UNIT TESTING, INTEGRATION TESTING, SYSTEM TESTING, ACCEPTANCE TESTING DALAM PERANCANGAN PERANGKAT LUNAK

Mata Kuliah: Software Engineering

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

Pertemuan ke-

Topik Bahasan ke-

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**

Dalam pengembangan perangkat lunak ada berbagai tingkatan pengujian yang dirancang untuk mengevaluasi sistem perangkat lunak, diantaranya adalah unit testing, integration testing, system testing dan acceptance testing. Unit testing adalah tahap pengujian yang berfokus pada komponen terkecil, seperti fungsi atau metode, untuk memastikan bahwa setiap unit berfungsi sesuai spesifikasinya. Setelah itu, integration testing menguji interaksi antar-unit ketika digabungkan untuk memastikan integrasi yang lancar tanpa konflik. System testing menguji perangkat lunak secara keseluruhan dalam lingkungan yang menyerupai kondisi nyata guna memverifikasi bahwa sistem berfungsi sesuai persyaratan. Akhirnya, acceptance testing dilakukan oleh pengguna akhir atau klien untuk memastikan bahwa perangkat lunak memenuhi kebutuhan mereka dan siap digunakan. Keempat tahap pengujian ini membantu menjamin perangkat lunak berkualitas tinggi, fungsional, dan siap dirilis.

1. UNIT TESTING

Unit Testing dalam pengembangan perangkat lunak adalah metode pengujian yang menguji komponen terkecil dari perangkat lunak, biasanya berupa fungsi atau metode, secara terisolasi untuk memastikan bahwa bagian tersebut berfungsi sesuai dengan spesifikasinya. Unit testing merupakan langkah awal dalam proses pengujian karena membantu mengidentifikasi **faults** sejak awal, yang mengurangi kemungkinan terjadinya **errors** dan **failures** di tahap yang lebih lanjut dalam siklus pengembangan.

1) Tujuan Unit Testing

beberapa tujuan utama unit testing adalah

a) **Memastikan Fungsionalitas Dasar**: Unit testing memastikan bahwa setiap unit pada perangkat lunak, seperti fungsi atau metode, bekerja sesuai persyaratan.
b) **Deteksi Dini Kesalahan**: Dengan memvalidasi kode di level unit, pengembang dapat menemukan kesalahan lebih awal dalam proses pengembangan, yang lebih mudah dan lebih murah untuk diperbaiki daripada kesalahan di tahap akhir.
c) **Meningkatkan Kualitas Kode**: Unit testing mendorong pengembangan

kode yang modular, karena kode modular lebih mudah diuji dan dipelihara

2) Proses Unit Testing

beberapa proses utama unit testing adalah

a) **Menulis Test Case:** Pengembang membuat berbagai skenario uji untuk unit yang akan diuji, termasuk skenario normal dan skenario batas untuk menguji respons unit terhadap input yang berbeda.
b) **Menjalankan Test Case**: Unit diuji dengan menjalankan setiap kasus uji yang dirancang untuk melihat apakah hasil aktual sesuai dengan hasil yang diharapkan.
c) **Evaluasi Hasil:** Jika hasil sesuai dengan ekspektasi, unit dianggap berfungsi dengan benar. Jika tidak, pengembang mengidentifikasi dan memperbaiki kesalahan sampai unit lulus uji.
d) **Pengulangan**: Unit tests dilakukan setiap kali kode mengalami perubahan untuk memastikan bahwa perubahan tersebut tidak merusak fungsionalitas yang ada (regression testing)

3) Contoh Penerapan Unit Testing

Misalnya, dalam aplikasi spada, fungsi yang menghitung total nilai setelah setiap soal diuji untuk berbagai skenario penilaian. Pengujian ini akan memastikan bahwa fungsi tersebut memberikan hasil yang benar untuk semua skenario yang masuk akal.

4) Contoh Alat untuk Unit Testing

beberapa contoh alat untuk unit testing adalah

a) **JUnit** untuk Java
b) **pytest** untuk Python
c) **NUnit** untuk C#
d) **Mocha** untuk JavaScript

2. INTEGRATION TESTING

Integration Testing adalah tahap pengujian perangkat lunak yang fokus pada pengujian interaksi antar-komponen atau modul yang telah diuji secara mandiri (unit testing) untuk memastikan mereka bekerja bersama dengan benar. Pada tahap ini, integration testing memastikan bahwa berbagai modul dalam sistem dapat berkomunikasi dan berfungsi bersama sesuai dengan spesifikasi. Pengujian ini penting untuk mendeteksi masalah yang mungkin muncul saat modul atau unit digabungkan, termasuk konflik data, kesalahan antarmuka, atau kesalahan logika antar-modul..

1) Tujuan Integration Testing

beberapa tujuan integration testing adalah

a) **Mendeteksi Masalah Interaksi**: Integration testing berguna bagi pengembang untuk mendeteksi masalah interaksi antar-modul yang tidak terlihat pada unit testing.
b) **Meningkatkan Keandalan Sistem**: Dengan menguji semua jalur komunikasi antara modul, integration testing membantu memastikan sistem bekerja sebagai satu kesatuan yang andal.
c) **Mempersiapan untuk System Testing**: Integration testing memvalidasi bahwa sistem berfungsi di tingkat modul gabungan, sehingga sistem siap diuji secara keseluruhan pada tahap berikutnya (system testing).

2) Proses Integration Testing

beberapa pendekatan dalam proses Integration testing adalah

a) **Big Bang Integration**: Semua modul digabungkan sekaligus, dan sistem diuji secara keseluruhan. Pendekatan ini memungkinkan pengujian menyeluruh, tetapi sulit untuk mengidentifikasi sumber masalah jika banyak kesalahan muncul setelah integrasi.
b) **Incremental Integration**: Modul-modul diintegrasikan dan diuji secara bertahap. Pendekatan ini dilakukan secara top-down, bottom-up, atau kombinasi keduanya (sandwich testing), sehingga masalah dapat diatasi sebelum lebih banyak modul ditambahkan ke dalam sistem.
c) **Top-Down dan Bottom-Up Integration**: Top-down integration menguji modul dari level atas ke bawah, sementara bottom-up integration menguji dari level bawah ke atas, memudahkan penemuan kesalahan di masing-masing level hierarki modul..

3) Contoh Penerapan Integration Testing

Dalam sistem spada, modul "akun" perlu terhubung dengan modul "nilai" dan "mata kuliah". Integration testing memastikan bahwa proses nilai dari mata kuliah ke akun berjalan lancar, menguji bahwa informasi nilai terupdate di seluruh modul tanpa error.

4) Contoh Alat untuk Integration Testing

beberapa contoh alat untuk Integration Testing adalah

a) **JUnit**: JUnit adalah framework pengujian untuk aplikasi berbasis Java
b) **Postman**: Postman adalah alat pengujian integrasi layanan web dan RESTful API untuk memastikan bahwa modul backend saling berkomunikasi dengan benar
c) **Selenium**: Selenium adalah alat otomatisasi pengujian berbasis web yang populer untuk pengujian antarmuka pengguna.

d) **pache JMeter**: JMeter adalah alat pengujian untuk memastikan bahwa integrasi antara server, API, dan modul backend berfungsi dengan benar

e) **SoapUI**: SoapUI adalah alat pengujian khusus untuk layanan web SOAP dan REST

f) **Pytest**: Pytest adalah framework pengujian untuk Python yang dapat digunakan untuk unit testing dan integration testing.

g) **Telerik Test Studio**: Tools untuk mengotomatisasi pengujian aplikasi berbasis web, desktop, dan seluler, mendukung pengujian integrasi di berbagai environment

3. SYSTEM TESTING

System Testing adalah tahap pengujian perangkat lunak yang fokus pada evaluasi sistem secara menyeluruh untuk memastikan bahwa perangkat lunak berfungsi sesuai dengan persyaratan yang telah ditetapkan. Pada tahap ini, sistem diuji dalam kondisi yang mendekati penggunaan nyata, sehingga semua komponen yang telah terintegrasi diuji bersama-sama untuk mengevaluasi fungsionalitas, performa, keamanan, dan keandalan.

1) Tujuan System Testing

beberapa tujuan utama system testing adalah

a) **Mendeteksi Masalah Lebih Awal**: System testing mendeteksi masalah sebelum perangkat lunak diimplementasikan, sehingga pengembang dapat mengatasi kesalahan lebih awal dan mengurangi risiko kesalahan di lingkungan produksi.

b) **Memastikan Kepatuhan Spesifikasi**: Pengujian ini membantu memastikan bahwa semua persyaratan teknis dan bisnis telah terpenuhi, memberikan keyakinan bahwa perangkat lunak siap digunakan oleh pengguna akhir.

c) **Menjamin Kualitas dan Stabilitas**: Dengan menguji perangkat lunak sebagai kesatuan, system testing membantu memastikan stabilitas sistem secara keseluruhan, terutama di bawah kondisi penggunaan nyata.

2) Proses System Testing

beberapa pendekatan dalam proses System testing adalah

a) **Functional Testing**: Menguji setiap fitur sistem untuk memastikan bahwa perangkat lunak berfungsi sesuai dengan yang diharapkan. Ini mencakup pengujian alur kerja, antarmuka pengguna, dan interaksi antar-komponen.

b) **Performance Testing**: Mengukur kinerja sistem di bawah beban kerja tertentu untuk memastikan bahwa perangkat lunak memenuhi standar kecepatan, responsivitas, dan stabilitas.

c) **Security Testing**: Memastikan bahwa perangkat lunak terlindungi dari ancaman keamanan, seperti akses tidak sah dan serangan eksternal, dengan menguji sistem dari sudut pandang keamanan

d) **Usability Testing**: Mengukur kemudahan penggunaan perangkat lunak dari sudut pandang pengguna akhir. Ini bertujuan untuk memastikan bahwa perangkat lunak intuitif dan mudah digunakan

e) **Compatibility Testing**: Menguji kompatibilitas perangkat lunak pada berbagai perangkat, sistem operasi, dan lingkungan jaringan yang berbeda

3) Contoh Penerapan System Testing

Pada aplikasi Spada, system testing dapat mencakup pengujian berbagai fitur, seperti login, mendaftar matakuliah, pengumpulan tugas, dan penilaian ujian. Semua fungsi ini diuji untuk memastikan bahwa aplikasi bekerja dengan baik saat digunakan di berbagai perangkat, dan dalam skenario yang mungkin dihadapi pengguna.

4) Contoh Alat untuk System Testing

beberapa contoh alat dalam proses System testing adalah

a) **Selenium**: Alat open-source yang populer untuk mengotomatisasi pengujian aplikasi web. Selenium mendukung berbagai browser dan memungkinkan pengujian interaksi antarmuka pengguna, membuatnya ideal untuk pengujian sistem pada aplikasi berbasis web.

b) **Apache JMeter**: Alat pengujian kinerja yang sering digunakan untuk menguji seberapa baik aplikasi berfungsi di bawah beban tertentu

c) **Postman**: Digunakan untuk pengujian API, Postman memungkinkan pengujian interaksi antar-layanan dan endpoint dalam sistem.

d) **SoapUI**: Alat pengujian untuk layanan web yang mendukung protokol SOAP dan REST, memungkinkan pengujian terhadap layanan dan API

e) **Appium**: Alat pengujian open-source untuk aplikasi seluler. Appium mendukung platform Android dan iOS dan memungkinkan pengujian aplikasi mobile pada berbagai perangkat, ideal untuk menguji fungsionalitas sistem pada platform seluler

4. ACCEPTANCE TESTING

Acceptance Testing adalah tahap terakhir dalam pengujian perangkat lunak yang berfokus pada verifikasi bahwa perangkat lunak telah memenuhi semua persyaratan yang disepakati dan siap untuk digunakan oleh pengguna akhir. Tahap ini bertujuan untuk memastikan bahwa perangkat lunak dapat memenuhi kebutuhan bisnis dan spesifikasi pengguna, serta bekerja dengan baik di lingkungan nyata. Acceptance

testing sering kali dilakukan oleh pengguna akhir atau tim yang berperan sebagai pengguna untuk memastikan bahwa perangkat lunak berfungsi sesuai harapan mereka.

1) Tujuan Acceptance Testing

   beberapa tujuan utama acceptance testing adalah

   a) **Memvalidasi Kepuasan Pengguna**: Acceptance testing memastikan bahwa perangkat lunak benar-benar sesuai dengan harapan pengguna, membantu meningkatkan kepuasan dan kepercayaan pengguna.
   b) **Mengurangi Risiko Produksi**: Dengan pengujian akhir ini, tim dapat mengidentifikasi dan memperbaiki masalah sebelum perangkat lunak dirilis ke produksi, mengurangi risiko masalah besar di lingkungan nyata.
   c) **Memastikan Kepatuhan Bisnis dan Regulasi**: Acceptance testing membantu memastikan bahwa sistem telah memenuhi semua persyaratan bisnis dan peraturan, yang penting untuk menghindari kesalahan yang dapat berdampak hukum atau finansial.

2) Proses Acceptance Testing

   beberapa pendekatan dalam proses Acceptance testing adalah

   a) **User Acceptance Testing (UAT)**: Pengguna akhir atau perwakilan bisnis menguji perangkat lunak untuk memverifikasi bahwa sistem memenuhi persyaratan dan dapat digunakan sesuai kebutuhan. UAT sering dilakukan dalam lingkungan yang mirip dengan produksi dan mencakup alur kerja bisnis utama.
   b) **Operational Acceptance Testing (OAT)**: Pengujian ini fokus pada kesiapan operasional, seperti cadangan data, manajemen akses, dan prosedur pemulihan. OAT memastikan bahwa perangkat lunak dapat dioperasikan dengan andal di lingkungan nyata dan siap untuk dikelola oleh tim IT.
   c) **Regulatory Acceptance Testing**: Untuk perangkat lunak yang tunduk pada regulasi tertentu (misalnya, di sektor keuangan atau kesehatan), regulatory acceptance testing memastikan bahwa sistem memenuhi semua standar dan peraturan hukum yang berlaku
   d) **Contract Acceptance Testing**: Pengujian ini dilakukan berdasarkan kriteria yang telah ditentukan dalam kontrak atau perjanjian proyek antara pengembang dan klien. Jika perangkat lunak memenuhi spesifikasi kontrak, maka pengujian ini dinyatakan lulus
   e) **Beta Testing**: Pengujian dilakukan oleh sekelompok pengguna akhir yang sebenarnya, yang dikenal sebagai penguji beta

3) Contoh Penerapan Acceptance Testing

Dalam aplikasi Spada, acceptance testing akan mencakup skenario seperti pencarian mata kuliah, registrasi mata kuliah, pengumpulan tugas dan ujian di mata kuliah, dan mendapatkan nilai. Pengguna akhir memastikan setiap langkah dalam proses transaksi berfungsi dengan baik dan memberikan hasil yang diharapkan.

4) Contoh Alat untuk Acceptance Testing

Acceptance testing dilakukan dengan R&D dari aplikasi tersebut dengan metode-metode pendekatan dalam proses Acceptance testing.

a) **UAT** memastikan perangkat lunak sesuai dengan kebutuhan pengguna
b) **OAT** memastikan perangkat lunak siap secara operasional.
c) **Compliance testing** menghindari masalah hukum atau regulasi
d) **Regulatory Acceptance testing** memvalidasi kepatuhan dengan kontrak proyek.
e) **Beta testing** membantu menemukan dan memperbaiki bug dari perspektif pengguna sebelum peluncuran resmi

5. KESIMPULAN

Perbandingan Faults, Errors, dan Failures

Jenis Pengujian	**Deskripsi**	**Tujuan Utama**	**Dilakukan Oleh**
Unit Testing	Menguji komponen terkecil (modul/fungsi) perangkat lunak secara individual	Memastikan setiap komponen berfungsi sesuai spesifikasi.	Pengembang
Integration Testing	Menguji interaksi antara beberapa modul	Memastikan modul bekerja dengan baik saat digabungkan.	Pengembang atau tim QA
System Testing	Menguji perangkat lunak secara keseluruhan dalam lingkungan yang menyerupai kondisi sebenarnya	Memastikan sistem bekerja sesuai persyaratan	Tim QA
Acceptance Testing	Menguji perangkat lunak berdasarkan kebutuhan pengguna akhir dan persyaratan bisnis	Memastikan perangkat lunak siap digunakan oleh pengguna.	Pengguna atau klien

6. DAFTAR REFERENSI

1. ISO/IEC/IEEE 29119: *Software Testing* - Panduan Standar Internasional untuk praktik terbaik dalam pengujian perangkat lunak
2. IEEE Std. 829-2008: *Standard for Software and System Test Documentation* - Panduan standar untuk dokumentasi dalam pengujian perangkat lunak dan sistem
3. Myers, G. J., Sandler, C., & Badgett, T. (2011). *The Art of Software Testing*. Wiley.
4. ISTQB Foundation Level Syllabus, 2018
5. Kaner, Falk, & Nguyen, 1999, *Testing Computer Software*
6. *Nielsen, J. (1994). Usability Engineering. Academic Press*
7. *ISO 9241-11:2018 - Ergonomics of human-system interaction - Part 11: Usability: Definitions and concepts.*
8. *Rubin, J. (2012). Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. Wiley*
9. *Kohavi, R., Longbotham, R., & Tang, D. (2009). Testing for Usability in Online Systems. ACM Conference on Human Factors in Computing Systems.*

7. Daftar Bacaan

1. Sama seperti pada daftar referensi

8. 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	Idem

	c. Spesifikasi dan validasi kebutuhan	
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 dalam User Interface b. Prinsip Desain Antarmuka (user experience, user guidance, user diversity)	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	Pengujian unit, Pengujian integrasi, Pengujian sistem, Pengujian Penerimaan	Idem
13	a. Quality assurance pada perangkat lunak b. Keamanan data akses	Idem
14	Definisi pemeliharaan perangkat lunak dan Konsep Pemeliharaan Perangkat lunak	Idem
15	Teknik pemeliharaan perangkat lunak (Pemeliharaan korektif, pemeliharaan adaptif, pemeliharaan perfektif, pemeliharaan preventif)	Idem
16	**UAS**	