ANALISIS KESIAPAN TIM DALAM MENGADOPSI SOFTWARE CRAFTSMANSHIP: STUDI KASUS PENGEMBANGAN APLIKASI RAPOR ONLINE

Ahmadi[1*], Arie Surachman[2], Alpi Mahisha Nugraha[3]
[1,2,3]Program Studi Teknik Informatika, Universitas Indraprasta PGRI
ahmadi@unindra.ac.id[1], ariesurachmanmkom@gmail.com[2], alpi.mahisha@gmail.com[3]

Submitted January 21, 2025; Revised September 16, 2025; Accepted November 25, 2025

Abstrak

Penelitian ini mengevaluasi kesiapan tim pengembang berskala kecil dalam menerapkan *software craftsmanship* pada pengembangan aplikasi rapor *online* untuk pendidikan anak usia dini menggunakan metode *Extreme Programming* (XP). Pendekatan ini dipilih untuk melihat kapasitas tim dalam menyajikan perangkat lunak yang berkualitas seara teknis. Tujuan yang ingin dicapai adalah melihat tim dalam mengadopsi praktik – praktik yang ada di *extreme programming* dalam menerapkan nilai – nilai *software craftsmanship*. Pengembangan perangkat lunak melalui dua kali iterasi XP, dengan sebelas (11) *user stories*. Hasil penelitian menunjukkan bahwa hanya sebagian praktik seperti *standard coding* dan *test-driven development* pada sebagian fungsi yang dapat dijalankan. Sedangkan *pair programming* belum berjalan optimal. Temuan utama penelitian ini adalah bahwa kesiapan tim menjadi faktor kunci keberhasilan penerapan *software craftsmanship*. Kontribusi penelitian ini terletak pada penyusunan kerangka evaluasi kesiapan tim dalam mengadopsi *software craftsmanship*, khususnya pada konteks pengembangan perangkat lunak oleh tim kecil di lingkungan pendidikan.

Kata Kunci: *Software Craftsmanship*, *Extreme Programming*, *Test-Driven Development*, *Standard Coding*, *Pair Programming*.

Abstract

This study assesses the readiness of a small-scale development team to implement software craftsmanship in the development of an online report card application for early childhood education, utilizing the Extreme Programming (XP) method. This approach was chosen to overcome the quality limitations that often arise in agile-based development. The development process was carried out through two XP iterations with eleven (11) user stories. The results of the study indicate that only some practices, such as standard coding and test-driven development in some functions, can be implemented. Meanwhile, pair programming has not been running optimally. The main finding of this study is that team readiness is a key factor in the successful implementation of software craftsmanship. The contribution of this study lies in the development of an evaluation framework for team readiness in adopting software craftsmanship, especially in the context of software development by small teams in an educational environment.

Keywords: *Software Craftsmanship*, *Extreme Programming*, *Test-Driven Development*, *Standard Coding*, *Pair Programming*

1. PENDAHULUAN

Latar belakang penelitian ini dibangun di atas kebutuhan sekolah taman kanak – kanak yang membutuhkan aplikasi rapor online dalam rangka melakukan digitalisasi dalam proses pembelajaran. Tim pengembang yang bertanggung jawab terdiri dari tiga anggota. Sehingga membutuhkan metode yang tepat untuk mengembangkan aplikasi dengan cepat dan berkualitas. Oleh karena itu metode yang digunakan adalah metode *agile*. Namun penerapan metode *agile* dalam pengembangan perangkat lunak tidak cukup, karena banyak temuan pengembangan dengan metode *agile* menghasilkan perangkat lunak yang tidak berkualitas, pendekatan *software*

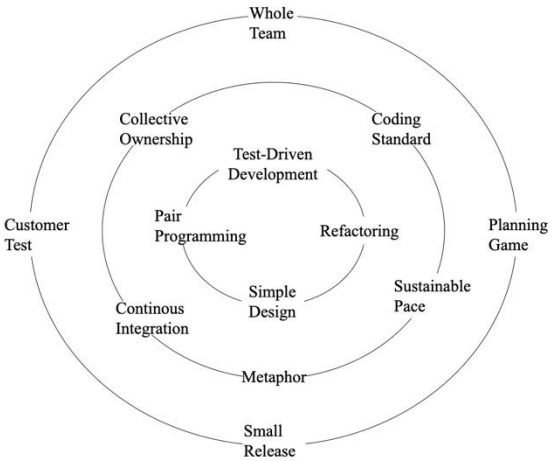
craftsmanship diperlukan dalam melengkapi metode *agile* [1]. Metode *agile* yang digunakan adalah *extreme programming* karena dapat dikerjakan dengan tim yang kecil (3 orang) [2], [3], [4], [5], selain itu merupakan metode *agile* yang disarankan oleh komunitas *software craftsmanship* [1], [6].

Namun terdapat permasalahan yang dihadapi bahwa tim pengembang belum memiliki pengalaman dalam menerapkan *software craftsmanship*. Oleh karena itu perlu melakukan analisis kesiapan dari tim pengembang dalam kesiapan mengadopsi *software craftsmanship*. Kesiapan tim dalam mengadopsi *software craftsmanship* menjadi penting karena kapasistas tim pengembang merupakan faktor penting dalam kesuksesan penerapan metode *agile* itu sendiri [7], [8], [9], [10], [11].

Software craftsmanship adalah pendekatan yang menekankan pada kualitas dan keunggulana teknis [12], [13], [14]. *Software craftmanship* bukan merupakan pendekatan yang menggantikan metode *agile*, melainkan menyempurnakan metode agile tersebut. Oleh karena itu hadirlah *software craftsmanship manifesto* yang bertujuan untuk melengkapi *agile manifesto* [1], [2], [3], [15]. Sehingga dalam menerapan *software crafstmanship* tetap membutuhkan metode *agile* sehingga saling melengkapi. Metode *agile* yang diusung untuk mempercepat proses dan *software craftsmanship* yang menjamin kualitas perangkat lunak.

Extreme programming merupakan salah satu metode *agile* yang disarankan dalam menerapkan nilai – nilai *software craftsmanhsip* [6]. *Extreme programming* menjadi metode *agile* yang banyak disarankan karena metode tersebut menekankan pada keunggulan teknis sehingga penjaminan kualitas perangkat lunak terjaga. Gambar 1 adalah praktik – praktik yang ditekankan pada metode *extreme programming* untuk penjaminan kualitas tersebut. Sehingga pada penelitian ini akan meninjau tim pengembang dalam penerapan praktik – praktik tersebut dalam proses pengembangan aplikasi.

Tujuan yang ingin dicapai adalah melakukan analisis kesiapan tim pengembang serta menyiapkan rekomendasi yang diperlukan kedepannya. Batasan pada analisis yaitu mencakup pada tiga praktik utama yaitu *test-driven development*, *coding standard*, dan *pair programming*. Alasan memilih tiga praktik tersebut untuk memudahkan adaptasi *software crafstmanship* serta menjadi representasi dari penerapan teknik pemrograman dengan baik, komunikasi yang jelas, dan kerjasama antara tim pengembang [2], [3], [16], [17]. Kondisi tersebut tidak terlepas dari tantangan dalam mengadopsi metode *agile* antara lain belum adanya panduan dari penerapan praktik – praktik *agile* dan faktor sumber daya manusia [18].



Sumber : [3]

Gambar 1. Praktik - Praktik Extreme Programming

2. METODE PENELITIAN

Sumber : dokumen pribadi

Gambar 2. Metode Penelitian

Tahap pertama dari metode penelitian ini diawali dengan melakukan studi literatur terhadap metode *agile*, *extreme programming*, dan *software craftsmanship*. Tujuan yang ingin dicapai adalah menggali informasi tentang bagaimana menerapkan metode *agile* khususnya dalam tim pengembang yang kecil. Selain itu menggali pendekatan yang tepat dalam pengembangan untuk mencapai aplikasi yang berkualitas. Oleh karena itu disimpulkan bahwa adopsi *software craftsmanship* dengan menggunakan *extreme programming* menjadi solusi yang akan digunakan.

Tahap kedua adalah menyimpulkan nilai – nilai yang digunakan untuk melakukan evaluasi terhadap kesiapan tim pengembang dalam adopsi tersebut. Berdasarkan hasil studi literatur disimpulkan bahwa penggunaan tiga praktik dari *extreme programming* menjadi acuan penilaian. Ketiga praktik tersebut adalah *standard coding*, *test-driven development*, dan *pair programming*. Tabel 1 menjadi acuan terhadap penggunaan praktik tersebut. Penilaian menggunakan angka dengan rentang 1 – 5, dengan nilai 1 menandakan praktik sangat sedikit digunakan atau tidak sama sekali dan nilai 5 menandakan praktik banyak digunakan atau secara menyeluruh.

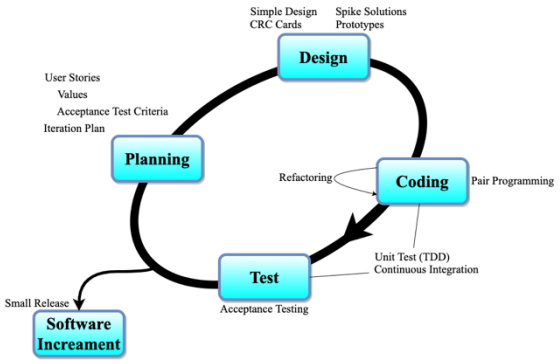
Tahap ketiga adalah pengembangan aplikasi rapor dengan menggunakan metode *extreme programming*. Gambar 3 merupakan metode *extreme programming* yang akan digunakan yang terdiri dari lima (5) fase yang terdiri dari *planning*, *design*, *coding*, *test*, dan *software increament*. Masing – masing fase memiliki praktik – praktik yang dapat dikerjakan. Kelima fase tersebut dilakukan setiap iterasi *extreme programming* dengan yang dikerjakan selama dua sampai tiga pekan per iterasi [2], [3], [19].

Tabel 1. Evaluasi Penerapan Praktik Extreme Programming

Iterasi	Standard Coding	Test-Driven Development	Pair Programming
1.	(1 – 5)	(1 – 5)	(1 – 5)
2.	(1 – 5)	(1 – 5)	(1 – 5)

Sumber : dokumen pribadi

Tahap keempat adalah melakukan evaluasi terhadap kesiapan tim pengembang dalam mengadopsi nilai – nilai *software craftsmanship* dan menjalankan metode *extreme programming*. Untuk penilaian adopsi *software craftsmanship* yaitu dengan menilai penerapan tiga praktik pada Tabel 1. Penilaian dari penerapan metode *exteme programming* yang melihat hasil penerapan dalam kelima fase tersebut.



Sumber : [19]

Gambar 3. Metode Extreme Programming

3. HASIL DAN PEMBAHASAN

Pengembangan aplikasi dilakukan melalui dua kali iterasi *extreme programming*. Dengan penjelasan berikut.

Iterasi Pertama

Fase *planning* pada iterasi pertama melakukan analisis kebutuhan aplikasi dan menuangkannya menjadi *user stories* yang terdiri dari sebelas (11) *user stories* yang akan dikerjakan. Setiap *user story* dilakukan analisis terhadap *value* yang akan disajikan dan skenario pengujiannya dalam *acceptance test criteria*. Selanjutnya melakukan perencanaan pengembangan dengan memilih delapan (8) *user stories* yang akan dikerjakan dalam iterasi pertama.

Fase *design* pada iterasi pertama yaitu melakukan analisis kebutuhan yang dituangkan menjadi *usecase diagram, class diagram*, dan *prototype*. *Prototype* merupakan visualisasi dari aplikasi yang akan dikembangkan. *Usecase diagram* merupakan visualisasi dari fungsi yang ada di dalam aplikasi. *Class diagram* merupakan visualisasi dari *database* dari aplikasi. Hasil rancangan dari iterasi pertama terdiri dari delapan (8) *usecase diagram*, enam belas (16) *class diagram*, dan dua (2) *prototype* untuk *web apps* dan *mobile apps*. Dengan hasil rancangan tersebut menjadi panduan bagi tim dalam mengembangkan aplikasi pada fase *coding* dan *test*.

Fase *coding* dan *testing* adalah dua fase yang dilakukan secara bersamaan pada metode *extreme programming*. Pada iterasi pertama ini bertujuan untuk mengembangkan aplikasi dari delapan (8) *usecase diagram* yang dirancang. Penerapan praktik seperti *standard coding*, *test-driven development*, dan *pair programming* diobservasi. Hasil observasi menunjukkan bahwa *standard coding* merupakan praktik yang dilakukan secara menyeluruh karena banyak referensi yang bisa dicari di internet. Sehingga pada iterasi ini *standard coding* dinilai dengan nilai lima (5). Namun untuk *test-driven development* tim pengembang belum terbiasa dan baru mencoba beberapa fungsi. Sehingga pada iterasi ini *test-driven development* dinilai dengan nilai tiga (3). Sedangkan *pair programming* masih belum bisa dilakukan karena belum adanya kesepakatan soal waktu dalam mengembangkan aplikasi. Sehingga pada iterasi ini *pair programming* dinilai dengan nilai satu (1).

Fase berikutnya adalah *software increament* yaitu dengan melakukan *deployment* terhadap aplikasi yang sudah dikembangkan. Tujuan dari fase ini adalah rilis aplikasi dalam versi kecil sehingga fungsi – fungsi yang disepakati pada fase *planning* bisa digunakan oleh pengguna.

Iterasi Kedua

Fase *planning* pada iterasi kedua adalah melakukan *review* dari proses pengembangan pada iterasi pertama dan memilih *user story* yang akan dikerjakan. Karena delapan (8) *user stories* sudah dikerjakan, maka tiga (3) *user stories* dipilih dalam pengembangan aplikasi iterasi kedua.

Fase *design* pada iterasi kedua yaitu melakukan analisis kebutuhan yang dituangkan menjadi *usecase diagram, class diagram*, dan *prototype*. Hasil rancangan dari iterasi pertama terdiri dari tiga (3) *usecase diagram*, lima (5) *class diagram*, dan satu (1) *prototype* untuk *web*.

Fase *coding* dan *testing* adalah dua fase yang dilakukan secara bersamaan pada metode *extreme programming*. Pada iterasi pertama ini bertujuan untuk mengembangkan aplikasi dari tiga (3) *usecase diagram* yang dirancang. Penerapan praktik seperti *standard coding*, *test-driven development*, dan *pair programming* diobservasi. Hasil observasi menunjukkan bahwa *standard coding* merupakan praktik yang dilakukan secara

menyeluruh karena banyak referensi yang bisa dicari di internet. Sehingga pada iterasi ini *standard coding* dinilai dengan nilai lima (5). Namun untuk *test-driven development* tim pengembang tidak diterapkan. Sehingga pada iterasi ini *test-driven development* dinilai dengan nilai tiga (1). Sedangkan *pair programming* masih belum bisa dilakukan karena belum fokus pada *coding* saja. Sehingga pada iterasi ini *pair programming* dinilai dengan nilai satu (1).

Fase terakhir yaitu *software increament* yaitu *deployment* terhadap versi terakhir aplikasi rapor ini. Tujuannya adalah aplikasi dapat digunakan oleh pengguna secara menyeluruh sesuai dengan hasil analisis kebutuhan.

Pembahasan pertama adalah hasil rancangan aplikasi pada fase *design* dari dua iterasi dapat dilihat di tabel 2. Pada tabel 2 tersebut dapat ditarik kesimpulan, bahwa proses perancangan berjalan dengan baik. Penggunaan praktik dan penggunaan *diagram* dari *extreme programming* mulai dari fase *planning* dan fase *design* dapat dijalankan dengan baik.

Tabel 2. Hasil Rancangan Aplikasi

Iterasi	Usecase Diagram	Class Diagram	Prototype
1.	8	16	2
2.	3	5	1

Sumber : dokumen pribadi

Pembahasan kedua adalah hasil penerapan praktik pada fase *coding* dan *testing* yang dapat dilihat pada Tabel 3. Pada tabel 3 tersebut dapat ditarik kesimpulan bahwa tim pengembang baru bisa menerapkan praktik *standard coding* dengan nilai lima (5) untuk setiap iterasi dengan rata – rata penilaian lima (5). Sedangkan untuk dua praktik yang lain yaitu *test-driven development* dan *pair programming* belum bisa diterapkan. Walaupun praktik *test-driven development* masih diusahakan untuk diterapkan pada iterasi pertama meskipun tidak menyeluruh. Pada iterasi pertama penilaian penerapan praktik *test-driven development* adalah tiga (3) yang menunjukkan usaha dalam menerapkan *test-driven development* sebagian. Sedangkan pada iterasi kedua tidak dapat diterapkan. Untuk praktik *pair programming* masih menjadi ganjalan bagi tim pengembang karena belum bisa mengambil kesepakatan untuk menjalankan praktik tersebut. Sehingga untuk kedua iterasi hasil observasi menilai praktik tersebut mendapatkan nilai satu (1).

Tabel 3. Hasil Penerapan Praktik *Software Craftsmanship*

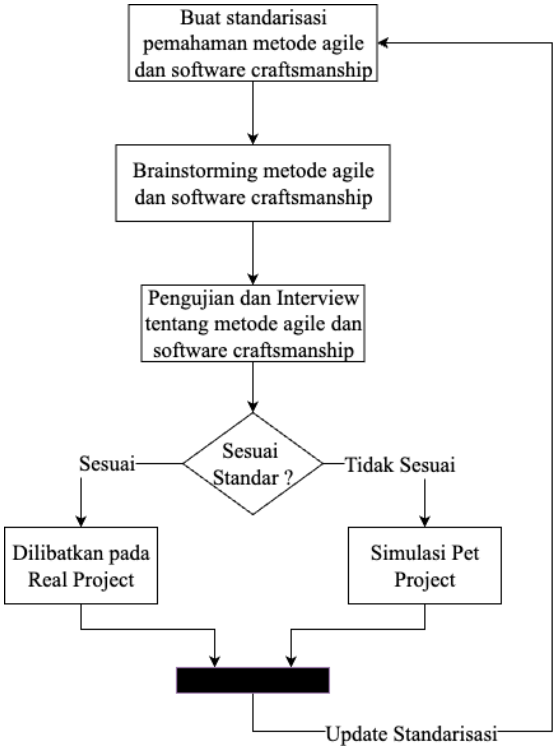
Iterasi	Standard Coding	Test-Driven Development	Pair Programming
1.	5	3	1
2.	5	1	1
Rataan	**5**	**2**	**1**

Sumber : dokumen pribadi

Hasil analisis kesiapan tim pengembang dalam mengadopsi dalam proses observasi pengembangan belum siap secara menyeluruh. Sehingga esensi dari *software craftsmanship* belum bisa tercapai. Sehingga pentingnya perisapan atau pelatihan tentang *software craftsmanship* bagi tim pengembang. Kondisi tersebut sesuai dengan penelitian terdahulu yang melakukan pelatihan dan simulasi dalam memahami *software craftsmanship* [13]. Oleh karena itu faktor kritikal dalam pengembangan aplikasi dengan metode *agile* adalah sumber daya manusia yang terlibat dalam pengembangan aplikasi [7], [8], [10], [11], [20].

Oleh karena itu usulan kerangka evaluasi kesiapan tim dari hasil observasi penelitian ini adalah sebagai berikut. (1) Membuat standarisasi tentang *metode agile* dan *software craftsmanship*, (2) Melakukan *brainstorming* tentang metode *agile* dan *software craftsmanship* secara berkala kepada tim pengembang, (3) Melakukan *interview* dan pengujian kepada setiap tim pengembang, (4) Buat *pet project* untuk tim yang belum memenuhi standar, dan (5) Libatkan tim pada *real project* yang sudah

memenuhi standar. Lakukan evaluasi setiap selesai proyek dan jadikan hasil evaluasi sebagai *update* untuk standarisasi. Gambar 4 merupakan usulan kerangka evaluasi kesiapan tim.



Sumber : dokumen pribadi

Gambar 4. Usulan Kerangka Evaluasi Tim

4. SIMPULAN

Simpulan dari penelitian ini adalah pentingnya persiapan tim pengembang dalam mengadopsi metode *agile* secara umum dan nilai *software craftsmanship* secara khusus. Sehingga nilai utama dari mengadopsi metode *agile* dan *software craftsmanship* tersebut dapat dimanfaatkan.

Saran untuk penelitian kedepannya dapat melakukan membuat standarisasi untuk mengevaluasi tim pengembang tentang metode *agile* dan *software craftsmanship*. Selain itu dapat melakukan eksperimen untuk melihat perbedaan antara tim yang sudah memahami *agile* dan *software craftsmanship* dan yang belum memahami.

Terima kasih kepada LPPM Unindra yang telah membantu pendanaan penelitian ini melalui program **Dana Bantuan Universitas Indraprasta PGRI Jakarta Melalui LPPM sesuai dengan Kontrak Program Penelitian Nomor : 0622/SPP/KP/LPPM/UNINDRA/V/2024, tanggal 20 Mei 2024**.

DAFTAR PUSTAKA

[1] M. Marabesi, F. J. García-Peñalvo, and A. García-Holgado, “Towards a TDD maturity model through an anti-patterns framework,” in *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, Jun. 2023, pp. 1–6. doi: 10.23919/CISTI58278.2023.10211890.

[2] A. Ahmadi and A. Surachman, “Pengembangan Aplikasi Penjualan dengan Metode Extreme Programming dan Penerapan Model Multi-Tenancy,” *STRING (Satuan Tulisan Riset dan Inovasi Teknologi)*, vol. 8, no. 3, p. 303, Apr. 2024, doi: 10.30998/string.v8i3.20276.

[3] A. Ahmadi, E. K. Budiardjo, and K. Mahatma, “Software Craftsmanship Skill using Extreme Programming for Quality Improvement: A Case of Very Small Software Organization,” in *2021 10th International Conference on Software and Computer Applications*, New York, NY, USA: ACM, Feb. 2021, pp. 94–99. doi: 10.1145/3457784.3457835.

[4] A. Shrivastava, I. Jaggi, N. Katoch, D. Gupta, and S. Gupta, “A Systematic Review on Extreme Programming,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, 2021. doi: 10.1088/1742-6596/1969/1/012046.

[5] S. Barahona Rojas, L. Pucha Guzmán, P. Villamarín Coronel, and A. Yunga Benítez, "Scrum with eXtreme Programming: An Agile Alternative in Software Development," *Advances in Intelligent Systems and Computing*, vol. 1277, pp. 350–361, 2021, doi: 10.1007/978-3-030-60467-7_29.

[6] R. Martin, "Clean Agile: Back to Basics," 2019, Accessed: Dec. 16, 2021. [Online]. Available: https://raw.githubusercontent.com/patrickbucher/docs/master/clean-agile/clean-agile.pdf

[7] L. Barros, C. Tam, and J. Varajão, "Agile software development projects–Unveiling the human-related critical success factors," *Inf Softw Technol*, vol. 170, p. 107432, Jun. 2024, doi: 10.1016/j.infsof.2024.107432.

[8] H. Edison, X. Wang, and K. Conboy, "Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review," *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 2709–2731, Aug. 2022, doi: 10.1109/TSE.2021.3069039.

[9] R. Jia, Y. Yang, J. Grundy, J. Keung, and L. Hao, "A systematic review of scheduling approaches on multi-tenancy cloud platforms," *Inf Softw Technol*, vol. 132, p. 106478, Apr. 2021, doi: 10.1016/j.infsof.2020.106478.

[10] C. Uwasomba *et al.*, "Data-Driven Agility: Assessing Agile Culture transformation in a technology organisation," *Inf Softw Technol*, vol. 183, p. 107729, Jul. 2025, doi: 10.1016/j.infsof.2025.107729.

[11] C. Santos, J. Varajão, N. Takagi, and A. Manuela Gonçalves, "Model of driving factors for success in public health project management using structural equation modeling," *Sci Rep*, vol. 14, no. 1, p. 24647, Oct. 2024, doi: 10.1038/s41598-024-75437-7.

[12] A. Sundelin, "Towards Understanding Software Craftsmanship," 2021, Accessed: Dec. 16, 2021. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1586608

[13] A. Barji, T. Jachmann, and F. Paulisch, "Mastering Software Craftsmanship: An Industrial Practitioner Perspective," in *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*, IEEE, Jul. 2024, pp. 1–6. doi: 10.1109/CSEET62301.2024.10663039.

[14] A. Alami, O. Krancher, and M. Paasivaara, "The journey to technical excellence in agile software development," *Inf Softw Technol*, vol. 150, p. 106959, Oct. 2022, doi: 10.1016/j.infsof.2022.106959.

[15] A. Sundelin, J. Gonzalez-huerta, K. Wnuk, and T. Gorschek, "Towards an Anatomy of Software Craftsmanship," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 1, pp. 1–49, Jan. 2022, doi: 10.1145/3468504.

[16] A. Akhtar, B. Bakhtawar, and S. Akhtar, "EXTREME PROGRAMMING VS SCRUM: A COMPARISON OF AGILE MODELS," *International Journal of Technology, Innovation and Management (IJTIM)*, vol. 2, no. 2, Oct. 2022, doi: 10.54489/ijtim.v2i2.77.

[17] L. Gren and P. Ralph, "What makes effective leadership in agile software development teams?," in *Proceedings of the 44th International Conference on Software Engineering*, New York,

NY, USA: ACM, May 2022, pp. 2402–2414. doi: 10.1145/3510003.3510100.

[18] A. Prakash, K. Maddulety, and V. Bhoola, "Agile Project Management: An Empirical Exploration of Adoption Factors and Implementation Strategies Across Industries," *IEEE Engineering Management Review*, pp. 1–7, 2024, doi: 10.1109/EMR.2024.3487291.

[19] B. G. Sudarsono, S. P. Lestari, A. U. Bani, J. Chandra, and J. F. Andry, "Using an extreme programming method for hotel reservation system development," *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 6, pp. 2223–2228, 2020, doi: 10.30534/ijeter/2020/01862020.

[20] S. P. Rath, N. K. Jain, G. Tomer, and A. K. Singh, "A systematic literature review of agile software development projects," *Inf Softw Technol*, vol. 182, p. 107727, Jun. 2025, doi: 10.1016/j.infsof.2025.107727.