Jurnal Masyarakat Informatika *ISSN: 2086-4930*
Vol. 14, No. 1, 2023 *e-ISSN: 2777-0648*

Perbandingan Model Pengembangan Perangkat Lunak Untuk Proyek Tugas Akhir Program Vokasi

Angger Binuko Paksi[1)], Nisa'ul Hafidhoh[2)], Sigit Kariagil Bimonugroho[3)]

Program Studi D3 Teknologi Informasi, Politeknik Negeri Madiun
[1)]angger.binuko@pnm.ac.id, [2)] nisa@pnm.ac.id, [3)]sigit@pnm.ac.id

Abstrak

Salah satu syarat penyelesaian pendidikan pada perguruan tinggi adalah membuat tugas akhir. Hasil akhir dari tugas akhir pada perguruan tinggi vokasi diarahkan dalam bentuk produk atau alat. Untuk membuat produk perangkat lunak yang tersistematis maka dibutuhkan sebuah metodologi atau siklus pengembangan yang lebih dikenal dengan Software Development Life Cycle (SDLC). Secara umum SDLC memiliki beberapa tahapan yang dimulai dari tahap perencanaan, analisis, perancangan, implementasi, pengujian sampai pemeliharaan sistem. Kajian yang dilakukan adalah studi komparatif pada model pengembangan prototype, iterative dan agile. Perbandingan dari ketiga model tersebut menyatakan hasil bahwa model prototype membutuhkan perencanaan kebutuhan lebih matang dibanding kedua model lainnya, sedangkan model iterative dan model agile memiliki fleksibilitas lebih baik dalam menghadapi pengembangan ulang berjangka. Ketiga model cocok untuk perangkat lunak yang bersifat customizable, yang membedakan adalah kapan fase penyesuaian perubahan dilakukan. Dari kajian ketiga model didapatkan bahwa masing-masing mempunyai kelebihan dan kekurangan, sehingga pengembang dalam hal ini mahasiswa dapat menentukan model mana yang sesuai untuk pengembangan perangkat lunak proyek tugas akhir berdasarkan karakteristik setiap model.

Kata kunci : *SDLC, Prototype, Iterative, Agile, Tugas Akhir, Vokasi*

Abstract

One of the requirements for completing education at college is making a final project. The final result of the final project at a vocational college is in the form of a product or tool. To create a systematic software product requires a methodology or development cycle which is better known as the Software Development Life Cycle (SDLC). In general, SDLC has several stages starting from planning, analysis, design, implementation, testing to system maintenance. The study conducted was a comparative study on prototype, iterative and agile development models. Comparison of the three models shows that the prototype model requires more careful planning than the other two models, while the iterative and agile models have better flexibility in dealing with future re-developments. All three models are suitable for software that is customizable, the difference is when the adjustment phase of changes is made. From the study of the three models, it was found that each has advantages and disadvantages, so developers can determine which model is appropriate for the final software development project based on the characteristics of each model.

Keywords : *SDLC, Prototype, Iterative, Agile, Final Project, Vocational*

1 PENDAHULUAN

Tugas Akhir (TA) ataupun skripsi merupakan bagian kegiatan dalam pendidikan tinggi untuk mengaplikasikan semua pengetahuan dan keterampilan yang diperoleh selama proses pembelajaran di pendidikan tinggi. TA dibuat berdasarkan hasil penelitian, kajian terhadap permasalahan yang diperoleh dari pelaksanaan Praktik Kerja Lapangan (PKL), atau permasalahan riil lainnya. TA pada perguruan tinggi yang berbasis vokasi mempunyai bentuk yang berbeda karena hasil akhir diarahkan dalam bentuk produk atau alat [1].

Setiap mahasiswa yang akan menyelesaikan studi pada Perguruan Tinggi diwajibkan untuk menyusun TA, tidak terkecuali mahasiswa program vokasi Program Studi D3 Teknologi Informasi (D3-TI). Salah satu bentuk produk TA dari program studi D3-TI yang umum dikerjakan mahasiswa adalah perangkat lunak (*software*). Dalam proses pembuatan perangkat lunak tersebut dibutuhkan siklus pengembangan yang cocok agar hasil dari perangkat lunak sesuai dengan kebutuhan klien atau pengguna. Siklus pengembangan perangkat lunak dalam dunia industri lebih dikenal dengan istilah *Software Development Life Cycle* (SDLC) [2].

Prinsip SDLC menjadi dasar model pengembangan perangkat lunak seperti *Waterfall*, *Prototype*, *Iterative*, *Spiral*, *Rapid Application Development* (RAD), *Agile* dan lainnya [3]. Dengan adanya beragam jenis model pengembangan perangkat lunak tersebut, terkadang membuat mahasiswa kebingungan untuk memilih model yang cocok untuk proyek TA. Permasalahan tersebut melatarbelakangi untuk dilakukannya studi komparatif model pengembangan perangkat lunak. Penelitian ini membandingkan tiga model yang dinilai memiliki kesamaan terkait fleksibilitas dalam menghadapi perubahan kebutuhan saat proses pengembangan perangkat lunak proyek TA sudah berjalan, yaitu *Prototype Model*, *Iterative Model* dan *Agile Model*.

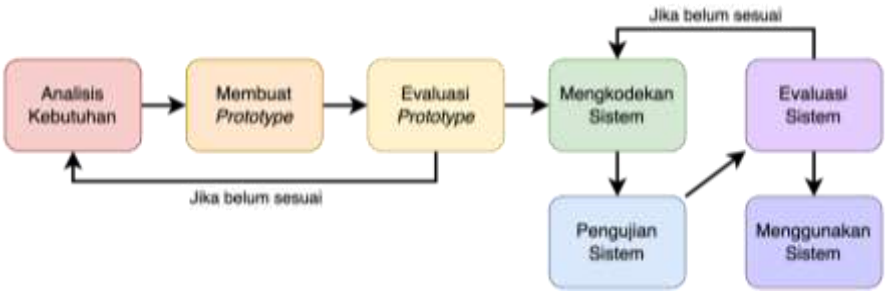
2 TINJAUAN PUSTAKA

2.1 *SOFTWARE DEVELOPMENT LIFE CYCLE*

Software Development Life Cycle (SDLC) merupakan proses pengembangan perangkat lunak yang dirancang untuk berjalan secara sistematis sehingga dapat menghasilkan perangkat lunak yang berkualitas. SDLC juga berfungsi untuk membagi peran dan tanggung secara jelas antara *designer*, *business analyst* dan *project manager*. Fungsi lain dari SDLC juga dapat memberikan gambaran jelas tentang *input* dan *output* dari satu tahap ke tahap berikutnya. Secara umum ada enam tahapan dalam SDLC yaitu perencanaan sistem, analisis sistem, perancangan sistem, implementasi sistem, pengujian sistem, dan pemeliharaan sistem [4].

2.2 *PROTOTYPE MODEL*

Prototype model merupakan pendekatan rekayasa perangkat lunak yang secara langsung menunjukkan bagaimana perangkat lunak atau komponen perangkat lunak akan berfungsi di lingkungannya sebelum tahap konstruksi aktual dilakukan. Model ini memungkinkan pengguna memiliki gambaran awal terkait perangkat lunak yang akan dikembangkan dan pengguna dapat melakukan uji di awal pengembangan sebelum perangkat lunak dirilis [5].



Gambar 1 Model Pengembangan Perangkat Lunak *Prototype*

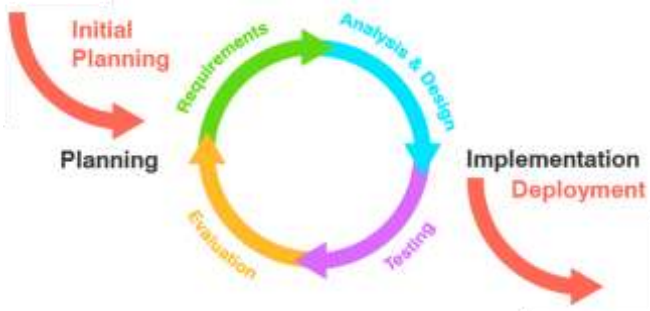
Tahapan model pengembangan perangkat lunak *prototype* diilustrasikan pada Gambar 1 dengan diawali tahap analisis kebutuhan, dimana pengembang mengidentifikasi dan mendefinisikan semua kebutuhan perangkat lunak yang akan dibuat. Tahap selanjutnya membuat *prototype*, yaitu membuat rancangan sementara sistem dengan fokus pada alur perangkat lunak serta metode maupun algoritma yang digunakan. Setelah *prototype* dibuat, maka dilanjut dengan evaluasi *prototype* yang merupakan tahap tinjauan awal sistem untuk mengetahui apakah *prototype* sudah menggambarkan produk akhir dan sesuai dengan harapan klien. Jika hasil evaluasi *prototype* dinilai sudah sesuai maka dapat dilanjutkan mengkodekan sistem dengan konstruksi menggunakan bahasa pemrograman yang sesuai. Tahap selanjutnya adalah pengujian sistem untuk menguji kinerja dari perangkat lunak berdasarkan fungsi. Pengujian ini biasanya dilakukan dengan beberapa metode seperti *whitebox testing*, *blackbox testing*, ataupun metode lainnya. Setelah pengujian sistem dilakukan, maka akan dilakukan evaluasi sistem dimana pengguna melakukan peninjauan apakah perangkat lunak sudah sesuai dengan harapan. Jika sudah sesuai maka dapat dilanjutkan tahap berikutnya. Jika belum, ulangi tahapan mengkodekan sistem dan pengujian sistem. Jika keseluruhan tahap sudah dilakukan maka tahapan terakhir adalah menggunakan sistem dimana perangkat lunak yang telah diuji dan disetujui siap untuk digunakan oleh pengguna.

2.3 *ITERATIVE MODEL*

Iterative model merupakan konsep pengembangan perangkat lunak dengan membangun sebagian kecil dari keseluruhan fitur. Model ini membantu memenuhi jangkauan awal sistem lebih cepat dan merilisnya untuk mendapatkan umpan balik. Dalam *iterative model*, proyek dikembangkan dengan membagi menjadi komponen-komponen yang lebih kecil, yang kemudian ditinjau untuk mengidentifikasi kebutuhan lebih lanjut sehingga hasilnya dapat dimanfaatkan pada iterasi berikutnya. Proses ini kemudian diulangi, menghasilkan versi baru dari perangkat lunak di setiap akhir iterasi. Sistem dianggap siap apabila keseluruhan fitur dari perangkat lunak sudah lengkap [6].

Tahapan model pengembangan perangkat lunak *iterative* diilustrasikan pada Gambar 2 dengan tahap awal berupa perencanaan (*planning*). Proyek dimulai dengan perencanaan menyeluruh sesuai dengan kebutuhan yang ditetapkan. Untuk menentukan kebutuhan, umpan balik dari pengguna perlu dikumpulkan dan dianalisis. Kebutuhan ini bisa menjadi kebutuhan baru atau perpanjangan dari kebutuhan yang sudah dibangun. Ketika tahap perencanaan selesai, selanjutnya tahap analisis dan perancangan (*analysis & design*) dengan menentukan logika

bisnis dari proyek. Logika bisnis memungkinkan komunikasi antara sistem dan pengguna akhir.



Gambar 2 Model Pengembangan Perangkat Lunak *Iterative*

Desain yang tepat dapat memberikan hasil yang paling optimal dan akan mengurangi tekanan dana dari klien. Desain ini dapat berupa yang baru atau perluasan dari kebutuhan yang sudah dibangun. Desain yang telah diputuskan akan diimplementasikan pada tahapan penerapan (*implementation*) oleh pengembang dengan standar koding dan metrik yang telah ditentukan. Pengembang perlu menerapkan pengujian unit pada setiap tahap pengembangan saat mengembangkan kode. Sebelum memulai setiap iterasi, pengembang menegosiasikan prioritas dan detail tugas dengan klien. Semua perubahan kode biasanya diunggah ke *staging environment* terlebih dahulu (*deployment process*).

Ketika tim pengembang selesai dengan pengkodean, perlu dilakukan pengujian (*testing*) untuk menemukan dan memperbaiki semua *bug* dan kesalahan. Pada tahap pengujian tidak cukup hanya dengan menguji produk saja, teknisi *Quality Assurance* (QA) juga harus memperhatikan dokumentasi pengguna dan mengujinya juga. Penguji dapat menulis kasus uji baru atau menggunakan yang sudah ada. Setelah dilakukan pengujian selanjutnya akan dilakukan evaluasi (*evaluation*), pada tahap ini klien bersama dengan pengembang mengevaluasi prototipe dan memeriksa apakah semua kebutuhan sudah terpenuhi. Berdasarkan rencana kebutuhan akan disusun dan diimplementasikan lebih lanjut sebagai bagian dari siklus iterasi berikutnya.

2.4 *AGILE MODEL*

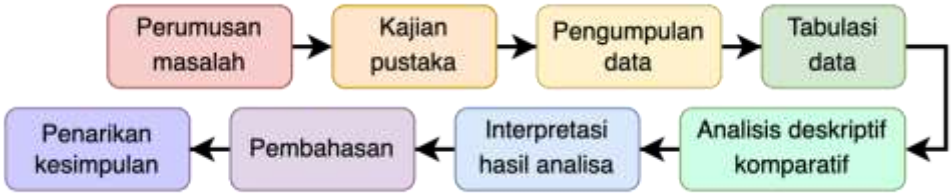
Agile model merupakan konsep pengembangan perangkat lunak berbasis pengulangan proses dimana aturan dan solusi yang disepakati akan diterapkan dalam kolaborasi yang terorganisir dan terstruktur antar kelompok dan merupakan model pengembangan perangkat lunak jangka pendek. *Agile model* memecah tugas menjadi iterasi yang lebih kecil, atau bagian-bagian yang tidak secara langsung melibatkan perencanaan jangka panjang yang biasa dikenal dengan istilah *sprint*. Rencana mengenai kebutuhan proyek, jumlah *sprint*, durasi dan ruang lingkup setiap *sprint* ditentukan dengan jelas pada awal proses pengembangan [7].

Gambar 3 Model Pengembangan Perangkat Lunak *Agile*

Tahapan model pengembangan perangkat lunak *agile* diilustrasikan pada Gambar 3 dimulai dengan tahap perencanaan (*planning*) dimana pengembang dan klien mendefinisikan kebutuhan apa saja yang diperlukan dalam menentukan perangkat lunak yang akan dibangun. Setelah mengidentifikasi kebutuhan proyek, dilanjutkan dengan tahap perancangan (*design*) alur kerja dari sebuah sistem, tahap ini dapat menggunakan diagram alir atau diagram lainnya untuk menunjukkan cara kerja dari fitur serta menunjukkan bagaimana penerapan UI/UX pada sistem. Ketika rancangan sudah dibuat dapat dilanjutkan dengan proses pengembangan (*development*) yaitu proses dimana *programmer* melakukan pengkodean perangkat lunak. Pada tahap pengembangan ini produk akan menjalani beberapa tahap penyempurnaan, sehingga mencakup fungsionalitas yang dibutuhkan dan minimalis.Perangkat lunak yang sudah dibuat, akan melewati pengujian (*testing*) oleh bagian *quality assurance* agar *bug* yang ditemukan dapat segera diperbaiki dan kualitas perangkat lunak terjaga. Setelah sistem dianggap memenuhi syarat dari *quality assurance* maka perangkat lunak siap disebarkan (*deployment*). Aplikasi web akan di-*deploy* ke server, dan aplikasi mobile akan di-*deploy* ke *playstore* atau *appstore*. Setelah merilis produk, langkah terakhir adalah meninjau (*review*) produk. Dalam hal ini, tim menerima umpan balik tentang produk dan bekerja melalui umpan balik tersebut. *Agile model* dapat dibagi menjadi beberapa jenis metode pengembangan perangkat lunak, diantaranya *Adaptive Software Development* (ASD), *Agile Unified Process* (AUP), *Crystal, Dynamic Systems Development Method* (DSDM), *Extreme Programming* (XP), *Feature Driven Development* (FDD), Kanban, *Lean Software Development* (LSD), *Scaled Agile Framework* (SAFe), Scrum, dan juga Scrumban.

3 METODE PENELITIAN

Penelitian ini memanfaatkan data tinjauan beberapa jurnal dan literasi untuk dilakukan komparasi menggunakan metode penelitian deskriptif komparatif. Penelitian deskriptif memiliki tujuan untuk memberikan deskripsi, penjelasan, juga validasi mengenai fenomena yang sedang diteliti. Adapun jenis-jenis penelitian deskriptif diantaranya penelitian tindakan, penelitian kepustakaan dan penelitian komparatif. Penelitian komparatif berfungsi membandingkan dua perlakuan atau lebih dari beberapa variabel sekaligus [8]. Tahapan yang dilakukan dalam penelitian ini dijabarkan dalam Gambar 4.



Gambar 4 Tahapan-tahapan Penelitian

4 HASIL DAN PEMBAHASAN

4.1 PERBANDINGAN KELEBIHAN DAN KEKURANGAN

Bagian ini menjelaskan kelebihan dan kekurangan dari tiga model pengembangan perangkat lunak berdasarkan studi literatur [2], [3], [5], [6], [9]–[11]. Hasil dari perbandingan disajikan dalam Tabel 1.

4.2 PERBANDINGAN BERDASARKAN TAHAPAN SDLC

Bagian ini merupakan penjabaran dari perbandingan tiga model yang dipilih untuk dievaluasi berdasarkan tahapan pada siklus hidup pengembangan perangkat lunak secara umum. Evaluasi dilakukan menggunakan tahapan-tahapan SDLC sebagai dasar kriteria dalam perbandingan. Kriteria tersebut mencakup enam tahapan umum dari metode SDLC yaitu, perencanaan, analisis, perancangan, implementasi, pengujian dan pemeliharaan. Hasil dari perbandingan disajikan dalam Tabel 2.

4.3 PERBANDINGAN BERDASARKAN KRITERIA TUGAS AKHIR

Melalui studi literatur yang telah di dilakukan, penelitian ini membandingkan tiga model pengembangan perangkat lunak ke dalam poin-poin yang menjadi karakteristik tugas akhir program vokasi [1], [12]. Hasil dari perbandingan disajikan dalam Tabel 3.

Tabel 1 Perbandingan Kelebihan dan Kekurangan Model *Prototype*, *Iterative* dan *Agile*

Model	Kelebihan	Kekurangan
Prototype	Dapat memberikan gambaran sistem yang jelas kepada klien	Sistem akan terhambat jika komunikasi antara klien dan pengembang tidak berjalan efektif
	Klien berpartisipasi aktif dalam pengembangan sistem, sehingga hasil produk pengembangan mudah disesuaikan dengan kebutuhan pelanggan	Klien dapat terus menambah persyaratan sistem saat proses *prototyping*, sehingga menambah kerumitan pembuatan sistem
	Mempercepat kerja karena telah memiliki prototipe dan menyamakan gambaran perangkat lunak dalam kelompok	Jika pengembangan prototipe sistem berbeda dari keinginan klien maka perlu dikembangkan ulang sehingga membutuhkan waktu dan biaya lebih
	Implementasi sistem lebih mudah karena klien sudah mengetahui gambaran sistem sebelumnya	
	Hemat waktu dan biaya dalam pengembangan perangkat lunak jika klien puas di tahap awal	

Model	Kelebihan	Kekurangan
Iterative	Memungkinkan identifikasi dan koreksi kekurangan pada tahap awal pengembangan agar tidak terlanjur masuk ke proses pengembangan yang lebih jauh	Tidak semua kebutuhan ditentukan sejak awal proyek, sehingga dibutuhkan perencanaan yang baik untuk meminimalisir iterasi yang menyebabkan proses tersendat
	Sebagian komponen sistem dapat dikembangkan di awal siklus dengan cepat	Biaya untuk perubahan lebih rendah, namun kurang cocok untuk mengubah kebutuhan
	Perangkat lunak diproduksi lebih awal yang memfasilitasi evaluasi dan umpan balik pelanggan	Perlu pendefinisian modul yang baik agar menekan kerugian waktu, materi, dan biaya
	Perangkat lunak memiliki beberapa versi terkait tahap peluncuran aplikasi, *versioning* ini memudahkan tiap iterasi terus diperbarui	Jika pada tahap akhir terdapat permasalahan maka perbaikan untuk masalah tersebut akan mahal
	Jika iterasi baru gagal, iterasi sebelumnya bisa *rolled-back* dengan kerugian minimal	Akhir proyek mungkin tidak diketahui yang merupakan risiko
	Mudah beradaptasi antara kebutuhan dan sistem yang dibangun jika ada perubahan	Sumber daya yang sangat terampil diperlukan untuk analisis risiko
	Cocok untuk organisasi yang cepat berubah, terutama yang memiliki tim kecil	Kemajuan proyek sangat tergantung pada tahap analisis risiko
	Memudahkan proses *error debugging*	
	Proses penyelesaian sistem relatif cepat	
Agile	Waktu yang singkat dalam proses pengembangan perangkat lunak dan tidak memerlukan sumber daya yang banyak	Memiliki sedikit perencanaan sehingga bentuk akhir dari perangkat lunak sulit ditentukan dan kurang tergambar jelas
	Memprioritaskan kebutuhan dan kepuasan klien, sehingga bersifat fleksibel karena mudah diubah	Membutuhkan komitmen tim yang tinggi agar hasil sesuai dengan kebutuhan klien
	Memberikan visibilitas bagi proyek yang sedang berlangsung, sehingga klien dapat memberikan umpan balik secara langsung	Dokumentasi dibuat dalam waktu yang singkat sehingga berujung pada hasil yang kurang lengkap
	Perubahan dapat ditangani dengan cepat sesuai dengan kebutuhan klien	Menguras banyak energi serta waktu karena perubahan bisa terjadi kapan saja
	Pengembangan memiliki fokus yang bertahap dan terprediksi, sehingga resiko dapat diminimalisir	Kurang tepat jika diimplementasikan pada tim dengan skala besar (> 20 orang)
	Mudah melacak proses pengembangan secara berkala	Adanya ketidakpastian waktu berakhirnya proyek, terlebih ketika ada banyak perubahan
	Memudahkan proses *error debugging*	
	Proses penyebaran perangkat lunak relatif cepat	

Tabel 2 Perbandingan Model *Prototype*, *Iterative* dan *Agile* Berdasarkan Tahapan SDLC

Tahapan SDLC	*Prototype*	*Iterative*	*Agile*
Perencanaan Sistem	Berawal dari kebutuhan, dilakukan secara menyeluruh	Berawal dari kebutuhan bagian-bagian kecil, dilakukan secara menyeluruh per bagian	Berawal dari kebutuhan bagian-bagian kecil, dilakukan secara bertahap dan sering
Analisis Sistem	Kebutuhan data dapat ditambah ataupun dikurangi sesuai dengan kebutuhan user saat dilakukan evaluasi *prototype*	Kebutuhan data dapat ditambah ataupun dikurangi sesuai dengan kebutuhan user, ketika dilakukan evaluasi pada iterasi komponen sistem	Kebutuhan data dapat ditambah ataupun dikurangi sesuai dengan kebutuhan user melalui umpan balik langsung
	Selama sistem atau perangkat lunak masih dalam bentuk prototype, perubahan dapat dilakukan	Selama sistem atau perangkat lunak masih dalam iterasi komponen sistem, perubahan dapat dilakukan	Perubahan dapat dilakukan sesering mungkin bahkan setelah perencanaan awal selesai

Tahapan SDLC	*Prototype*	*Iterative*	*Agile*
Perancangan Sistem	*Prototype* berfungsi memberi gambaran sistem yang akan dibangun, jadi user dapat melihat dan berinteraksi dengan gambaran akhir sistem	Rancangan dibuat per komponen sebagai gambaran sistem akhir yang akan dibangun	Rancangan dibuat sesederhana mungkin, sehingga user masih melihat secara abstrak gambaran sistem akhir
	Pengujian dapat dilakukan ketika *prototype* telah dibuat, sehingga hasil pengujian dapat merubah rancangan sistem	Pengujian dapat dilakukan ketika komponen sistem telah dibuat, sehingga hasil uji dapat merubah rancangan komponen sistem pada iterasi	Testing dapat dilakukan sesering mungkin setelah adanya perubahan, sehingga penyesuaian rancangan sering dilakukan
	Klien berperan aktif dalam pengembangan sistem	Klien berperan aktif dalam pengembangan sistem	Klien berperan aktif dalam pengembangan sistem
	Sistem yang dibangun akan sesuai dengan kebutuhan user	Sistem yang dibangun akan sesuai dengan kebutuhan user	Sistem yang dibangun akan sesuai dengan kebutuhan user
Implementasi Sistem	Pengkodean dilakukan setelah evaluasi *prototype* selesai	Pengkodean dilakukan setelah rancangan komponen selesai	Pengkodean dilakukan setelah perancangan fitur selesai
	Rilis dilakukan setelah evaluasi sistem selesai	Rilis dilakukan setelah evaluasi komponen selesai	Rilis dilakukan setelah pengkodean dan pengujian selesai
	Mengedepankan aspek kenyamanan user	Mengedepankan aspek kenyamanan user dan kecepatan pembangunan	Mengedepankan aspek kenyamanan user dan kecepatan pembangunan
Pengujian Sistem	Evaluasi dilakukan ketika *prototype* telah dibangun	Evaluasi dilakukan ketika komponen telah dibangun	Evaluasi dilakukan setiap saat bersamaan dengan proses pengkodean
Pemeliharaan Sistem	Dilakukan sesuai kesepakatan	Dilakukan sesuai kesepakatan dan dapat bersifat kontinuitas	Dilakukan sesuai kesepakatan dan dapat bersifat kontinuitas

Tabel 3 Perbandingan Model *Prototype*, *Iterative* dan *Agile* Berdasarkan Kriteria Tugas Akhir

Karakteristik TA Program Vokasi	*Prototype*	*Iterative*	*Agile*
Pemecahan terhadap masalah	Sedang	Tinggi	Tinggi
Perencanaan produk	Sedang	Sedang	Rendah
Observasi objek	Tinggi	Sedang	Sedang
Pengumpulan data	Tinggi	Sedang	Rendah
Integrasi antar disiplin ilmu	Tinggi	Tinggi	Tinggi
Aplikatif/penerapan disiplin ilmu	Tinggi	Tinggi	Tinggi
Komitmen	Sedang	Sedang	Tinggi
Kolaborasi antar aktor	Sedang	Tinggi	Tinggi
Kompleksitas pengembangan	Sedang	Rendah	Rendah
Fleksibilitas pengembangan	Sedang	Tinggi	Tinggi
Biaya pengembangan	Sedang	Sedang	Sedang
Kecepatan pengembangan	Sedang	Tinggi	Tinggi
Kelengkapan dokumentasi	Sedang	Sedang	Rendah

Karakteristik TA Program Vokasi	*Prototype*	*Iterative*	*Agile*
Kesesuaian produk akhir	Tinggi	Tinggi	Tinggi
Kontinuitas	Sedang	Tinggi	Tinggi

5 KESIMPULAN

Berdasarkan kajian dan perbandingan yang telah dilakukan, maka dapat ditarik kesimpulan bahwa ketiga model pengembangan perangkat lunak yaitu *prototype*, *iterative* dan *agile* mempunyai karakteristiknya masing-masing. Model pengembangan *prototype* cocok digunakan dalam sistem atau perangkat lunak yang bersifat *customizable*. Perangkat lunak yang dibuat berdasarkan permintaan dan kebutuhan tertentu dapat disesuaikan jika ada perubahan dalam proses pembuatannya, penerapan model sesuai untuk tugas akhir yang mempunyai tujuan dalam mengimplementasikan suatu metode maupun algoritma pada suatu kasus karena pengguna mempunyai gambaran awal terkait perangkat lunak yang akan dibuat sebelum lanjut ke fase pengembangan berikutnya. Model pengembangan *iterative* cocok digunakan dalam sistem atau perangkat lunak yang bersifat *customizable* dan berkelanjutan. Perangkat lunak yang dibuat berdasarkan permintaan dan kebutuhan tertentu dapat disesuaikan jika ada perubahan dalam proses pembuatanya, penerapan model sesuai untuk tugas akhir yang mempunyai tujuan dalam menyelesaikan permasalahan atau kasus pada objek industri tertentu serta memiliki kemungkinan untuk kebutuhan pengembangan kembali dalam jangka waktu yang cukup panjang karena mendukung pengelolaan perangkat lunak berbasis *versioning*. Model pengembangan *agile* cocok digunakan dalam sistem atau perangkat lunak yang bersifat *customizable,* berkelanjutan, dan butuh waktu yang singkat. Perangkat lunak yang dibuat berdasarkan permintaan dan kebutuhan tertentu dapat disesuaikan jika ada perubahan dalam proses pembuatanya, penerapan model sesuai untuk tugas akhir yang mempunyai tujuan dalam menyelesaikan permasalahan publik maupun objek industri yang terbatas dalam pengumpulan data serta memiliki kemungkinan untuk kebutuhan pengembangan kembali dalam waktu singkat. Dari ketiga model yang telah dikaji tidak dapat ditentukan mana yang lebih baik karena masing-masing model mempunyai kelebihan dan kekurangan. Pengembang dalam hal ini adalah mahasiswa dapat menentukan model mana yang paling sesuai untuk pengembangan perangkat lunak proyek tugas akhir berdasarkan karakteristik masing-masing model.

DAFTAR PUSTAKA

[1] P4MP PNM, Panduan Penulisan Praktek Kerja Lapangan, Tugas Akhir Dan Skripsi. Madiun: Politeknik Negeri Madiun, 2020.
[2] H. Subakti dkk., Rekayasa Perangkat Lunak. Bandung: Media Sains Indonesia, 2022.
[3] T. Pricillia dan Zulfachmi, "Perbandingan Metode Pengembangan Perangkat Lunak (Waterfall, Prototype, RAD)," Jurnal Bangkit Indonesia, vol. 10, no. 1, hlm. 6–12, Mar 2021, doi: 10.52771/bangkitindonesia.v10i1.153.
[4] Deni Murdiani dan Muhamad Sobirin, "Perbandingan Metodologi Waterfall dan RAD (Rapid Application Development) dalam Pengembangan Sistem Informasi," Jurnal Informatika Teknologi dan Sains, vol. 4, no. 4, hlm. 302–306, Nov 2022, doi: 10.51401/jinteks.v4i4.2008.

[5] H. S. Dewi dkk., "Smart Farming Teknologi Monitoring Produksi Dan Pemasaran Kebun Organik," Jusikom : Jurnal Sistem Komputer Musirawas, vol. 7, no. 1, hlm. 20–31, Jun 2022, doi: 10.32767/jusikom.v7i1.1587.
[6] D. S. Umar, "Perancangan Aplikasi Sistem Informasi Laundry Berbasis Microsoft Accsess," Universitas Komputer Indonesia, Bandung, 2021.
[7] N. Hikmah, A. Suradika, dan R. A. Ahmad Gunadi, "Metode Agile untuk Meningkatkan Kreativitas Guru Melalui Berbagi Pengetahuan (Knowledge Sharing) (Studi Kasus: SDN Cipulir 03 Kebayoran Lama, Jakarta," Instruksional, vol. 3, no. 1, hlm. 30, Okt 2021, doi: 10.24853/instruksional.3.1.30-39.
[8] M. Ramdhan, Metode Penelitian. Surabaya: Cipta Media Nusantara, 2021.
[9] H. A. Prasetyo, B. Priyambadha, dan A. Arwan, "Pembangunan Aplikasi Sistem Informasi Pergudangan pada Rumah Sakit Umum Daerah Dr. Murjani Sampit Kabupaten Kotawaringin Timur," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 2, no. 7, hlm. 2789–2800, Jul 2018.
[10] M. Sulemani, *What is a software process model? Top 7 models explained*, Educative, 24 Januari 2021. https://www.educative.io/blog/software-process-model-types (diakses 2 Mei 2023).
[11] S. Soobia.et.al., "Analysis of Software Development Methodologies," International Journal of Computing and Digital Systems, vol. 8, no. 5, hlm. 445–460, Jan 2019, doi: 10.12785/ijcds/080502.
[12] Vokasi UB, Pedoman Tugas Akhir D-III Vokasi. Malang: Universitas Brawijaya, 2022.