BAB II

TINJAUAN PUSTAKA

2.1. Studi Sebelumnya

Prayoga dkk. melakukan penelitian mengenai pengembangan sistem *MyITS Dorm* menggunakan metode *Domain Driven Design* dan *Onion Architecture*, yang dipilih untuk memisahkan kompleksitas domain dari infrastruktur teknis dalam aplikasi. Penelitian ini menjelaskan proses rancang bangun sistem *MyITS Dorm* dengan memfokuskan pada kebutuhan bisnis dan domain utama terlebih dahulu, sebelum melibatkan lapisan infrastruktur seperti *database* dan antarmuka pengguna. Penggunaan *Onion Architecture* membantu memastikan bahwa ketergantungan terhadap teknologi eksternal dapat diminimalisir, sehingga perubahan pada domain bisnis tidak mempengaruhi lapisan lain. Hasil penelitian ini menunjukkan bahwa penerapan *Domain Driven Design* dan *Onion Architecture* dapat meningkatkan skalabilitas dan fleksibilitas sistem, yang terbukti dari kemudahan dalam menambahkan fitur baru tanpa merusak struktur yang sudah ada [6].

Khoirunnisa melakukan penelitian mengenai rancang bangun sistem *e-learning* yang mengadopsi pendekatan *microservices* dan *Domain-Driven Design* (DDD). Penelitian ini bertujuan untuk menciptakan sistem yang lebih fleksibel dan mudah diintegrasikan dengan layanan lain, serta mampu memenuhi kebutuhan pengguna yang beragam. Dalam penelitian ini, proses pengembangan sistem dilakukan melalui beberapa tahapan, termasuk analisis kebutuhan, perancangan, pengembangan, dan pengujian, yang semuanya berfokus pada kolaborasi antara tim pengembang dan pengguna. Hasil dari penelitian menunjukkan bahwa penerapan *microservices* dan DDD berhasil meningkatkan kinerja sistem *e-learning*, memberikan kemudahan dalam pemeliharaan dan pengembangan lebih lanjut, serta meningkatkan pengalaman pengguna secara keseluruhan [7].

Simanjunta dan timnya melakukan penelitian mengenai tren dan tantangan implementasi *Domain Driven Design* (DDD) dalam era digital, yang penting untuk meningkatkan kualitas pengembangan perangkat lunak. Penelitian ini mengidentifikasi

berbagai tantangan yang dihadapi dalam penerapan DDD, termasuk kompleksitas pengelolaan domain, kolaborasi antar tim, dan integrasi dengan teknologi baru. Dengan pendekatan analisis kualitatif, penelitian ini meneliti berbagai kasus implementasi DDD di industri dan menyimpulkan bahwa, meskipun ada tantangan, penerapan DDD dapat membantu organisasi dalam menciptakan arsitektur perangkat lunak yang lebih fleksibel dan responsif terhadap perubahan kebutuhan bisnis. Hasil dari penelitian menunjukkan bahwa adopsi DDD yang efektif dapat memberikan keunggulan kompetitif melalui peningkatan inovasi dan efisiensi dalam proses pengembangan [8].

Ispandi melakukan penelitian tentang pengembangan Sistem Informasi Perpustakaan Digital berbasis web dengan menggunakan metode *Software Development Life Cycle* (SDLC) model *Waterfall*. Penelitian ini bertujuan untuk mengatasi keterbatasan sistem perpustakaan konvensional dengan membangun sistem berbasis digital yang dapat diakses kapan saja dan di mana saja. Proses pengembangan perangkat lunak meliputi lima tahap utama: analisis kebutuhan, desain sistem, implementasi, integrasi & pengujian, serta pemeliharaan. Hasil penelitian menunjukkan bahwa sistem informasi perpustakaan digital ini mampu meningkatkan efisiensi pengelolaan data peminjaman dan pengembalian buku serta mempermudah pengguna dalam mengakses koleksi buku. Pengujian sistem dengan metode *black-box* dan *white-box* testing memberikan hasil positif dengan persentase kelayakan sebesar 84,8%, menyatakan bahwa sistem layak digunakan oleh masyarakat luas [9].

Wati Erawati dkk. melakukan penelitian mengenai pengembangan Sistem Informasi Akademik berbasis web menggunakan metode *Software Development Life Cycle* (SDLC) dengan model *Waterfall*. Penelitian ini bertujuan untuk mempermudah pengelolaan data akademik seperti kehadiran siswa, rencana kegiatan harian, dan rapor siswa. Tahapan SDLC yang diterapkan meliputi analisis kebutuhan, desain sistem menggunakan UML, pengkodean menggunakan PHP, JavaScript, dan HTML, hingga pengujian sistem. Pengujian dilakukan dengan uji regresi, yang menunjukkan bahwa sistem memenuhi kriteria keberhasilan dalam hal akurasi dan kecepatan pengolahan data. Hasil penelitian ini membuktikan bahwa metode SDLC dapat membantu

mengembangkan sistem yang sesuai dengan kebutuhan pengguna secara efektif dan efisien [10].

Hayat dkk. melakukan penelitian tentang perancangan ulang sistem informasi Balai Penjaminan Mutu Pendidikan (BPMP) Sulawesi Tenggara dengan menggunakan metode Agile. Penelitian ini bertujuan untuk meningkatkan kualitas antarmuka, kemudahan navigasi, dan efisiensi pengelolaan data pada sistem informasi yang digunakan. Sistem baru dirancang menggunakan teknologi seperti PHP, HTML, CSS, JavaScript, dan MySQL, serta dilengkapi dengan desain yang responsif. Hasil penelitian menunjukkan bahwa perancangan ulang sistem informasi ini berhasil meningkatkan kualitas layanan pendidikan, mempermudah pengguna dalam mengakses informasi, dan memberikan pengalaman yang lebih baik bagi pengguna [11].

Dimyati dkk. melakukan penelitian tentang perancangan ulang sistem informasi penjualan dan pemesanan produk *furniture* di UD. Utama Karya dengan metode *Rapid Application Development* (RAD) yang disesuaikan dengan tahapan *Software Development Life Cycle* (SDLC). Penelitian ini menghasilkan sistem *e-commerce* terintegrasi yang mempermudah pelanggan dalam mengakses katalog produk, melakukan pemesanan, dan menyelesaikan transaksi. Selain itu, sistem ini meningkatkan efisiensi operasional perusahaan melalui pengelolaan data pelanggan yang lebih sistematis, laporan yang akurat, dan mendukung pengambilan keputusan bisnis yang lebih baik [12].

Tabel 2. 1. Perbandingan Studi Sebelumnya

No	Penulis	Tahun	Metode	Hasil
1	Prayoga, dkk. [6]	2023	*Domain Driven Design*, *Onion Architecture*	Penerapan *Domain Driven Design* dan *Onion Architecture* meningkatkan skalabilitas dan fleksibilitas sistem MyITS Dorm, mempermudah penambahan fitur baru tanpa merusak struktur yang sudah ada.
2	Khoirunnisa [7]	2021	*Microservices, Domain Driven Design*	Penggunaan *microservices* dan DDD meningkatkan kinerja sistem *e-learning,* memudahkan pemeliharaan dan pengembangan lebih lanjut, serta meningkatkan pengalaman pengguna secara keseluruhan.
3	Simanjunta, dkk. [8]	2022	*Domain Driven Design*	Penerapan DDD membantu organisasi menciptakan arsitektur perangkat lunak yang fleksibel dan responsif terhadap perubahan kebutuhan bisnis meskipun terdapat tantangan seperti kompleksitas pengelolaan domain.
4	Ispandi [9]	2019	*Software Development Life Cycle* (SDLC)	Penerapan SDLC dengan model *waterfall* menghasilkan Sistem Informasi Perpustakaan Digital berbasis web yang meningkatkan efisiensi proses pencarian, peminjaman, dan pengembalian buku. Tiap tahap SDLC, mulai dari analisis kebutuhan hingga pengujian, berhasil memastikan sistem memenuhi kebutuhan pengguna dengan tingkat kepuasan mencapai 84,8%.
5	Wati Erawati, dkk. [10]	2023	*Software Development Life Cycle* (SDLC)	Pengembangan Sistem Informasi Akademik berbasis web menggunakan SDLC dengan model *Waterfall* berhasil meningkatkan akurasi dan efisiensi pengolahan data akademik.

6	Hayat, dkk. [11]	2025	*Agile*	Sistem informasi BPMP yang dirancang ulang meningkatkan kualitas antarmuka, navigasi, serta efisiensi pengelolaan data menggunakan PHP, HTML, CSS, JavaScript, dan MySQL dengan desain responsif.
7	Dimyati dkk. [12]	2024	RAD	Perancangan ulang sistem informasi penjualan dan pemesanan produk furniture menghasilkan sistem e-commerce yang terintegrasi. Setiap tahapan SDLC, mulai dari analisis kebutuhan, desain, implementasi, hingga pengujian, disesuaikan dengan pendekatan RAD untuk percepatan pengembangan. Hasilnya, sistem baru meningkatkan efisiensi transaksi, mempermudah pelanggan dalam mengakses katalog dan melakukan pemesanan, serta mendukung pengelolaan data pelanggan dan laporan yang lebih akurat.

2.2. Dasar Teori

2.2.1. *Revamp* Sistem Informasi

Revamp sistem informasi adalah proses mendesain ulang atau memperbarui suatu sistem untuk meningkatkan fungsionalitas, efisiensi, dan pengalaman pengguna. Dalam konteks pengembangan perangkat lunak, *revamp* bertujuan untuk memperbaiki kelemahan yang ada pada sistem lama dan memperkenalkan fitur-fitur baru sesuai dengan kebutuhan dan tuntutan bisnis yang berkembang. Menurut Sumarno (2020), proses *revamp* pada sistem informasi melibatkan berbagai aspek, mulai dari pembaruan antarmuka pengguna hingga perubahan dalam struktur data ataupun logika pemrograman, untuk memastikan sistem dapat memenuhi tuntutan baru yang berkembang dalam organisasi [13].

Proses *revamp* sistem informasi dilakukan melalui tahapan yang telah terbukti efektif dalam pengembangan perangkat lunak, sebagaimana dijelaskan oleh *Setiawan* [14]. Tahap pertama adalah analisis kebutuhan, yang berfungsi untuk mengidentifikasi masalah yang ada pada sistem lama dan memetakan kebutuhan baru yang lebih relevan. Setelah kebutuhan pengguna dipahami, tahap berikutnya adalah desain sistem baru, yang mencakup pembuatan prototipe untuk memvalidasi konsep dan perubahan desain dengan melibatkan pengguna secara langsung. Selanjutnya, sistem baru dikembangkan, diperbaharui, dan diintegrasikan dengan sistem yang ada untuk memastikan transisi yang mulus, tanpa mengganggu kelancaran operasional. Pengujian dilakukan secara menyeluruh untuk memastikan sistem baru berfungsi dengan baik, diikuti dengan peluncuran bertahap dan pemeliharaan untuk menjaga kestabilan serta memastikan sistem terus beradaptasi dengan perubahan yang terjadi di masa depan.

2.2.2. Domain Driven Design

Domain Driven Design (DDD) adalah pendekatan pengembangan perangkat lunak yang menekankan pentingnya pemahaman dan pemodelan *domain* dalam merancang sistem yang kompleks. DDD dipilih sebagai

metodologi untuk membantu dalam merancang sistem yang kompleks, karena pendekatan ini menekankan pentingnya pemahaman dan pemodelan domain secara mendalam. Menurut Millett dan Tune (2015), DDD mendorong kolaborasi antara pengembang perangkat lunak dengan ahli *domain* untuk memastikan bahwa model yang dibuat mencerminkan kebutuhan bisnis secara akurat [15].

Selain itu, DDD memberikan fleksibilitas dalam menghadapi perubahan bisnis menurut Özkan dkk. [16], karena setiap perubahan hanya perlu dilakukan dalam *bounded context* yang relevan, seperti yang dijelaskan juga oleh Medeiros dan Anacleto dalam penelitian mereka tentang tantangan DDD dalam industri perangkat lunak [17]. Hal ini membuat DDD lebih unggul dibandingkan dengan metodologi lain yang mungkin tidak memberikan batasan kontekstual yang jelas, sehingga bisa mengakibatkan perubahan yang tidak terkelola dengan baik di seluruh sistem.

Komponen utama dari DDD adalah *domain expert*, yaitu pihak yang paling memahami detail *domain* bisnis atau sistem yang sudah ada. Mereka berperan penting dalam memastikan model yang dibuat sesuai dengan realitas bisnis, memvalidasi, dan mengarahkan pengembangan sistem sesuai dengan kebutuhan dan proses bisnis. Menurut Evans, *domain expert* berfungsi sebagai "penjaga" pemahaman bisnis yang tepat dalam setiap langkah pengembangan sistem [18].

Salah satu elemen krusial dalam DDD adalah *ubiquitous language* atau bahasa bersama antara pengembang dan pemangku kepentingan domain. Bahasa ini digunakan secara konsisten di seluruh proses pengembangan, mulai dari kode hingga diskusi sehari-hari. Dengan bahasa yang seragam, semua pihak terlibat dalam pengembangan sistem dapat memiliki pemahaman yang sama, sehingga meminimalkan kesalahpahaman dan meningkatkan efisiensi kerja [19].

Selain itu, DDD memperkenalkan konsep *bounded context*, yaitu batasan yang jelas antara satu konteks domain dengan konteks lainnya. Konsep bounded

context dalam DDD memungkinkan pembagian domain yang kompleks menjadi subdomain yang lebih kecil dan terisolasi, yang memudahkan pengelolaan sistem. Tavares dan Silva menyoroti pentingnya *bounded context* dalam pengembangan sistem, di mana setiap layanan diwakili oleh *bounded context* yang independen dan dapat dikembangkan secara terpisah [20]. Dengan pemahaman ini, DDD memberikan pendekatan yang efisien untuk merancang sistem yang dapat beradaptasi dengan perubahan dan permintaan bisnis yang terus berkembang.

Komponen lain yang penting adalah *context map*, yang digunakan untuk mengidentifikasi hubungan antar *bounded contexts* dalam sistem. *Context map* memberikan gambaran tentang bagaimana *bounded contexts* berinteraksi dan menunjukkan dependensi di antara mereka. Dengan *context map*, pengelolaan sistem menjadi lebih jelas karena memberikan pemahaman menyeluruh tentang integrasi dan koordinasi antar bagian dalam sistem.

Subdomain dalam DDD adalah bagian spesifik dari domain yang dipecah menjadi bagian-bagian yang lebih kecil dan spesifik. Setiap subdomain mencakup fungsi yang terfokus sehingga lebih mudah dikelola dan diimplementasikan. Pemrosesan subdomain dapat dilakukan secara bertahap untuk memastikan setiap bagian diproses secara efisien dan menyeluruh.

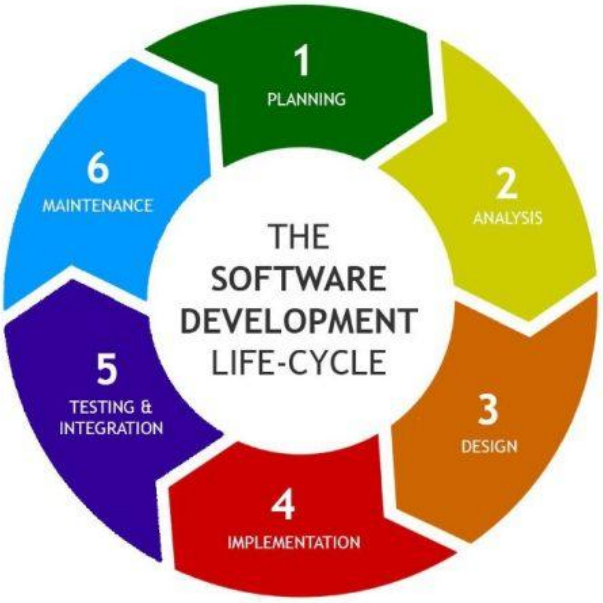
Di dalam DDD, juga terdapat elemen lain seperti *entities* dan *value objects*. *Entities* adalah objek dalam *domain* dengan identitas unik yang melekat pada mereka, seperti Mahasiswa atau Nilai dalam sistem akademik. *Value objects* mewakili konsep tanpa identitas unik, seperti tanggal presensi, namun tetap memiliki nilai. *Aggregates* adalah kumpulan *entities* dan *value objects* yang dikelola sebagai satu unit, menjaga konsistensi dalam domain, terutama saat menangani perubahan data dalam *bounded context* tertentu.

Pendekatan DDD ini membantu pengembang dalam menangani perubahan dalam domain bisnis, seperti perubahan aturan dalam sistem presensi kelas atau penilaian akademik, tanpa mengganggu kestabilan sistem secara keseluruhan. Prinsip-prinsip DDD ini tidak hanya memberikan panduan

teknis, tetapi juga mendukung komunikasi yang lebih baik antara tim pengembang dan ahli domain, memastikan bahwa solusi yang dikembangkan relevan dengan kebutuhan bisnis [18].

2.2.3. Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) adalah kerangka kerja yang menggambarkan tahapan yang terstruktur dalam pengembangan perangkat lunak. SDLC bertujuan untuk memastikan bahwa perangkat lunak yang dikembangkan memenuhi kebutuhan pengguna dengan kualitas yang tinggi, efisien, dan dapat dikelola dengan baik. Tahapan dalam SDLC biasanya mencakup perencanaan, analisis kebutuhan, desain, implementasi, pengujian, peluncuran, dan pemeliharaan.



Gambar 2 1. Tahapan SDLC

Menurut penelitian oleh Shylesh (2017), SDLC menyediakan panduan sistematis bagi pengembang untuk menangani kompleksitas proyek perangkat lunak, memastikan semua aspek terdokumentasi dengan baik, dan mengurangi risiko kesalahan selama siklus pengembangan [21]. Selain itu, penelitian dari Anisa, dkk. menyebutkan bahwa pendekatan SDLC sangat relevan dalam memastikan keselarasan perangkat lunak dengan kebutuhan pengguna karena tahapannya yang sistematis dan iteratif [22]. Samad, dkk. dalam penelitiannya

juga menyoroti pentingnya model SDLC dalam pengembangan sistem berbasis web untuk meningkatkan efisiensi dan kualitas pelayanan publik [23].

Dalam konteks *revamp* fitur kelola nilai dan presensi pada SIAKAD, penerapan SDLC memberikan pendekatan yang terorganisir dalam merancang, mengimplementasikan, dan menguji perubahan dari sistem lama (SPKP) ke sistem yang diperbarui. Dengan memanfaatkan metode SDLC, tim pengembang dapat memastikan bahwa semua fitur berfungsi sesuai kebutuhan pengguna serta memastikan integritas sistem selama proses transisi.

2.2.4. Prototyping

Metode *prototyping* atau *prototype* merupakan pendekatan *iteratif* yang sering digunakan dalam pengembangan perangkat lunak untuk menciptakan model awal dari sistem. Menurut penelitian oleh Pressman , metode ini memungkinkan pengembang dan pengguna untuk berkolaborasi dalam memahami kebutuhan sistem melalui prototipe awal yang dirancang untuk memvalidasi alur kerja utama [24].

Prototyping terdiri dari empat langkah utama sebagaimana dijelaskan oleh Pressman. Langkah pertama adalah pengumpulan kebutuhan, di mana kebutuhan dasar pengguna diidentifikasi sebagai acuan untuk membangun prototipe. Langkah kedua adalah pembuatan prototipe awal, yang merupakan representasi sederhana dari sistem seperti *wireframe* atau *mockup*, bertujuan untuk menggambarkan struktur dan alur kerja sistem secara umum. Selanjutnya, prototipe ini diuji oleh domain *expert* dalam tahap evaluasi untuk mendapatkan umpan balik. Tahap terakhir adalah penyempurnaan prototipe, di mana masukan dari domain *expert* diimplementasikan, sehingga prototipe semakin mendekati sistem akhir yang dirancang.

Dalam penelitian ini, metode *prototyping* digunakan pada tahap desain untuk merancang ulang fitur Kelola Nilai dan Presensi di SIAKAD Universitas Atma Jaya Yogyakarta. Prototipe awal berupa *wireframe* membantu memetakan struktur dasar antarmuka pengguna, sedangkan *mockup* digunakan untuk memperjelas elemen visual dan interaksi sistem. Dengan pendekatan ini, desain yang dihasilkan dapat divalidasi lebih awal, memastikan kesesuaian dengan kebutuhan yang telah diidentifikasi melalui analisis DDD.

2.2.5. Sistem Informasi Akademik (SIAKAD) Universitas Atma Jaya Yogyakarta

Universitas Atma Jaya Yogyakarta (UAJY) adalah salah satu perguruan tinggi swasta terkemuka di Indonesia yang memiliki komitmen tinggi terhadap peningkatan mutu pendidikan. Saat ini, UAJY memiliki 6 fakultas dan 28 program studi, yang mencakup jenjang sarjana, pascasarjana, dan doktoral, dengan total mahasiswa aktif mencapai 9.558 orang.

UAJY menggunakan sistem informasi bernama SIAKAD (Sistem Informasi Akademik) untuk mengelola berbagai aktivitas akademik. Sistem ini dirancang untuk memfasilitasi pengelolaan data akademik secara terpadu, termasuk pendaftaran mata kuliah, MBKM, dan administrasi akademik lainnya. Sistem SIAKAD mengalami *revamp* fitur untuk menggantikan fungsi pengelolaan nilai dan presensi yang sebelumnya dilakukan melalui SPKP, guna meningkatkan efisiensi dan kemudahan operasional bagi pengguna.