of rework between P2-P1 = 6 = (P2-T) – (P1-T) = (91) – (85) = 6; likewise, effort difference becomes P2-P1 = (-9) – (82) = -91.

DOI : 10.21817/indjcse/2022/v13i2/221302(

p#	T		T-no_user involvement	P1				P1-partial_user involvement	P2	UAT	total P2	P1-T	P2-T	P2-P1
	wk 1-15	UAT		wk 1-5	wk 6-12	wk 13-14	UAT							
0	28	16	44	15	7	0	9	31	33	7	40	-13	-4	9
1	7	0	7	11	5	12	5	33	29	3	32	26	25	-1
2	4	0	4	9	4	8	4	25	24	2	26	21	22	1
3	10	0	10	12	6	7	7	32	37	3	40	22	30	8
4	12	0	12	14	5	9	8	36	40	3	43	24	31	7
5	11	25	36	17	4	14	6	41	21	2	23	5	-13	-18
												85	91	6

Table 4. Frequency of rework results with and without user involvement.

p#	T		T-no_user involvement	P1				P1-partial_user involvement	P2	UAT	total P2	P1-T	P2-T	P2-P1
	wk 1-15	UAT		wk 1-5	wk 6-12	wk 13-14	UAT							
0	51	84	135	30	0	0	35	65	58	16	74	-70	-61	9
1	18	0	18	18	16	19	18	71	42	11	53	53	35	-18
2	11	0	11	15	22	11	12	60	30	7	37	49	26	-23
3	17	0	17	14	19	10	21	64	39	10	49	47	32	-15
4	22	0	22	19	10	14	24	67	40	8	48	45	26	-19
5	14	92	106	11	12	24	17	64	32	7	39	-42	-67	-25
												82	-9	-91

Table 5. COSMIC function point measure of effort.

Model	Ideal time	Velocity	Load factor
T	3	0/-	5/3
P1	2	46	3/2
P2	1	83	2/1

Table 6. Time analysis.

Table 6 shows time analysis of all models based on the definitions given earlier. Model T exhibits the highest ideal time since the developers could work continuously without any interruption from the user, except meeting on work-in-progress update. However, the functional and non-functional requirements were inherently different from user stories, thereby the velocity was inapplicable. Developer of Model P1 only interacted with user at the beginning and final stages kept the velocity somewhat less than that of P2, where user injected change requests regularly. As a consequence, the velocity was relatively high. Finally, Model P2 showed the highest load factor as the developers had to take care of change requests frequently.

5. Discussions

The results reveal some beneficial aspects that can be drawn below.

5.1. *Practitioner's benefits*

The amount of rework is quite heavy at the beginning and the end of the project under T. This is predictable since the emphasis must be placed on SRS to get a correct project outcome. A stringent UAT must also be exercised at the end to reach the stated outcome in SRS, causing high rework effort inevitably since there is no user involvement in between. Nonetheless, the overall of rework frequency and effort for the project are moderate as shown in box-plot 'A' of Figure 5 and 6.

P1 is in no different situation. The coarse-grained user involvement in architectural design (wk 1-5) and testing (wk 13-14) stages cause slight increase in rework frequencies, while the middle development stage (wk 6-12) remains low. The overall project numbers are relatively even with T. This is shown in box-plot 'B' of Figure 5 and 6 which enclose all the above three stages.

Model P2 exhibits considerable lower rework than those of P1 and the T models as shown in box-plot 'C' of Figure 5. This is the direct effect of user involvement. It is apparent from Figure 6 that these numbers are also proportional to the project effort. The higher the effort, the higher the development cost.

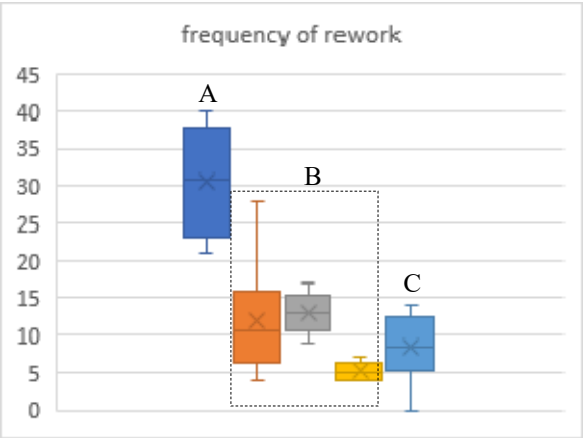

Fig. 5. Frequency of rework.

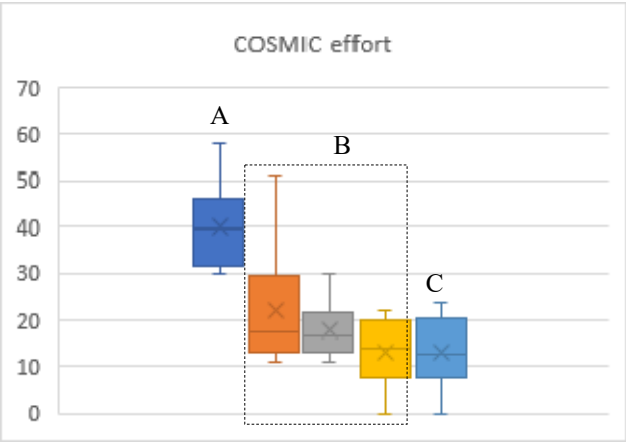

Fig. 6. COSMIC effort.

If we focus closer to the end of project, we can see that the UAT plots turn the table around. Frequency of rework and effort of model T exploded as demonstrated in box-plot 'A' of Figure 7 and 8, respectively. More user involvement under P1 than T reduces UAT rework and effort considerably as shown in box-plot 'B' in the figures. The best outcome turns out as predicted—under P2 with full user involvement. From users and software product point of view, these numbers mean higher user satisfaction.

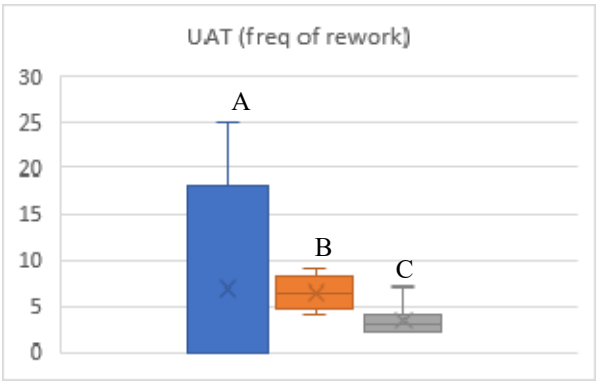

Fig. 7. UAT frequency of rework.

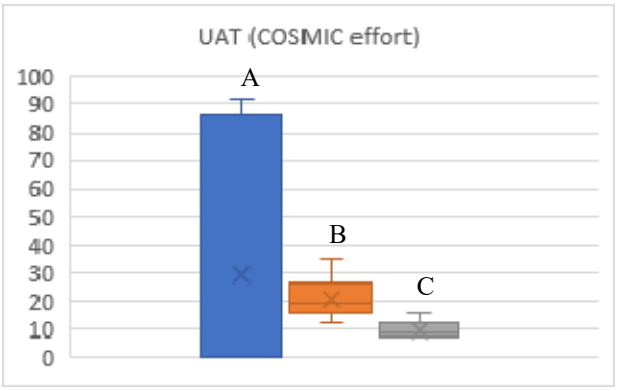

Fig. 8. UAT COSMIC effort.

5.2. *Small production*

We requested preliminary the local developers to exercise P1 since they employed similar coarse-grained waterfall-agile method. Table 7 shows the effort expended by a 3-person team working on a small sub-system of a project. The project manager served as the user of the team.

p#	P1				total
	wk1-5	wk6-12	wk13-14	UAT	
0	118	65	0	75	258
1	86	48	89	49	272
2	72	37	71	36	216
3	91	58	64	63	276
4	110	47	76	72	305
5	124	39	110	57	330

Table 7. COSMIC measure of effort.

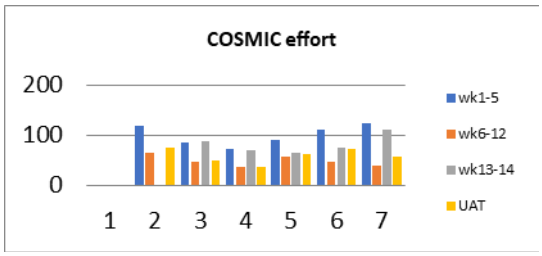

Fig. 9. COSMIC effort.

Figure 9 shows the plot of output COSMIC effort which reflects unsurprising results similar to those of the experiments. The initial requirements setup (wk1-5) took the highest effort expended, followed by corrective effort during wk13-14 in preparation for UAT step. As a consequence, the practicality of user involvement in proper level of activity breakdown helped reduce the effort as proposed by this study.

6. Conclusion

This study has reaffirmed the benefits from two aspects of the proposed method, i.e., user involvement and level of activity breakdowns in software development process model. We did not target for any innovative method since we focused on solving the problems for software developers to obtain some practical ways of working that constituted the bread-and-butter of software project. The results were satisfactory in both qualified computer science seniors and a small professional software development team that contributed to slightly higher user acceptance.

As far as time analysis was concerned, the three factors ideal, time, velocity, and load factor were not quite consistently contributed to any models. No single model out-performed others for adopters to decide what to follow.

One important result obtained from this study is domains of applicability. Care must be taken in adapting the results to the appropriate team size, project scale, and development model since agile may not fit some development/application settings. At any rate, future work should focus on the quality of software deliverables that can be directly or indirectly affected by the degree of user involvement.

References

[1] Abrahamsson, P.; Hanhineva, A.; Hulkko, H.; Ihme, T.; Jäälinoja, J.; Korkala, M.; Koskela, J.; Kyllönen, P.; Salo, O. (2004): Mobile-D: An Agile Approach for Mobile Application Development, OOPSLA'04, Vancouver, British Columbia, Canada.

[2] Albrecht, A.; Gaffney, J **.J.** (1983): Software function, source lines of code, and development effort prediction **:**A software science validation, IEEE Transactions on Software Engineering,9(6), pp **.**639–648**.**

[3] Anwer, F.; Aftab, S.; Waheed, U.; Muhammad, S. S. (2017): Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey, International Journal of Multidisciplinary Sciences and Engineering, 8(2), pp. 1-9.

[4] Banker, R**.**; Kauffman, R**.**; Kumar, R**.** (1992): An empirical test of object-based output measurement metrics in a computer aided software engineering (case) environmen, Journal of Management Information Systems, 8(3), pp **.**127–150**.**

[5] Begel, A.; Nagappan, N. (2007): Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study, First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007).

[6] Boehm, B.; Abts, C.; Brown, A.; Chulani, S.; Clark, B.; Horowitz, E.; Madachy, R.; Reifer, D.; Steece, B. (2000). *Software Cost Estimation with COCOMO II*, Upper Saddle River, NJ :Prentice Hall PTR.
[7] Brede Moe, N.; Rolland, K.; Dingsøyr, T. (2018): To schedule or not to schedule? An investigation of meetings as an inter-team coordination mechanism in large-scale agile software development, International Journal of Information Systems and Project Management, 6(3), pp. 45-59.
[8] Buglione, L.; Abran, A. (2007): Improving Estimations in Agile Projects: Issues and Avenues, Proceedings Software Measurement European Forum (SMEF), pp. 265-274.
[9] Drury-Grogan, M. L. (2021): The Changes in Team Cognition and Cognitive Artifact Use During Agile Software Development Project Management. Project Management Journal, 52(2), pp. 127–145.
[10] Gemino, A.; Reich, B. H.; Serrador, P. M. (2021): Agile, Traditional, and Hybrid Approaches to Project Success: Is Hybrid a Poor Second Choice? Project Management Journal, 52(2), pp. 161–175.
[11] Glaiel, F.; Moulton, A.; Madnick, S. (2013): Agile Project Dynamics: A System Dynamics Investigation of Agile Software Development Methods, Working Paper CISL# 2013-05, Composite Information Systems Laboratory (CISL), Sloan School of Management, MIT, Cambridge, MA 02142, pp. 1-29.
[12] Głodziński, E. (2019): Performance measurement of complex project: framework and means supporting management of project-based organizations, International Journal of Information Systems and Project Management, 7(2), pp. 21-34.
[13] Guideline for Sizing Business Application Software—The COSMIC Functional Size Measurement Method, versions 4.0.1/4.0.2 (2017): https://cosmic-sizing.org/wp-content/uploads/2017/05/ COSMIC-Method-v4.0.1-Bus-App-Guideline-v1.3-1.pdf.
[14] Kim, H. K. (2013): Architecture for Adaptive Mobile Applications, International Journal of Bio-Science and Bio-Technology, 5(5), pp. 197-210.
[15] Lee, G.; Xia, W. (2010): Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility, MIS Quarterly, 34(1), pp. 87-114.
[16] Matharu, G. S.; Singh, H.; Mishra, A.; Upadhyay, P. (2015): Empirical Study of Agile Software Development Methodologies: A Comparative Analysis, ACM SIGSOFT Software Engineering Notes, 40(1), pp. 1-6.
[17] Salman, I.; Misirli, A.; Juristo, N. (2015): Are students representatives of professionals in software engineering experiments?, 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, Italy, 2015, pp. 666-676.
[18] Shamshurin, I.; Saltz, J. S. (2019): Using a coach to improve team performance when the team uses a Kanban process methodology, International Journal of Information Systems and Project Management, 7(2), pp. 61-77.
[19] Sophatsathit, P. (2020): Software Analytics for Manual Activities using Developer Work Elements, Journal of Information Processing, Information Processing Society of Japan, 28, pp. 279–291.
[20] Wiesche, M. (2021): Interruptions in Agile Software Development Teams, Project Management Journal, 52(2), pp. 210–222.

Authors Profile

Nalinee Sophatsathit, receives her bachelor degrees in Communication Arts from Sukhothai Thammathirat University and Computer Science from Phetchaburi Teacher's College, Masters in Computer Science from College of Engineering, Chulalongkorn University, and Ph.D. in Innovation Management from Innovation College, Suan Sunandha Rajabhat University. She is a faculty member in Computer Science Program, Faculty of Science and Technology, Suan Sunandha Rajabhat University. Her research interests are innovation in IT and software usability. She can be reached at nalinee.so@ssru.ac.th.