MAKALAH
REKAYASA PERANGKAT LUNAK BERORIENTASI OBJEK
"UNIFIED SOFTWARE DEVELOPMENT PROCESS (USDP)"

OLEH :

NAYLA ANANDA	(13020230112)
RASMAWATI	(13020230118)
NUR AQIDAH SAFANIKIAH	(13020230121)
AQILA JAMELLYA WULANDARI PUTRISYAH	(13020230155)
MUTIAH SHAULATIYA RISWAN	(13020230173)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS MUSLIM INDONESIA

2025

DAFTAR ISI

BAB I
PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi informasi yang pesat telah mendorong organisasi dan bisnis untuk beradaptasi dengan sistem pengelolaan data yang semakin efektif dan efisien, termasuk dalam bidang manajemen akademik maupun bisnis seperti pengelolaan dormitory dan membership gym. Sistem manual yang masih digunakan di berbagai institusi sering kali menimbulkan masalah seperti kerentanan terhadap kehilangan data, proses bisnis yang lambat, serta kesulitan dalam pelaporan dan pengawasan secara real-time. Hal ini berdampak pada penurunan efisiensi operasional dan daya saing dalam industri yang semakin kompetitif. Selain itu, tuntutan regulasi untuk penyimpanan elektronik dokumen sesuai peraturan nasional juga menjadi kebutuhan.

Unified Software Development Process (USDP) hadir sebagai salah satu solusi untuk mengatasi permasalahan tersebut. USDP merupakan metode pengembangan perangkat lunak yang berorientasi objek, use-case driven, serta incremental dan iterative, yang didukung oleh penggunaan UML sebagai alat pemodelan. Penerapan USDP tidak hanya memudahkan perancangan dan pengembangan perangkat lunak yang kompleks dan adaptif, tetapi juga mendorong pelibatan aktif seluruh stakeholder, sehingga hasil akhir lebih selaras dengan kebutuhan bisnis maupun pengguna akhir. Implementasi USDP terbukti mampu meningkatkan efisiensi, efektivitas, serta akurasi pengelolaan data dalam aplikasi web-based dormitory management ataupun sistem repository dokumen akademik.

B. Tujuan Penulisan

Tujuan penulisan makalah ini adalah:

1. Menjelaskan definisi dan sejarah model proses USDP sebagai salah satu kerangka kerja pengembangan perangkat lunak modern.
2. Mendeskripsikan penerapan model USDP pada berbagai aplikasi, seperti manajemen membership gym, dormitory management, dan repository dokumen akademik.
3. Menguraikan karakteristik model USDP, termasuk tim, alur proses, tahapan utama (inception, elaboration, construction, transition), serta keunikan pendekatan use-case-driven.

4. Menganalisis kelebihan dan kelemahan model USDP dibandingkan model lain, seperti waterfall, baik dari segi fleksibilitas, keterlibatan stakeholder, dan kemampuan adaptasi terhadap perubahan kebutuhan.
5. Menyajikan perbandingan antara USDP dengan minimal tiga model/metodologi lain (misal waterfall, RUP, XP), serta membahas bagaimana USDP mampu mengatasi kelemahan model konvensional.
6. Mengidentifikasi dan menjelaskan alat bantu model, seperti teknologi, aplikasi, atau UML yang digunakan dalam pengembangan perangkat lunak berbasis USDP.

BAB II
KAJIAN TEORITIS

A. Definisi dan Sejarah

Unified Software Development Process (USDP) merupakan metodologi pengembangan perangkat lunak yang pertama kali diperkenalkan oleh Rational Team. Metodologi ini kemudian dikembangkan dan disempurnakan menjadi Rational Unified Process (RUP), yang sekaligus menjadi cikal bakal terbentuknya berbagai metodologi pengembangan perangkat lunak berorientasi objek lainnya. USDP hadir sebagai respons terhadap kebutuhan industri perangkat lunak akan metodologi yang mampu menangani kompleksitas proyek besar serta adaptif terhadap perubahan kebutuhan selama proses pengembangan.

USDP didefinisikan sebagai suatu kerangka kerja proses pengembangan perangkat lunak yang bersifat iteratif dan incremental, berbasis komponen, berorientasi kebutuhan pengguna, dan berpusat pada arsitektur perangkat lunak. Metodologi ini menggunakan teknik pemodelan visual seperti Unified Modeling Language (UML) untuk menggambarkan sistem yang akan dikembangkan secara detil dan terstruktur.

Sejarah USDP erat kaitannya dengan perkembangan rekayasa perangkat lunak berorientasi objek. Metodologi ini dikembangkan oleh tiga tokoh penting, yaitu Grady Booch, Ivar Jacobson, dan James Rumbaugh, yang juga merupakan pencipta UML. Mereka berkontribusi dalam menciptakan suatu metode pengembangan perangkat lunak yang dapat memberikan panduan lengkap mulai dari analisis, desain, implementasi, hingga pengujian perangkat lunak.

Dengan pendekatan ini, USDP memberikan kerangka kerja yang terstruktur namun fleksibel, yang dapat disesuaikan dengan berbagai tipe dan skala proyek perangkat lunak. Karena sifatnya yang iterative, proyek dapat beradaptasi terhadap perubahan selama siklus hidup pengembangan, sembari menjaga kualitas dan integritas sistem yang dibangun.

USDP juga memungkinkan penggunaan kembali komponen perangkat lunak (coding reuse), yang mempercepat pengembangan aplikasi sejenis dan meningkatkan efisiensi pengembangan. Keuntungan lain termasuk peningkatan keterlibatan pemangku kepentingan sepanjang proses pengembangan, yang berdampak positif pada hasil akhir yang lebih sesuai kebutuhan pengguna.

Secara keseluruhan, USDP merupakan metodologi pengembangan perangkat lunak yang komprehensif, memadukan teknik pemodelan visual dengan praktik

pengelolaan proyek modern, sangat relevan untuk pengembangan sistem berorientasi objek dalam konteks proyek besar dan kompleks.

B. Penerapan Model

USDP terbagi atas empat fase utama yang harus dilakukan secara berurutan, yaitu Inception, Elaboration, Construction, dan Transition. Setiap fase memiliki karakteristik dan tujuan yang berbeda dalam siklus pengembangan perangkat lunak.

1. *Fase Inception* merupakan tahap permulaan di mana pengembang perangkat lunak melakukan interaksi dengan customer untuk mengidentifikasi kebutuhan-kebutuhan sistem yang hendak dibuat. Pada fase ini dilakukan penentuan konsep, visi, dan lingkup produk yang sedang dikembangkan. Tujuan utama fase ini adalah menentukan ruang lingkup sistem, mengidentifikasi kebutuhan sistem, dan melakukan analisis terhadap segala kemungkinan risiko yang mungkin terjadi selama pengerjaan proyek.
2. *Fase Elaboration* digunakan untuk mematangkan konsep-konsep yang sudah terbentuk di fase Inception. Fase ini belum masuk ke tahap pembuatan perangkat lunak secara langsung, tetapi lebih kepada pemantapan konsep dan peninjauan kembali terhadap rencana-rencana yang sudah ditentukan sebelumnya. Hasil akhir fase ini adalah arsitektur sistem yang lebih baik dan handal, analisis risiko yang telah diperbaiki, serta perencanaan detail untuk fase selanjutnya.
3. *Fase Construction* merupakan fase coding, di mana pengembang perangkat lunak mulai melakukan pembuatan sistem secara nyata. Pembuatan sistem tersebut harus mengacu kepada parameter-parameter yang sudah ditentukan dan digariskan dari fase-fase sebelumnya. Pada fase ini dilakukan implementasi perancangan yang sudah dibuat secara bertahap untuk beberapa iterasi.
4. *Fase Transition* merupakan tahap untuk mematangkan produk akhir yang sudah jadi. Pematangan ini perlu dilakukan untuk menganalisa apakah perangkat lunak yang sudah dibuat sesuai dengan kebutuhan pengguna. Fase ini mencakup pengujian sistem secara keseluruhan, penyiapan rilis produk, dan deployment sistem

C. Karakteristik Model

Metodologi USDP memiliki tiga karakteristik utama yang menjadi fondasinya, membedakannya dari metodologi pengembangan perangkat lunak lainnya.

1. *Use-Case Driven* (Digerakkan oleh Kasus Penggunaan)

Karakteristik ini berarti bahwa seluruh proses pengembangan berpedoman pada *use case*. *Use case* menggambarkan fungsionalitas sistem dari sudut pandang pengguna dalam format yang mudah dipahami. Diagram *use case* menjadi model sentral yang mengarahkan pengembangan model-model lainnya, seperti :Model Analisis: Memperhalus dan merinci definisi setiap *use case*.

- Model Perancangan: Mendefinisikan struktur statis aplikasi, seperti kelas dan antarmuka.
- Model Konstruksi: Merepresentasikan kode-kode program.
- Model Implementasi: Mendefinisikan pemetaan komponen ke perangkat keras.
- Model Pengujian: Mendeskripsikan skenario pengujian untuk validasi dan verifikasi.

Dengan berfokus pada fungsionalitas yang dibutuhkan pengguna, pendekatan ini membantu optimalisasi proses bisnis dan dapat mempersingkat waktu pengembangan.

2. *Architecture-Centric* (Berpusat pada Arsitektur)

USDP menempatkan arsitektur sistem sebagai pusat perhatian. Metodologi ini merupakan kerangka kerja pengembangan berbasis komponen, yang artinya perangkat lunak yang dihasilkan akan terdiri dari komponen-komponen yang saling terhubung melalui antarmuka yang terdefinisi dengan baik. Pendekatan ini memberikan dasar yang jelas untuk membentuk sistem dan mendukung berbagai model serta sudut pandang arsitektur, sehingga sistem yang kompleks dapat dibangun dengan lebih terstruktur.

3. *Iterative dan Incremental* (Berulang dan Bertahap)

Pengembangan perangkat lunak dengan USDP dilakukan secara iteratif (berulang) dan *incremental* (bertahap). Prosesnya dibagi menjadi beberapa fase, dan di dalam setiap fase terdapat beberapa iterasi. Setiap iterasi menghasilkan penambahan fitur atau penyempurnaan pada produk, memungkinkan tim untuk :

- Mengurangi risiko dengan melakukan evaluasi berkelanjutan.
- Meningkatkan kualitas produk secara bertahap.
- Mengantisipasi perubahan kebutuhan dari pengguna, karena pengujian dilakukan di akhir setiap fase.

a. Alur Kerja Iterasi (Peran Tim Teknis)

Di setiap fase, tim teknis seperti developer atau programmer melakukan serangkaian alur kerja secara iteratif. Aktivitas ini melibatkan berbagai peran dalam tim, seperti analis, desainer, dan penguji. Enam alur kerja inti tersebut adalah:

- *Business Modeling:* Memahami dan memodelkan proses bisnis organisasi.
- *Requirements:* Mengumpulkan, menganalisis, dan mendefinisikan kebutuhan fungsional dan non-fungsional sistem.
- *Analysis & Design:* Menerjemahkan kebutuhan menjadi spesifikasi desain teknis menggunakan UML, seperti diagram kelas dan sekuens.
- *Implementation:* Menulis kode program berdasarkan model desain.
- *Test:* Melakukan verifikasi dan validasi untuk memastikan perangkat lunak berfungsi sesuai harapan dan bebas dari *bug*.
- *Deployment:* Menginstal dan mendistribusikan perangkat lunak kepada pengguna akhir.

Selain peran manajer proyek dan developer, USDP menekankan pentingnya keterlibatan aktif dari semua pemangku kepentingan (*stakeholders*), termasuk pengguna, dalam setiap tahap pengembangan untuk memastikan produk akhir sesuai dengan kebutuhan.

D. Kelebihan dan kelemahan

USDP memiliki berbagai kelebihan yang membuatnya menjadi pilihan metodologi pengembangan perangkat lunak yang efektif. Kelebihan utama USDP adalah kemampuannya untuk mengidentifikasi dan menyelesaikan risiko proyek melalui manajemen permintaan dan review yang lebih berhati-hati. Hal ini sangat penting dalam pengembangan perangkat lunak karena dapat mencegah kegagalan proyek yang disebabkan oleh risiko yang tidak teridentifikasi.

USDP juga cukup scalable, membuatnya cocok untuk tim dan proyek dengan skala besar maupun kecil, menawarkan fleksibilitas yang signifikan dalam manajemen proyek. Metodologi ini dapat diterapkan oleh single developer maupun tim yang besar. Review berkala dalam USDP membantu menjaga fokus dan meningkatkan transparansi, aspek krusial untuk kesuksesan proyek.

Keunggulan lain dari USDP adalah dapat menerima perubahan untuk meningkatkan prototipe yang ada sehingga dapat menghasilkan sistem yang dapat diterima, dan perubahan yang terjadi dianggap sebagai bagian dari proses pengembangan itu sendiri. USDP juga dapat mengantisipasi definisi kebutuhan sistem yang kurang detail pada tahap awal melalui pengujian di akhir setiap fase.

Namun, USDP juga memiliki kelemahan yang perlu dipertimbangkan. Kekurangan utama adalah proses pengembangan yang kompleks yang membutuhkan keterampilan mendalam dari tim, yang dapat menjadi tantangan bagi organisasi dengan sumber daya terbatas. Pengujian komponen yang berkelanjutan juga meningkatkan kompleksitas dan berpotensi menimbulkan banyak masalah saat pengujian, yang memerlukan perencanaan dan sumber daya yang cermat untuk diatasi.

E. Perbedaan USDP Dengan Metode Lainnya

USDP memiliki perbedaan yang signifikan dengan metodologi pengembangan perangkat lunak lainnya. Jika dibandingkan dengan metodologi waterfall, USDP lebih kompleks dalam implementasinya. Waterfall menggunakan pendekatan sequential dan linear, di mana setiap fase harus diselesaikan sebelum berpindah ke fase berikutnya. Sebaliknya, USDP menggunakan pendekatan iteratif dan incremental, di mana setiap fase dapat dilakukan secara berulang untuk memperbaiki dan menyempurnakan hasil.

Perbandingan dengan metodologi Agile menunjukkan bahwa USDP termasuk dalam kategori heavy-weight methodologies, sementara Agile merupakan light-weight methodologies. Agile lebih fokus pada fleksibilitas dan respon cepat terhadap perubahan, sementara USDP lebih menekankan pada dokumentasi yang lengkap dan arsitektur yang solid. Dalam hal manajemen risiko, USDP memiliki pendekatan yang lebih terstruktur dan formal dibandingkan dengan Agile.

Perbedaan dengan Extreme Programming (XP) terletak pada tingkat formalitas dan dokumentasi. XP menekankan pada coding dan testing yang intensif dengan dokumentasi minimal, sementara USDP memerlukan dokumentasi yang comprehensive dan pemodelan yang detail menggunakan UML. USDP juga lebih cocok untuk proyek berskala besar dengan tim yang besar, sementara XP lebih efektif untuk tim kecil dengan proyek yang relatif sederhana.

Scrum, sebagai salah satu framework Agile, memiliki pendekatan yang berbeda dalam hal struktur fase. Scrum menggunakan Sprint sebagai unit iterasi yang lebih pendek (2-4 minggu), sementara USDP memiliki fase yang lebih panjang dan terstruktur. Scrum juga tidak memiliki proses khusus untuk manajemen risiko, sehingga banyak peneliti mencoba mengintegrasikan manajemen risiko secara eksplisit ke dalam setiap proses siklus hidup pengembangan perangkat lunak dalam metodologi Scrum.

F. Alat bantu

USDP menggunakan Unified Modeling Language (UML) sebagai alat pemodelan utama untuk perangkat lunak yang dikembangkan. UML merupakan hasil kompilasi best engineering practice yang sudah terbukti sukses dalam pemodelan sistem yang besar dan kompleks, khususnya pada level arsitektural. UML mendukung tiga aspek utama dalam pemodelan sistem: Functional Model menggunakan Use Case Diagram, Object Model menggunakan Class Diagram, dan Dynamic Model menggunakan Sequence Diagram, Activity Diagram, dan Statechart Diagram.

Dibandingkan dengan alat bantu yang digunakan dalam metodologi Waterfall, USDP lebih lengkap dalam hal pemodelan visual. Waterfall umumnya menggunakan tools sederhana seperti flowchart dan Data Flow Diagram (DFD), sementara USDP menggunakan comprehensive UML diagrams yang dapat menggambarkan sistem dari berbagai sudut pandang. Hal ini membuat USDP lebih unggul dalam menangani kompleksitas sistem berorientasi objek.

Untuk Agile methodologies, tools yang digunakan lebih bervariasi dan cenderung lightweight. Agile sering menggunakan user stories, burndown charts, dan kanban boards untuk tracking progress. Meskipun Agile juga dapat menggunakan UML, penggunaannya tidak seasif dalam USDP. Perbedaan ini mencerminkan filosofi masing-masing metodologi, di mana USDP lebih menekankan pada dokumentasi dan pemodelan yang detail.

Scrum menggunakan tools seperti product backlog, sprint backlog, dan scrum board untuk manajemen proyek. Dalam hal risk management tools, Scrum tidak memiliki tools khusus, berbeda dengan USDP yang memiliki framework manajemen risiko yang terintegrasi dalam setiap fase. Beberapa penelitian mencoba mengembangkan software agents untuk mendukung identifikasi, assessment, dan monitoring risiko dalam Agile development.

Modern software development tools yang mendukung USDP antara lain IBM Rational Rose untuk UML modeling, Microsoft Visio untuk diagram creation, dan berbagai IDE yang mendukung UML seperti Eclipse dengan plugin UML. Tools ini membantu mengatasi kelemahan USDP dalam hal kompleksitas dengan menyediakan interface yang user-friendly dan automation dalam pembuatan dokumentasi.

Penggunaan tools yang tepat dapat mengatasi beberapa kelemahan USDP, seperti kompleksitas proses dan kebutuhan skill yang tinggi. Automated testing tools, continuous integration tools, dan project management software dapat membantu tim dalam mengimplementasikan USDP dengan lebih efisien. Integration dengan modern

DevOps tools juga memungkinkan USDP untuk lebih adaptif terhadap perubahan requirements tanpa mengorbankan kualitas arsitektur dan dokumentasi yang menjadi kekuatan utama metodologi ini.