B. *RQ2: What is the extent of rework effort required to resolve SATD in open-source projects?*

Table 2 indicates the estimated rework effort (measured in average commented LOC per SATD prone source file of each system) for the maintenance team to resolve these SATD within the source files of the respective systems studied. It should be noted that *Req't* and *Docu* in Table 2 denote Requirement and Document debts respectively. From the perspective of considering all the five classes of debts, it was realized that design debt required substantial rework effort as elaborated in Table 2. Thus, the rework effort for resolving design debt in AgroUML is 7.9, Chromium is 17.1, Eclipse is 11.8 and lastly, Apache is 12.6 commented LOC on average per SATD prone source file. Similarly, test and defect debts were also of key interest in this study which needed rework apart from design debts. These two debts even though known by the development team that it will lead to long-term bugs upon release were left unfixed. This we believe will be due to the time-to-market constraint as mentioned by Fernández-Sánchez et al. [9].

Based on results from Table 2, there is no unique pattern in relation to the SATD rework effort and the size of the open-source projects. A typical example is seen in Eclipse and Apache. Even though Eclipse has 437,640 commented LOC much larger than that of Apache with 54,295 (Table 1), the amount of SATD rework effort for Eclipse is 11.8 as compared to 12.6 in Apache (Table 2). It can be seen that the rework effort estimation of about 13-32 commented LOC on average per SATD prone source file across the selected projects could affect the quality of the software product.

TABLE 2: REWORK EFFORT FOR RESOLVING SATD

SATD Class	**Rework Effort for Projects**			
	Agro	**Chromium**	**Eclipse**	**Apache**
Req't	0.7	2	4.4	3.9
Design	7.9	17.1	11.8	12.6
Testing	3.1	4.6	5.1	7.3
Defect	1.6	5.3	3.1	5.6
Docu.	0	0.4	0.3	2.2
Total	13.3	29.4	24.7	31.6

IV. THREATS TO VALIDITY

The first threat to validity in this study is the use of well-commented open-source project datasets. This constraint might not be a representative sample of the total population of open-source projects since not all projects are well-commented. Thus, the findings of this study cannot confidently be generalized. The selected projects used are popular and large in size. Therefore, the examination of all the developers' comments from the projects with the intention of resolving the self-admitted technical debt (SATD) problem can form a good foundation for researchers to conduct more in-depth studies in this field. Secondly, the list of SATD indicators used from previous study might not be a generalized representation of all SATD in the software development and maintenance environment. Since this study focused on source code comment analysis, we were constraint of gathering more information especially from industry to validate the results obtained.

V. CONCLUSION

In this study, we performed an exploratory analysis with a proposed text mining approach on source code comments of four open-source projects. With the help of transforming the source code comments into term weights, we were able to estimate the rework effort for fixing these debts. This study addressed two main research questions:

RQ1: What is the dominant class of self-admitted technical debt?

Results from the study indicate that out of all the five classes of SATD, design debts (56.5% - 67.5%) is the predominant class of SATD for all the four systems.

RQ2: What is the extent of rework effort required to resolve SATD in open-source projects?

The result of this study indicate that rework effort of between 13 and 32 commented LOC on average per SATD prone source file will have be addressed in order to fix the SATD. In order to improve the long term quality of the software, it is essential that developers are encouraged to avoid SATD.

The proposed approach is a novel technique which can assist in the estimation of rework effort needed to fix SATD tasks that demands rework.

In going forward, we intend to validate our approach based on industrial case studies and different versions of open-source datasets to facilitate result generalization.

REFERENCES

[1] A. Potdar, and E. Shihab. "An exploratory study on self-admitted technical debt." *2014 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2014.

[2] F. Zhao, Y. Tang, Y. Yang, H. Lu, Y. Zhou, and B. Xu. "Is Learning-to-Rank Cost-Effective in Recommending Relevant Files for Bug Localization?." *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*. IEEE, 2015.

[3] E. S. Maldonado, and E. Shihab. "Detecting and quantifying different types of self-admitted technical Debt." *Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on*. IEEE, 2015.

[4] C. D. Manning, P. Raghavan and H. Schütze. *Introduction to information retrieval*. Vol. 1. No. 1. Cambridge: Cambridge university press, 2008.

[5] W. Sultan, E. Shihab, and L. Guerrouj. "Examining the Impact of Self-admitted Technical Debt on Software Quality." *23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering*. IEEE, 2015.

[6] Y. Padioleau, T. Lin, and Z. Yuanyuan. "Listening to programmers—Taxonomies and characteristics of comments in operating system code." *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on Software Engineering*. IEEE, 2009.

[7] H. J. Harrington, "Poor-Quality Cost: Implementing, Understanding, and Using the Cost of Poor Quality (Quality and Reliability)." (1987).

[8] A. Chatzigeorgiou, et al. "Estimating the breaking point for technical debt." *Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on*. IEEE, 2015.

[9] C. Fernández-Sánchez, J. Garbajosa, and A. Yagüe. "A framework to aid in decision making for technical debt management." *Managing Technical Debt (MTD), 2015 IEEE 7th International Workshop on*. IEEE, 2015.

[10] S. Fabrizio. "Machine learning in automated text categorization."*ACM computing surveys (CSUR)* 34.1 (2002): 1-47.

[11] M. Bhardwaj, and A. Rana. "Impact of Size and Productivity on Testing and Rework Efforts for Web-based Development Projects." *ACM SIGSOFT Software Engineering Notes* 40.2 (2015): 1-4.