SPESIFIKASI DAN VALIDASI KEBUTUHAN

Mata Kuliah: Software Engineering

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

Pertemuan ke-5

Topik Bahasan ke-12

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**

Spesifikasi kebutuhan perangkat lunak atau sering disebut software requirement specification (SRS) dan validasi kebutuhan perangkat lunak atau juga sering disebut software requirement validation (SRV) merupakan dokumen yang sangat penting dalam memulai pengembangan perangkat lunak. Dengan memiliki SRS dan SRV yang baik, proyek pengembangan dapat berjalan lebih efisien dan memiliki peluang yang lebih besar untuk sukses. Dokumen ini memastikan bahwa semua pihak memiliki pemahaman yang sama tentang apa yang diharapkan dari perangkat lunak, sehingga mengurangi risiko kesalahan dan meningkatkan kualitas produk akhir. Standar IEEE 830-1998 menyatakan bahwa SRS harus mendeskripsikan dengan jelas dan tepat setiap kebutuhan penting dari perangkat lunak, termasuk fungsi, kinerja, batasan desain, dan atribut serta antarmuka eksternal sedangkan SRV memastikan bahwa dokumen kebutuhan perangkat lunak mencerminkan kebutuhan dan harapan aktual dari pemangku kepentingan dan tidak mengandung kesalahan atau kelalaian, [1].

1. SPESIFIKASI KEBUTUHAN - SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Software Requirements Specification (SRS) adalah dokumen formal yang menguraikan kebutuhan dan spesifikasi dari perangkat lunak yang akan dikembangkan. SRS mendefinisikan secara detail semua aspek yang diharapkan dari perangkat lunak, termasuk fungsi, kinerja, antarmuka, dan batasan-batasan lainnya. Dokumen ini menjadi acuan bagi tim pengembang, penguji, dan pemangku kepentingan lainnya sepanjang siklus hidup proyek perangkat lunak. Sommerville menjelaskan bahwa SRS sebagai pernyataan resmi tentang apa yang harus diimplementasikan oleh pengembang sistem. SRS harus mencakup definisi kebutuhan pengguna dan spesifikasi kebutuhan sistem [2]. Sedangkan wiegers dan beatty menyatakan bahwa SRS adalah dokumen terstruktur yang merinci fungsionalitas yang harus disediakan oleh sistem, batasan-batasan di mana sistem harus beroperasi, serta kebutuhan sistem lainnya[3] . Maka dari itu SRS adalah dokumen yang sangat penting dalam pengembangan perangkat lunak, berfungsi sebagai panduan untuk memastikan bahwa semua kebutuhan pengguna dan spesifikasi teknis dipahami dengan jelas dan diimplementasikan dengan benar oleh tim pengembang. Dengan mendokumentasikan kebutuhan secara rinci, SRS

membantu meminimalkan kesalahpahaman, mengurangi risiko kesalahan, dan memastikan bahwa perangkat lunak yang dihasilkan memenuhi harapan dan kebutuhan pemangku kepentingan.

Pada bahasan Software Requirements Specification (SRS) akan membahas mengenai tujuan SRS, Komponen Utama SRS, dan Manfaat SRS

a. Tujuan SRS

Tujuan utama dari SRS adalah untuk memastikan bahwa semua pihak yang terlibat dalam proyek pengembangan perangkat lunak memiliki pemahaman yang sama tentang apa yang akan dibangun, berikut adalah tujuan SRS secara detail:

1) Sebagai Panduan Pengembangan: SRS berfungsi untuk menyediakan referensi yang jelas bagi pengembang untuk merancang, mengimplementasikan, dan menguji perangkat lunak. SRS berfungsi sebagai spesifikasi formal dari kebutuhan perangkat lunak. Dokumen ini memberikan deskripsi yang detail dan terstruktur dari kebutuhan sistem yang harus dipenuhi [2], sehingga SRS memastikan bahwa tim pengembang memiliki pemahaman yang seragam tentang apa yang harus dibangun.
2) Sebagai Dasar Pengujian: SRS berfungsi sebagai acuan untuk mengembangkan rencana pengujian, memastikan bahwa setiap kebutuhan fungsional dan non-fungsional diuji untuk memverifikasi bahwa perangkat lunak berfungsi sesuai dengan spesifikasi, sehingga dengan SRS penguji memiliki acuan yang tepat untuk mengukur apakah perangkat lunak memenuhi spesifikasi yang telah ditentukan [4] .
3) Sebagai Alat Komunikasi: Dokumen SRS menjadi alat komunikasi yang penting antara tim pengembang, pengguna, dan pemangku kepentingan lainnya [4], yang bertujuan untuk mengurangi kemungkinan terjadinya miskomunikasi dan kesalahpahaman. Sehingga dapat memastikan bahwa kebutuhan pengguna, pengembang, dan pemangku kepentingan lainnya diartikan dan dipahami dengan cara yang sama.
4) Untuk meminimalisir Risiko: Dengan mendefinisikan kebutuhan secara jelas sejak awal, SRS membantu untuk meminimalisir risiko perubahan besar yang berdampak pada biaya dan memakan waktu di tengah pengembangan. Menurut IEEE, salah satu tujuan utama SRS adalah untuk mengurangi risiko dengan mendefinisikan secara jelas dan tepat kebutuhan sistem [2]. Ini membantu mengidentifikasi dan mengatasi potensi masalah sejak dini.
5) Sebagai pedoman dalam Validasi: SRS memfasilitasi proses validasi [2], di mana kebutuhan perangkat lunak yang telah didefinisikan diuji untuk

memastikan bahwa mereka sesuai dengan kebutuhan pengguna dan pemangku kepentingan

b. Komponen Utama SRS:

Komponen-komponen SRS mempunyai fungsi agar SRS mencakup semua aspek penting yang dibutuhkan untuk pengembangan perangkat lunak yang sesuai dengan kebutuhan pengguna dan spesifikasi yang telah ditetapkan. komponen utama SRS umumnya meliputi Pendahuluan, Deskripsi Umum, Kebutuhan Fungsional, Kebutuhan Non-Fungsional, Antarmuka Sistem, dan Syarat & Ketentuan [1,2,3,4]. Berikut adalah komponen-komponen dari SRS secara detail:

1) Pendahuluan: Menguraikan tujuan, lingkup, definisi, dan referensi yang digunakan dalam dokumen, serta gambaran umum dari sistem yang akan dikembangkan
2) Deskripsi Umum: Menyediakan informasi tentang perspektif produk, fungsi produk, karakteristik pengguna, batasan umum, dan asumsi yang dibuat selama pengembangan.
3) Kebutuhan Fungsional: Mendefinisikan fungsi spesifik yang harus dilakukan oleh perangkat lunak, termasuk interaksi dengan pengguna dan proses internal
4) Kebutuhan Non-Fungsional: Menjelaskan persyaratan yang terkait dengan kualitas sistem, seperti kinerja, keamanan, keandalan, dan skalabilitas
5) Antarmuka Sistem: Menguraikan antarmuka bagaimana sistem akan berinteraksi dengan pengguna, perangkat keras, perangkat lunak lain, dan sistem eksternal
6) Syarat dan Ketentuan (Term of Condition): Menetapkan kriteria yang akan digunakan untuk menentukan apakah perangkat lunak yang dikembangkan memenuhi kebutuhan yang telah ditetapkan.

c. Manfaat SRS

Berikut adalah beberapa manfaat dari Software Requirement Specification (SRS):

1) Panduan yang Jelas untuk Pengembangan Perangkat Lunak: SRS berfungsi sebagai panduan utama yang mendokumentasikan semua kebutuhan dan spesifikasi perangkat lunak. SRS mengurangi ambiguitas dalam interpretasi kebutuhan, sehingga memastikan bahwa semua pihak memahami persyaratan dengan cara yang sama [4], dengan begitu pengembang menggunakan SRS sebagai acuan untuk merancang, mengimplementasikan, dan menguji perangkat lunak, memastikan bahwa sistem yang dibangun sesuai dengan kebutuhan yang telah ditetapkan.
2) Meningkatkan Komunikasi antara Pemangku Kepentingan dengan

pengembang: SRS menjadi alat komunikasi formal antara semua pihak yang terlibat, termasuk pengembang, pengguna, manajer proyek, dan pemangku kepentingan lainnya. Untuk itu dokumen SRS sangat berguna karena semua kebutuhan pengguna dan bisnis telah didokumentasikan dan dipahami dengan jelas [2] sehingga Ini membantu memastikan bahwa semua pihak memiliki pemahaman yang sama tentang apa yang akan dibangun, mengurangi risiko kesalahpahaman

3) Meminimalis Risiko dan Ketidakpastian: Dengan mendefinisikan kebutuhan secara rinci sejak awal, SRS membantu mengidentifikasi potensi masalah dan ketidakpastian sebelum pengembangan dimulai [4]. Ini mengurangi risiko perubahan besar yang bisa berdampak pada biaya dan jadwal proyek.

4) Pedoman untuk Pengujian dan Validasi: SRS menyediakan dasar untuk pengujian perangkat lunak [4] karena SRS menyediakan pedoman yang jelas untuk mengembangkan rencana pengujian. Dengan mendefinisikan kebutuhan dan spesifikasi secara eksplisit, SRS sangat berguna bagi pengembang untuk memastikan bahwa perangkat lunak diuji dan divalidasi sesuai dengan persyaratan yang telah disetujui.

5) Memuat RAB dan Estimasi Proyek: SRS memberikan dasar yang kuat untuk melakukan estimasi biaya, waktu, dan sumber daya yang diperlukan untuk proyek [3] Sehingga SRS membantu manajer proyek dalam perencanaan dan estimasi proyek, termasuk anggaran dan waktu. Dengan mendokumentasikan semua kebutuhan, SRS sangat berguna bagi manajer untuk membuat estimasi yang lebih akurat tentang sumber daya yang dibutuhkan.

6) Meningkatkan Kualitas Perangkat Lunak: Dengan mendefinisikan kebutuhan secara jelas dan terstruktur, SRS membantu meningkatkan kualitas perangkat lunak yang dihasilkan. Ini memastikan bahwa perangkat lunak memenuhi semua kebutuhan pengguna dan standar kualitas yang diinginkan, yang pada akhirnya meningkatkan kepuasan pengguna dan keberhasilan proyek [5].

7) Menyediakan aturan dan pedoman Pemeliharaan Perangkat Lunak: SRS menjadi dokumen referensi penting selama fase pemeliharaan perangkat lunak. Jika suatu saat terjadi perubahan atau perbaikan, maka SRS menyediakan gambaran lengkap tentang kebutuhan asli perangkat lunak [2], sehingga memudahkan tim pemeliharaan untuk melakukan perubahan maupun perbaikan yang sesuai aturan

8) Memudahkan Pengelolaan Revisi: SRS memberikan pedoman dalam pengelolaan manajemen revisi atau perubahan yang lebih efektif, sehingga

Dengan adanya SRS, perubahan kebutuhan dapat dikelola dengan lebih baik [1]. Dengan memiliki dokumen yang mendefinisikan kebutuhan perangkat lunak secara rinci, setiap revisi dapat dievaluasi secara sistematis untuk dampaknya terhadap keseluruhan proyek.

9) Meningkatkan Kepuasan Pengguna: Dengan mendokumentasikan kebutuhan pengguna secara rinci dalam SRS, pengembang dapat memastikan bahwa perangkat lunak yang dihasilkan sesuai dengan harapan pengguna. Hal ini meningkatkan peluang bahwa perangkat lunak akan memenuhi kebutuhan dan meningkatkan kepuasan pengguna [5].

2. VALIDASI KEBUTUHAN - SOFTWARE REQUIREMENTS VALIDATION

Software Requirement Validation adalah proses penting dalam pengembangan perangkat lunak yang memastikan bahwa kebutuhan yang telah diidentifikasi dan didokumentasikan sesuai dengan keinginan dan kebutuhan pengguna serta dapat diimplementasikan dengan sukses. Validasi kebutuhan dilakukan untuk menghindari kesalahan yang dapat menyebabkan proyek gagal memenuhi tujuan bisnisnya. Pressman mendefinisikan validasi kebutuhan adalah proses evaluasi kebutuhan perangkat lunak untuk memastikan bahwa kebutuhan tersebut lengkap, konsisten, dan benar sesuai dengan harapan pemangku kepentingan dan tujuan bisnis [4], selaras dengan pressman yang mana Sommerville menyatakan bahwa validasi kebutuhan berfokus pada memastikan bahwa kebutuhan yang didefinisikan benar-benar mencerminkan sistem yang diinginkan oleh pelanggan [2], sehingga validasi kebutuhan melibatkan pengecekan apakah kebutuhan tersebut benar, akan memenuhi kebutuhan pemangku kepentingan, dan dapat dicapai dalam batasan sistem yang ada, dalam dokumen Standar IEEE 830-1998 juga menyatakan bahwa Validasi memastikan bahwa dokumen kebutuhan perangkat lunak mencerminkan kebutuhan dan harapan aktual dari pemangku kepentingan dan tidak mengandung kesalahan atau kelalaian, maka dari itu Validasi kebutuhan sangat penting untuk memastikan bahwa perangkat lunak yang dikembangkan tidak hanya memenuhi spesifikasi teknis tetapi juga tujuan bisnis dan harapan pengguna.

Pada bahasan Software Requirement Validation (SRV) akan membahas mengenai tujuan SRV, dan Metode SRV

a. Tujuan Validasi Kebutuhan Perangkat Lunak

Tujuan dari validasi kebutuhan perangkat lunak adalah untuk memastikan bahwa produk akhir benar-benar memenuhi kebutuhan dan harapan pengguna serta pemangku kepentingan. Berikut adalah beberapa tujuan utama dari validasi kebutuhan perangkat lunak:

1) Memastikan Kesesuaian dengan Kebutuhan Pengguna dan Pemangku

Kepentingan: Validasi memastikan bahwa perangkat lunak yang dikembangkan akan memberikan solusi yang sesuai dengan harapan dan kebutuhan pengguna, baik dari segi fungsi maupun performa, artinya validasi kebutuhan adalah langkah untuk memastikan bahwa produk yang dikembangkan akan memenuhi ekspektasi dan tujuan bisnis [2].

2) Mengidentifikasi dan Mengurangi Kesalahan Sejak Awal: Validasi kebutuhan dapat mengidentifikasi kesalahan, ambiguitas, atau ketidaklengkapan dalam dokumen kebutuhan sebelum pengembangan perangkat lunak dimulai. Pfleeger dan Atlee menjelaskan bahwa salah satu tujuan validasi adalah mengurangi ketidakpastian dan ambiguitas dalam kebutuhan [6], sehingga validasi kebutuhan membantu memastikan bahwa semua pihak memiliki pemahaman yang sama tentang apa yang harus dikembangkan, sehingga mengurangi risiko misinterpretasi yang bisa mempengaruhi hasil akhir proyek.

3) Meningkatkan Kualitas Perangkat Lunak: Dengan memastikan bahwa semua kebutuhan telah diuji dan diverifikasi, validasi membantu dalam pengembangan perangkat lunak yang lebih stabil, handal, dan sesuai dengan spesifikasi. Wiegers dan Beatty memberikan penjelasan bahwa validasi kebutuhan bertujuan untuk meningkatkan kualitas dokumen kebutuhan, memastikan bahwa dokumen kebutuhan berisi konsisten, lengkap, dan bebas dari ambiguitas [3], sehingga ini membantu memastikan bahwa perangkat lunak yang dikembangkan berdasarkan kebutuhan ini akan memiliki kualitas yang lebih tinggi dan lebih dapat diandalkan.

4) Mendukung Pengambilan Keputusan: Validasi kebutuhan membantu manajer proyek dalam memahami mana kebutuhan yang paling kritis dan harus diprioritaskan, serta bagaimana kebutuhan tersebut mempengaruhi seluruh siklus hidup proyek. Boehm dan Turner menekankan bahwa validasi kebutuhan membantu tim pengembangan dalam membuat keputusan yang lebih baik tentang prioritas dan alokasi sumber daya [5]. Dengan kebutuhan yang sudah divalidasi, tim dapat memastikan bahwa mereka fokus pada fitur dan fungsi yang paling penting untuk pemangku kepentingan.

5) Memastikan Kesesuaian dengan Tujuan Bisnis: Validasi membantu memastikan bahwa perangkat lunak yang dikembangkan akan mendukung pencapaian tujuan bisnis, termasuk efisiensi operasional, peningkatan layanan pelanggan, atau inovasi produk. Thayer dan Dorfman menjelaskan bahwa validasi kebutuhan bertujuan untuk memastikan bahwa kebutuhan perangkat lunak tidak hanya sesuai dengan kebutuhan pengguna, tetapi juga sejalan dengan tujuan bisnis dan teknis organisasi [10], dengan begitu validasi

kebutuhan dapat memastikan bahwa perangkat lunak yang dikembangkan akan memberikan nilai maksimal bagi pemangku kepentingan.

6) Mendukung Komunikasi dan Kolaborasi: Validasi kebutuhan memerlukan dialog yang mendalam dan kolaboratif antara semua pihak yang terlibat, sehingga membantu memastikan bahwa semua orang memiliki pemahaman yang sama tentang apa yang akan dikembangkan. Leffingwell dan Widrig menyebutkan bahwa validasi kebutuhan bertujuan untuk memfasilitasi komunikasi yang lebih efektif antara pengembang, pemangku kepentingan, dan pengguna [9], dengan begitu dengan validasi kebutuhan maka kebutuhan dalam pengembangan perangkat lunak bisa dipahami secara jelas dan konsisten oleh semua pihak yang terlibat dalam proyek.

7) Menyiapkan Pedoman Pengujian Perangkat Lunak: Validasi kebutuhan memastikan bahwa semua kebutuhan dapat diuji dan divalidasi melalui pengujian, membantu tim pengembangan untuk merancang uji kasus yang tepat yang memastikan perangkat lunak berfungsi sesuai dengan kebutuhan. Humphrey menunjukkan bahwa validasi kebutuhan bertujuan untuk memastikan bahwa setiap kebutuhan dapat diuji dan diverifikasi [8] . Ini penting untuk memastikan bahwa kebutuhan tersebut dapat dipenuhi dan diuji dalam produk akhir, yang akan memfasilitasi proses pengujian dan kualitas perangkat lunak.

b. Metode Validasi Kebutuhan Perangkat lunak

Berikut adalah beberapa metode umum yang digunakan untuk validasi kebutuhan perangkat lunak:

1) Inspeksi (Inspection): Inspeksi adalah proses sistematis dengan melakukan pemeriksaan manual terhadap dokumen kebutuhan untuk menemukan kesalahan, inkonsistensi, atau ketidakjelasan. Sommerville menekankan pentingnya inspeksi sebagai cara untuk meningkatkan kualitas dokumen kebutuhan melalui evaluasi oleh rekan sejawat (peer review) [2]. Untuk itu Inspeksi dianggap sebagai salah satu metode paling efektif untuk menemukan masalah sejak awal.

2) Prototyping: Prototyping merupakan metode validasi kebutuhan dengan melakukan pembuatan model awal atau prototipe perangkat lunak yang mewakili beberapa aspek dari sistem yang diusulkan, Pressman dan Maxim menjelaskan bahwa prototyping sangat berguna untuk mendapatkan tanggapan langsung dari pengguna dan membantu dalam mengklarifikasi dan mengkonfirmasi kebutuhan yang kompleks atau ambigu [4], sehingga metode Prototyping memiliki tujuan untuk memvalidasi kebutuhan dengan memberikan model awal yang konkret kepada pengguna untuk diujicoba dan

dievaluasi.

3) Review Formal (Formal Review): Metode ini merupakan metode dengan melakukan tinjauan terstruktur terhadap dokumen kebutuhan oleh tim pengembang yang mencakup analis, pengguna, pengembang, dan pemangku kepentingan lainnya. Wiegers dan Beatty memberikan pandangan bahwa metode review formal menghasilkan dokumentasi tertulis dari hasil tinjauan dan memberikan kesepakatan di antara para pemangku kepentingan tentang kebutuhan yang telah ditetapkan [3], sehingga tujuan dari metode review formal adalah memastikan bahwa kebutuhan telah ditentukan dengan benar, lengkap, dan konsisten dengan tujuan sistem.

4) Walkthrough: Walkthrough adalah proses dimana analis sistem melakukan diskusi dan kolaborasi yang lebih mendalam terhadap dokumen kebutuhan, menjelaskan setiap bagian secara rinci kepada pemangku kepentingan. Seperti yang dijelaskan oleh Pfleeger dan Atlee bahwa walkthrough sebagai metode yang sangat berguna untuk mengidentifikasi kesalahan dan ketidakjelasan melalui diskusi kelompok yang terbuka [6]. Tujuan dari metode ini adalah Mengidentifikasi kesalahan, ketidakjelasan, atau masalah dalam dokumen kebutuhan melalui diskusi kelompok

5) Simulasi: Simulasi adalah metode validasi dengan menggunakan alat simulasi untuk mensimulasikan perilaku sistem berdasarkan spesifikasi kebutuhan yang ada. Boehm dan Turner menjelaskan bahwa simulasi dapat membantu memvisualisasikan bagaimana sistem akan berfungsi di bawah berbagai kondisi, membantu dalam memastikan bahwa kebutuhan sudah lengkap dan akurat sebelum implementasi dimulai [7]. Tujuan dari metode simulasi adalah Menguji bagaimana sistem akan bekerja di skenario yang berbeda, membantu dalam memahami apakah kebutuhan yang ada sudah tepat

6) Case study: Case study merupakan metode yang membuat uji kasus berdasarkan kebutuhan yang telah ditentukan dan menjalankan pengujian untuk memvalidasi bahwa kebutuhan tersebut telah terpenuhi. Humphrey menekankan bahwa pengujian berdasarkan uji kasus yang berasal dari dokumen SRS (Software Requirements Specification) adalah cara yang sangat efektif untuk memastikan bahwa perangkat lunak sesuai dengan kebutuhan yang telah disepakati [8] . Metode ini memiliki tujuan untuk memastikan bahwa setiap kebutuhan dapat diuji dan diverifikasi melalui pengujian.

7) Interview dan Questionnaires: Interviews dan Questionnaires merupakan metode validasi dengan melakukan pengumpulan tanggapan langsung dari

pengguna dan pemangku kepentingan melalui wawancara atau kuesioner. Leffingwell dan Widrig mencatat bahwa wawancara dan kuesioner adalah metode yang berguna untuk mendapatkan wawasan langsung dari pengguna, terutama ketika kebutuhan perangkat lunak sangat tergantung pada preferensi dan pengalaman pengguna akhir [9]. Maka dari itu tujuan dari metode ini adalah mendapatkan pandangan dan validasi langsung dari pemangku kepentingan mengenai apakah kebutuhan yang telah ditetapkan sesuai dengan harapan mereka.

8) Traceability Matrix: Traceability Matriks berfungsi untuk menghubungkan setiap kebutuhan dengan penggunan dan dengan pengujian yang akan memverifikasinya. Thayer dan Dorfman menekankan bahwa traceability matrix adalah alat yang penting dalam mengelola perubahan kebutuhan selama proyek berlangsunsg, memastikan bahwa semua kebutuhan telah dipenuhi dan diuji dengan benar [10]. Sehingga tujuan dari metode Traceability Matriks adalah memastikan bahwa semua kebutuhan telah diperiksa dan dilacak sepanjang siklus hidup pengembangan perangkat lunak.

3. DAFTAR REFERENSI

1. IEEE. IEEE Standard 830-1998 - Recommended Practice for Software Requirements Specifications. ISBN:978-0-7381-0448-5. Reterived at 08 August 2024 from https://ieeexplore.ieee.org/document/720574
2. Sommerville, Ian. (2015). Software Engineering 10th Edition. Pearson Education, Inc.. ISBN-13: 978-0-13-703515-1.
3. Karl Wiegers dan Joy Beatty. (2013). Software Requirements (Developer Best Practices) 3rd Edition. Microsoft Press. ISBN-10: 0735679665
4. Roger S. Pressman S. R, Maxim. B. (2019). Software Engineering: A Practitioner's Approach 9th Edition. McGraw Hill. ISBN 9780078022128.
5. Boehm, B.W. (2001). Software Engineering Economics. In: Broy, M., Denert, E. (eds) Pioneers and Their Contributions to Software Engineering. Springer, Berlin, Heidelberg. ISBN: 978-3-642-48354-7.
6. Pfleeger, S. L., & Atlee, J. M. (2010). Software Engineering: Theory and Practice. Pearson. ISBN-10: 0136061699
7. Boehm, B., & Turner, R. (2004). Balancing Agility and Discipline: A Guide for the Perplexed In: Ramamoorthy, C.V., Lee, R., Lee, K.W. (eds) Software Engineering Research and Applications. SERA 2003. Lecture Notes in Computer Science, vol

3026. Springer, Berlin, Heidelberg. ISBN: 978-3-540-21975-0

8. Humphrey, W. S. (1995). A Discipline for Software Engineering. Addison-Wesley Professional. ISBN-10: 1572485965
9. Leffingwell, D., & Widrig, D. (2003). Managing Software Requirements: A Use Case Approach. Addison-Wesley Professional. ISBN-10: 032112247X
10. Thayer, R. H., & Dorfman, M. (2000). Software Requirements Engineering. Wiley-IEEE Computer Society Pr . ISBN-10: 0818677384
11.
12. Kenneth E. Kendall dan Julie E. Kendall (2014). Systems Analysis and Design. Pearson. ISBN-10: 013478555X. 2014
13. Roger S. Pressman S. R, Maxim. B. (2019). Software Engineering: A Practitioner's Approach 9th Edition. McGraw Hill. ISBN 9780078022128.
14. Klaus Pohl (2010). Requirements Engineering: Fundamentals, Principles, and Techniques. Springer. ISBN-10: 3642125778.
15. Suzanne Robertson & James Robertson (2013). Mastering the Requirements Process: Getting Requirements Right. Addison-Wesley Professional. ISSN-10: 0321815742
16. Steve McConnell. (1996). Rapid Development: Taming Wild Software Schedules. Microsoft Press. ISBN-10: 9781556159008
17. Daniel R. Windle. (2002). Software Requirements Using the Unified Process: A Practical Approach. Prentice Hall. ISBN-10: 0130969729.
18. Gunnar Overgaard & Karin Palmkvist. (2005). Applying Use Cases: A Practical Guide. Addison-Wesley Professional. ISBN-10: 0201708531.
19. Dean Leffingwell & Don Widrig. (2003). Managing Software Requirements: A Unified Approach. Addison-Wesley. ISBN-10: 0201615932
20. Benjamin M. Brothers (2011). The Art of Software Modeling. Auerbach Publications. ISBN-10: 1420044621

4. Daftar Bacaan

1. Sama seperti pada daftar referensi

5. 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	Idem

	b. Keamanan data akses	
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**	