2014, pp. 7–10. [Online]. Available: http://doi.acm.org/10.1145/2593728.2593731

[11] M. Krieger, E. M. Stark, and S. R. Klemmer, "Coordinating tasks on the commons: Designing for personal goals, expertise and serendipity," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 1485– 1494. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518927

[12] S. Tranquillini, F. Daniel, P. Kucherbaev, and F. Casati, "Modeling, enacting, and integrating custom crowdsourcing processes," ACM Trans. Web, vol. 9, no. 2, pp. 7:1–7:43, May 2015. [Online]. Available: http://doi.acm.org/10.1145/2746353

[13] L. Jiang, C. Wagner, and B. Nardi, "Not just in it for the money: A qualitative investigation of workers' perceived benefits of micro- task crowdsourcing," in 2015 48th Hawaii International Conference on System Sciences, Jan 2015, pp. 773–782.

[14] U. Hassan, M. Bassora, A. H. Vahid, S. O'Riain, and E. Curry, "A collaborative approach for metadata management for internet of things: Linking micro tasks with physical objects," in 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, Oct 2013, pp. 593–598.

[15] M. Hossain, "Crowdsourcing: Activities, incentives and users' motiva- tions to participate," in 2012 International Conference on Innovation Management and Technology Research, May 2012, pp. 501–506.

[16] J. Cheng, J. Teevan, S. T. Iqbal, and M. S. Bernstein, "Break it down: A comparison of macro- and microtasks," in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 4061–4064.[Online]. Available: http://doi.acm.org/10.1145/2702123.2702146

[17] M. Hosseini, K. Phalp, J. Taylor, and R. Ali, "The four pillars of crowdsourcing: A reference model," in 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), May 2014, pp. 1–12.

[18] T. D. LaToza and A. v. d. Hoek, "A vision of crowd development," in 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, May 2015, pp. 563–566.

[19] M. Hosseini, A. Shahri, K. Phalp, and R. Ali, "Recommendations on adapting crowdsourcing to problem types," in 2015 IEEE 9th Inter- national Conference on Research Challenges in Information Science (RCIS), May 2015, pp. 423–433.

[20] K. Mao, Y. Yang, Q. Wang, Y. Jia, and M. Harman, "Developer recommendation for crowdsourced software development tasks," in 2015 IEEE Symposium on Service-Oriented System Engineering, March 2015, pp. 347–356.

[21] A. Suganthy and T. Chithralekha, "Application of crowdsourcing insoftware development," in 2016 International Conference on Recent Trends in Information Technology (ICRTIT), April 2016, pp. 1–6.

[22] N. Naik, "Crowdsourcing, open-sourcing, outsourcing and insourcing software development: A comparative analysis," in 2016 IEEE Sympo- sium on Service-Oriented System Engineering (SOSE), March 2016, pp. 380–385.

[23] R. L. Saremi and Y. Yang, "Empirical analysis on parallel tasks in crowd- sourcing software development," in 2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW), Nov 2015, pp. 28–34.

[24] W. S. de Deus, J. A. Fabri, and A. L'Erario, "The use of microtasks in crowdsourcing software development," in 2017 12th Iberian Conference on Information Systems and Technologies (CISTI), June 2017, pp. 1–6.