JENIS-JENIS

SOFTWARE DEVELOPMENT LIFE CYCLE

Mata Kuliah: Software Engineering

DOSEN: Yudhi Fajar Saputra, S.Kom., M.Sc

Pertemuan ke-3

Topik Bahasan ke-8

SEMESTER : 3/ TA. 2024-2025

KODE MK/SKS: MKP001/3 SKS

PRODI INFORMATIKA/ILMU KOMPUTER

UNIVERSITAS WIDYA GAMA MAHAKAM SAMARINDA

Nama Mata Kuliah	: Software Engineeing/Rekayasa Perangkat Lunak
Kode Mata Kuliah/SKS	: **MKP ____/3 SKS**
Dosen	: **Yudhi Fajar Saputra,**
Semester	: **3/ 2024**
Hari Pertemuan / Jam	: -
Tempat Pertemuan	: **Ruang Kelas A.06**

Model Software Development Life Cycle (SDLC) adalah kerangka kerja yang digunakan untuk mengorganisir dan mengelola pengembangan perangkat lunak. Beberapa pakar menjelaskan ada banyak Model SDLC untuk mengembangkan aplikasi atau software, Ian Sommerville menjelaskan didalam bukunya bahwa terdapat 5 jenis model SDLC yaitu Waterfall, V-Model, Incremental Development, Spiral, Agile [1]. Pressman dalam bukunya menjelaskan ada 6 model janis SDLC untuk mengembangkan SDLC, diantaranya Waterfall, Incremental, Evolutionary (Prototyping), Spiral, Component-based Development, Formal Methods [2]. Sementara Ken Schwaber dan Jeff Sutherland menyebutkan dalam bukunya sebuah model pengembangan aplikasi atau software yang bernama Agile [3] , berikut penjelasan lebih detail mengenai model-model dari SDLC:

1. WATERFALL MODEL

Waterfall model merupakan model pengembangan aplikasi atau software yang terdiri dari serangkaian fase yang linier dan sekuensial, di mana setiap fase harus diselesaikan sebelum melanjutkan ke fase berikutnya [4], mulai dari pengumpulan kebutuhan hingga pemeliharaan sehingga Waterfall Model dianggap sangat cocok untuk proyek pengembangan aplikasi atau software dengan persyaratan yang jelas dan mudah dipahami sejak awal. Secara umum, tahapan dalam Model Waterfall dapat dibagi menjadi beberapa tahap utama, diantara adalah sebagai berikut:

a. Analisis Kebutuhan (Requirement Analysis)

Tahap ini melibatkan pengumpulan semua kebutuhan sistem yang akan dikembangkan. Kebutuhan ini didokumentasikan dan dianalisis untuk memastikan pemahaman yang jelas sebelum melanjutkan ke tahap berikutnya, sehingga dapat dirincikan sebagai berikut:

- **Tujuan**: Memahami dengan jelas kebutuhan pengguna dan sistem.
- **Aktivitas**: Mengumpulkan, menganalisis, dan mendokumentasikan semua kebutuhan pengguna.
- **Hasil**: Dokumen spesifikasi kebutuhan yang detail.

b. Perancangan (Design)

Pada tahap ini, arsitektur sistem dirancang berdasarkan persyaratan yang telah ditentukan. Ini termasuk desain high-level (struktur sistem) dan desain low-level (desain rinci), sehingga dapat dirincikan sebagi berikut:

- **Tujuan**: Menerjemahkan kebutuhan menjadi desain teknis yang lebih detail.
- **Aktivitas**: Membuat diagram alur, diagram entitas-relasi, dan spesifikasi antarmuka.
- **Hasil**: Desain sistem yang lengkap dan rinci

c. Implementasi (Implementation)

Pada tahap ini, desain yang sudah dibuat diterjemahkan ke dalam kode program. Tim pengembang mulai menulis dan mengembangkan perangkat lunak sesuai dengan desain yang sudah dibuat, sehingga dapat dirincikan sebagi berikut:

- **Tujuan**: Membangun perangkat lunak berdasarkan desain yang telah dibuat.
- **Aktivitas**: Menulis kode program, membangun database, dan konfigurasi perangkat lunak.
- **Hasil**: Kode program yang berfungsi.

d. Pengujian (Testing)

Setelah implementasi, semua komponen sistem digabungkan dan diuji untuk memastikan bahwa mereka bekerja sesuai dengan persyaratan. Pengujian dilakukan untuk menemukan dan memperbaiki bug atau kesalahan, sehingga dapat dirincikan sebagi berikut:

- **Tujuan**: Memastikan perangkat lunak berfungsi sesuai dengan spesifikasi yang telah ditentukan.
- **Aktivitas**: Melakukan berbagai jenis pengujian, seperti pengujian unit, pengujian integrasi, dan pengujian sistem.
- **Hasil**: Laporan hasil pengujian yang menunjukkan bahwa perangkat lunak telah memenuhi persyaratan kualitas.

e. Penerapan (Deployment)

Setelah pengujian berhasil, perangkat lunak siap untuk di-deploy atau diterapkan di lingkungan produksi. sehingga dapat dirincikan sebagi berikut:

- **Tujuan**: Meluncurkan perangkat lunak ke lingkungan produksi.
- **Aktivitas**: Melakukan instalasi, konfigurasi, dan migrasi data.
- **Hasil**: Perangkat lunak yang siap digunakan oleh pengguna akhir.

f. Pemeliharaan (Maintenance)

Setelah perangkat lunak di-deploy, fase pemeliharaan dimulai. Pemeliharaan

mencakup perbaikan bug, pembaruan, dan peningkatan sistem yang berkelanjutan berdasarkan *feed back* para pengguna dan perubahan *environment,* sehingga dapat dirincikan sebagi berikut:

- **Tujuan**: Mempertahankan dan meningkatkan perangkat lunak setelah diluncurkan.
- **Aktivitas**: Melakukan perbaikan bug, menambahkan fitur baru, dan mengadaptasi perangkat lunak.
- **Hasil**: Perangkat lunak yang terus bisa beradaptasi.

2. V-MODEL

V-Model adalah sebuah pengembangan dari Model Waterfall yang menekankan pada verifikasi dan validasi pada setiap tahap pengembangan. Bentuk "V" pada model ini menggambarkan hubungan yang erat antara setiap tahap pengembangan dengan tahap pengujian, yang mana Model ini menggabungkan pengembangan di sisi kiri V dan validasi di sisi kanan V, tahapan validasi memastikan bahwa produk akhir memenuhi kebutuhan pengguna [5]. Tahapan dalam V-Model dapat dibagi menjadi dua bagian utama:

a. Tahap Pengembangan (Sisi Kiri V):

1) Analisis Kebutuhan: tahap ini fokus pada pengumpulan dan analisis kebutuhan pengguna.
2) Perancangan Sistem: Merancang arsitektur sistem secara keseluruhan berdasarkan persyaratan yang telah diidentifikasi yang meliputi desain arsitektur, basis data, antarmuka, dan spesifikasi tingkat tinggi lainnya..
3) Perancangan Arsitektur: Membuat desain lebih detail dari sistem, termasuk modul, komponen, dan bagaimana saling berinteraksi antar komponen.
4) Perancangan Modul: Desain modul-modul individu yang akan membentuk sistem, meliputi algoritma, struktur data, dan sebagainya.

b. Tahap Pengujian (Sisi Kanan V):

1) Unit Testing: Menguji setiap modul secara individu untuk memastikan fungsinya sesuai dengan desain.
2) IntegrationTesting: Menguji bagaimana modul-modul saling berinteraksi dan berkomunikasi.
3) Sistem Testing: Menguji sistem secara keseluruhan untuk memastikan semua fungsi bekerja sesuai dengan spesifikasi.
4) User Acceptance Testing (UAT): Menguji sistem oleh pengguna akhir untuk memastikan sistem tersebut memenuhi harapan end-user.

3. AGILE MODEL

Agile model merupakan model pengembangan yang dilakukan secara iteratif, berulang-ulang, dan adaptif terhadap perubahan, , metode Agile sangat efektif bagi tim pengembangan untuk merespon kebutuhan dan perubahan proyek dengan cepat. James Shore dan Shane Warden dalam bukunya menggambarkan bahwa Agile sebagai pendekatan untuk pengembangan perangkat lunak yang mengutamakan interaksi manusia, fleksibilitas, dan kolaborasi [6]. Meskipun tidak ada standar yang baku untuk tahapan dalam Agile Model, namun secara umum Model Agile memiliki beberapa tahapan diantaranya:

a. Planning (Perencanaan):

Tujuan: Mengidentifikasi fitur-fitur utama yang diinginkan, merencanakan sprint atau iterasi, dan menentukan backlog produk (daftar fitur dan perbaikan yang harus dikerjakan).

Output: Backlog produk yang diprioritaskan dan rencana sprint pertama.

b. Design (Desain):

Tujuan: Desain dilakukan pada model agile mencakup perancangan secara interaktif yang dapat dilakukan secara penyesuaian selama pengembangan.

Output: Desain yang cukup untuk memulai pengembangan sprint pertama.

c. Development (Pengembangan):

Tujuan: Mengembangkan fitur yang telah direncanakan dalam sprint atau iterasi. Pengembangan dilakukan secara iteratif, di mana setiap iterasi menghasilkan bagian dari produk yang bisa diuji.

Output: Inkrement produk yang berfungsi.

d. Testing (Pengujian):

Tujuan: Pengujian dilakukan pada setiap iterasi, memastikan setiap bagian perangkat lunak yang dihasilkan memenuhi standar kualitas dan persyaratan.

Output: Versi produk yang sudah diuji dan berfungsi.

e. Review and Retrospective (Tinjauan dan Retrospektif):

Tujuan: Setelah setiap sprint, tim dan pemangku kepentingan meninjau hasil sprint dan membahas apa yang berjalan dengan baik dan apa yang perlu ditingkatkan.

Output: Tanggapan yang diintegrasikan ke dalam iterasi berikutnya, bersama dengan perbaikan proses jika diperlukan.

f. Deployment (Penerapan):

Tujuan: Setelah beberapa iterasi, produk atau fitur yang telah diuji dan berfungsi

sepenuhnya akan di-deploy ke lingkungan produksi.

Output: Produk atau fitur yang siap digunakan oleh pengguna akhir.

g. Maintenance (Pemeliharaan):

Tujuan: Memperbaiki bug dan merespons tanggapan dari pengguna, sering kali dalam iterasi yang lebih pendek atau melalui pembaruan berkelanjutan.

Output: Perangkat lunak yang tetap stabil dan ditingkatkan berdasarkan umpan balik pengguna.

4. DEVOPS MODEL

DevOps adalah pendekatan dalam membangun aplikasi atau software yang menggabungkan pengembangan (Development) dan operasi IT (Operations) untuk mempercepat siklus pengembangan, meningkatkan kualitas produk, dan memfasilitasi pengiriman yang lebih cepat dan andal, Sehingga tujuan utama DevOps adalah untuk meningkatkan kecepatan dan kualitas pengiriman perangkat lunak, serta meningkatkan kolaborasi antara tim pengembangan dan operasi. DevOps dijelaskan oleh Gene Kim bahwa Devops sebagai pendekatan yang mengintegrasikan proses pengembangan dan operasi untuk menciptakan alur kerja yang lebih efektif dan efisien [7]. Model Devops memiliki beberapa tahapan diantaranya:

a. Plan (Perencanaan): Tahap ini yang mencakup mulai perencanaan proyek hingga menentukan apa saja yang diperlukan untuk pengembangan perangkat lunak.
b. Code (Pengkodean): Tim pengembang mulai menulis kode perangkat lunak berdasarkan rencana yang telah dibuat.
c. Build (Pembangunan): Kode yang ditulis diintegrasikan dan dikompilasi menjadi aplikasi atau komponen yang dapat dijalankan.
d. Test (Pengujian): Aplikasi atau komponen yang telah dibangun diuji untuk memastikan bahwa mereka berfungsi dengan benar dan memenuhi kebutuhan yang telah ditentukan.
e. Release (Rilis): Setelah pengujian selesai dan aplikasi siap, aplikasi tersebut dirilis ke lingkungan produksi.
f. Deploy (Penerapan): Aplikasi yang telah dirilis diterapkan ke server produksi dan siap digunakan oleh pengguna.
g. Operate (Operasi): Aplikasi yang telah diterapkan dikelola dan dipantau untuk memastikan bahwa mereka berfungsi dengan baik dalam lingkungan produksi.
h. Monitor (Pemantauan): Kinerja aplikasi dipantau secara terus-menerus untuk mendeteksi dan memperbaiki masalah yang mungkin muncul

5. ITERATIVE MODEL

Iterative Model adalah pendekatan pengembangan perangkat lunak di mana proyek dibagi menjadi beberapa siklus atau iterasi. Setiap iterasi mencakup perencanaan, desain, pengembangan, pengujian, dan evaluasi. Van Vliet menguraikan Iterative Model sebagai proses yang memungkinkan revisi dan perbaikan secara bertahap [8]. Model ini sangat efektif dalam proyek-proyek di mana kebutuhan awal tidak sepenuhnya dipahami atau di mana perubahan mungkin terjadi selama pengembangan. Model Iterative memiliki beberapa tahapan diantaranya

a. Requirements (Kebutuhan)

Tujuan: Mengumpulkan dan memahami kebutuhan sistem yang akan dibangun. Pada tahap ini, tidak semua kebutuhan harus ditentukan secara detail, karena akan ada kesempatan untuk mengumpulkan dan memperbaiki kebutuhan dalam iterasi berikutnya.

Output: Kebutuhan fungsional dan non-fungsional utama, yang akan menjadi dasar untuk iterasi pertama.

b. Design (Desain)

Tujuan: Merancang arsitektur dan komponen sistem berdasarkan kebutuhan yang telah dikumpulkan. Desain dibuat secara modular, sehingga dapat diadaptasi dan diperbaiki dalam iterasi berikutnya.

Output: Desain arsitektur dan komponen utama, yang siap untuk diimplementasikan.

c. Implementation (Implementasi)

Tujuan: Melakukan koding berdasarkan desain yang telah dibuat. Pengembangan dilakukan secara bertahap, dengan fokus pada fitur atau modul tertentu dalam setiap iterasi.

Output: Kode perangkat lunak yang sudah dikembangkan sesuai dengan desain, biasanya sebagian dari sistem secara keseluruhan.

d. Testing (Pengujian)

Tujuan: Menguji perangkat lunak yang telah diimplementasikan untuk memastikan bahwa perangkat lunak tersebut telah memenuhi kebutuhan yang ditentukan dan bekerja sesuai harapan. Pengujian dilakukan pada tingkat unit, integrasi, dan sistem.

Output: Hasil pengujian, termasuk bug atau masalah yang perlu diperbaiki.

e. Evaluation (Evaluasi)

Tujuan: Menilai hasil dari iterasi yang sudah dilakukan, mengumpulkan tanggapan dari pengguna atau pemangku kepentingan, dan mengidentifikasi

hal-hal yang perlu perbaikan atau penyesuaian yang diperlukan untuk iterasi berikutnya.

Output: Umpan balik dan rencana untuk iterasi berikutnya, termasuk perbaikan pada kebutuhan, desain, atau implementasi.

f. Refinement (Penyempurnaan)

Tujuan: Berdasarkan hasil evaluasi, perbaikan dilakukan pada desain, implementasi, dan pengujian di iterasi berikutnya. Tahap ini mungkin melibatkan revisi kebutuhan, penyesuaian desain, atau pengembangan ulang komponen tertentu.

Output: Versi yang diperbarui dari perangkat lunak, siap untuk iterasi berikutnya.

g. Deployment (Penerapan)

Tujuan: Jika iterasi menghasilkan versi perangkat lunak yang stabil dan siap digunakan, maka perangkat lunak dapat diterapkan ke environment produksi atau diserahkan untuk pengujian akhir.

Output: Perangkat lunak yang diterapkan atau diserahkan ke pengguna akhir atau pengujian lebih lanjut.

6. SPIRAL MODEL

Spiral Model adalah salah satu model dalam Siklus Hidup Pengembangan Perangkat Lunak (SDLC) yang menggabungkan elemen dari model Waterfall dan model Iteratif dimana setiap elemen atau fase pada Waterfall Model mencakup perencanaan, analisis risiko, pengembangan, pengujian, dan evaluasi sehingga Spiral Model sangat berfokus pada manajemen risiko yang sangat berguna untuk proyek yang kompleks dan dinamis. Untuk itu Spiral Model memungkinkan tim untuk bereaksi terhadap perubahan dengan lebih baik, dengan memfokuskan pada manajemen risiko dan evaluasi berulang [9]. Model Spiral memiliki beberapa tahapan diantaranya:

a. Identifikasi Tujuan dan Perencanaan

Tujuan: Menentukan tujuan spesifik untuk iterasi atau fase tertentu dalam pengembangan perangkat lunak. Ini termasuk penetapan kebutuhan, fungsi, dan kriteria performa yang harus dicapai.

Aktivitas: Mengidentifikasi kebutuhan pengguna dan sistem, Menentukan tujuan, batasan, dan fitur utama, Merencanakan aktivitas dan menetapkan jadwal untuk iterasi berikutnya.

Output: Dokumen perencanaan iterasi yang mencakup tujuan, jadwal, dan sumber daya yang dibutuhkan.

b. Analisis Risiko dan Evaluasi Alternatif

Tujuan: Mengidentifikasi dan menganalisis risiko yang terkait dengan tujuan iterasi dan mengevaluasi alternatif solusi.

Aktivitas: Mengidentifikasi potensi risiko yang bisa mempengaruhi keberhasilan proyek (misalnya, risiko teknis, manajemen, atau finansial), Menganalisis dampak dari setiap risiko, Mengevaluasi berbagai pendekatan untuk mengurangi risiko, termasuk prototyping, simulasi, atau studi kelayakan.

Output: Laporan analisis risiko yang merinci risiko yang diidentifikasi dan strategi mitigasi.

c. Pengembangan dan Validasi

Tujuan: Mengembangkan solusi teknis berdasarkan perencanaan dan analisis risiko, serta melakukan validasi terhadap hasil pengembangan.

Aktivitas: Merancang dan mengimplementasikan prototipe atau modul sistem., Melakukan pengujian terhadap solusi yang dikembangkan untuk memastikan bahwa kebutuhan terpenuhi., Melakukan evaluasi untuk menentukan apakah solusi yang dihasilkan dapat diterima atau perlu direvisi.

Output: Prototipe atau versi awal dari perangkat lunak yang telah diuji dan divalidasi.

d. Evaluasi dan Perencanaan Iterasi Berikutnya

Tujuan: Mengevaluasi hasil dari iterasi saat ini dan merencanakan langkah-langkah untuk iterasi berikutnya.

Aktivitas: Mengevaluasi kinerja dan hasil dari iterasi yang telah selesai, Mengumpulkan umpan balik dari pemangku kepentingan, Menyempurnakan rencana untuk iterasi berikutnya berdasarkan evaluasi dan umpan balik.

Output: Laporan evaluasi iterasi dan rencana perbaikan untuk iterasi berikutnya.

7. DAFTAR REFERENSI

1. Sommerville, Ian. (2015). Software Engineering 10th Edition. Pearson Education, Inc.. ISBN-13: 978-0-13-703515-1.
2. Roger S. Pressman S. R, Maxim. B. (2014). Software Engineering: A Practitioner's Approach 8th Edition. McGraw Hill. ISBN 9780078022128.
3. Ken Schwaber & Jeff Sutherland. (2020). The Scrum Guide. 2020.
4. Joseph Ingeno. (2018). Software Architect's Handbook. Packt Publishing. ISBN 9781788624060.
5. Rajib Mall. (2018.) Fundamentals Of Software Engineering, 5Th Ed. PHI LearningPrivate Limited. ISBN 978-93-88028-02-8.

6. James Shore & Shane Warden. (2021) . The Art of Agile Development, 2nd Edition. O'Reilly Media, Inc. ISBN: 9781492080695.
7. Gene Kim, Kevin Behr, & George Spafford. (2018). The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win 3rd Edition. IT Revolution Press. ISBN: 0988262592 ;
8. Hans van Vliet. (2021). Software Engineering: Principles and Practice, 4th Edition. Wiley. ISBN: 978-0-470-03146-9.
9. Philippe Kruchten , Peggy Gregory. (2023). Agile Processes in Software Engineering and Extreme Programming. Springer Nature Switzerland. ISBN: 3031485491

8. Daftar Bacaan

1. Sama seperti pada daftar referensi

9. JADWAL PERKULIAHAN DAN TOPIK BAHASAN

Pertemuan Ke-	TOPIK BAHASAN	BACAAN
1	a. Kontrak Perkuliahan, Perkenalan dan Penjelasan b. Pengenalan Rekayasa Perangkat Lunak	Kontrak Perkuliahan
2	a. Karakteristik perangkat lunak b. Komponen perangkat lunak c. Model perangkat lunak d. Fungsi dan peran dari software engineer	1-6
3	a. Definisi SDLC b. Jenis-jenis SDLC	Idem
4	a. Observasi dan estimasi dalam perencanaan proyek b. Tujuan perencanaan proyek c. Manajemen proyek perangkat lunak yang efektif	Idem
5	a. Proses analisis kebutuhan b. Metode analisis kebutuhan c. Spesifikasi dan validasi kebutuhan	Idem
6	a. Perangkat bantu proses analisis kebutuhan b. Konsep dasar, Konteks, Proses, dan Prinsip Perancangan Perangkat Lunak; c. Isu mendasar dalam perancangan perangkat lunak	Idem
7	a. Alat bantu perancangan (DFD dan UML) b. Macam-macam diagram yang terdapat pada UML (Class Diagram, Use Case Diagram, Activity Diagram, Sequence Diagram)	Idem

8	**UTS**	
9	a. Konsep dan Isu dalam b. Desain User Interface c. Prinsip Desain antarmuka (user experience, user guidance, user diversity) d. Software configuration management: definisi dan skenario kerja	Idem
10	a. Perencanaan dalam pengujian b. Proses testing: (black box testing, white box testing) c. Integration testing dan user testing d. Faults, Error dan Failures	Idem
11	Review Teknik Pengujian Perangkat Lunak dari proses testing	Idem
12	a. Pengujian unit b. Pengujian integrasi c. Pengujian sistem d. Debugging dan quality assurance	Idem
13	a. Quality assurance pada perangkat lunak b. Keamanan data akses	Idem
14	a. Definisi pemeliharaan perangkat lunak. b. Konsep Pemeliharaan Perangkat lunak	Idem
15	Teknik pemeliharaan perangkat lunak (Pemeliharaan korektif, pemeliharaan adaptif, pemeliharaan perfektif, pemeliharaan preventif)	Idem
16	**UAS**	