# Penguji Bersertifikat

# Silabus Tingkat Dasar - Silabus

Versi 2018 v3.1.1

International Software Testing Qualifications Board

# Pemberitahuan Hak Cipta

# Riwayat Revisi

| Versi | Tanggal | Catatan |
|---|---|---|
| ISTQB® 2018 v3.1.1 | 1-Juli-2021 | Pembaruan Hak Cipta dan Catatano |
| ISTQB® 2018 v3.1 | 11-November-2019 | Rilis Pemeliharaan Silabus Penguji Bersertifikat Tingkat Dasar dengan pembaruan kecil – lihat Catatan Rilis |
| ISTQB® 2018 | 27-April-2018 | Versi rilis umum kandidat |
| ISTQB® 2011 | 1-April-2011 | Rilis Pemeliharaan Silabus Penguji Bersertifikat Tingkat Dasar – lihat Catatan Rilis |
| ISTQB® 2010 | 30-Mar-2010 | Rilis Pemeliharaan Silabus Penguji Bersertifikat Tingkat Dasar – lihat Catatan Rilis |
| ISTQB® 2007 | 01-Mei-2007 | Rilis Pemeliharaan Silabus Penguji Bersertifikat Tingkat Dasar |
| ISTQB® 2005 | 01-Juli-2005 | Silabus Penguji Bersertifikat Tingkat Dasar |
| ASQF V2.2 | Juli-2003 | Silabus ASQF Tingkat Dasar Versi 2.2 “Lehrplan Grundlagen des Software-testens“ |
| ISEB V2.0 | 25-Feb-1999 | Silabus Dasar Pengujian Perangkat Lunak ISEB V2.0 |

# Daftar Isi

# Ucapan Terima Kasih

Dokumen ini secara resmi dirilis oleh Majelis Umum ISTQB® (11 November 2019).

Itu diproduksi oleh tim dari Dewan Kualifikasi Pengujian Perangkat Lunak Internasional: Klaus Olsen (ketua), Meile Posthuma dan Stephanie Ulrich.

Tim mengucapkan terima kasih kepada Dewan Anggota atas komentar tinjauan mereka terkait dengan Silabus Dasar 2018.

Silabus Dasar 2018 diproduksi oleh tim dari Badan Kualifikasi Pengujian Perangkat Lunak Internasional: Klaus Olsen (ketua), Tauhida Parveen (wakil ketua), Rex Black (manajer proyek), Debra Friedenberg, Judy McKay, Meile Posthuma, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh, dan Eshraka Zakaria.

Tim berterima kasih kepada Rex Black dan Dorothy Graham atas penyuntingan teknis mereka, dan tim peninjau, tim peninjau silang, dan Dewan Anggota atas saran dan masukan mereka.

Orang-orang berikut berpartisipasi dalam meninjau, mengomentari, dan memilih silabus ini: Tom Adams, Tobias Ahlgren, Xu Aiguo, Chris Van Bael, Katalin Balla, Graham Bath, Gualtiero Bazzana, Arne Becher, Veronica Belcher, Lars Hilmar Bjørstrup, Ralf Bongard, Armin Born, Robert Bornelind, Mette Bruhn-Pedersen, Geza Bujdoso, Earl Burba, Filipe Carlos, Young Jae Choi, Greg Collina, Alessandro Collino, Cui Zhe, Taz Daughtrey, Matthias Daigl, Wim Decoutere, Frans Dijkman, Klaudia Dussa-Zieger, Yonit Elbaz, Ofer Feldman, Mark Fewster, Florian Fieber, David Frei, Debra Friedenberg, Conrad Fujimoto, Pooja Gautam, Thorsten Geiselhart, Chen Geng, Christian Alexander Graf, Dorothy Graham, Michel Grandjean, Richard Green, Attila Gyuri, Jon Hagar, Kobi Halperin, Matthias Hamburg, Zsolt Hargitai, Satoshi Hasegawa, Berit Hatten, Wang Hongwei, Tamas Horváth, Leanne Howard, Chinthaka Indikadahena, J. Jayapradeep, Kari Kakkonen, Gábor Kapros, Beata Karpinska, Karl Kemminger, Kwanho Kim, Seonjoon Kim, Cecilia Kjellman, Johan Klintin, Corne Kruger, Gerard Kruijff, Peter Kunit, Hyeyong Kwon, Bruno Legeard, Thomas Letzkus, Alon Linetzki, Balder Lingegård, Tilo Linz, Hongbiao Liu, Claire Lohr, Ine Lutterman, Marek Majernik, Rik Marselis, Romanos Matthaios, Judy McKay, Fergus McLachlan, Dénes Medzihradszky, Stefan Merkel, Armin Metzger, Don Mills, Gary Mogyorodi, Ninna Morin, Ingvar Nordström, Adam Novak, Avi Ofer, Magnus C Ohlsson, Joel Oliviera, Monika Stocklein Olsen, Kenji Onishi, Francisca Cano Ortiz, Gitte Ottosen, Tuula Pääkkönen, Ana Paiva, Tal Pe'er, Helmut Pichler, Michaël Pilaeten, Horst Pohlmann, Andrew Pollner, Meile Posthuma, Vitalijs Puiso, Salvatore Reale, Stuart Reid, Ralf Reissing, Shark Ren, Miroslav Renda, Randy Rice, Adam Roman, Jan Sabak, Hans Schaefer, Ina Schieferdecker, Franz Schiller, Jianxiong Shen, Klaus Skafte, Mike Smith, Cristina Sobrero, Marco Sogliani, Lagu Murian, Emilio Soresi, Helder Sousa, Michael Sowers, Michael Stahl, Lucjan Stapp, Li Suyuan, Toby Thompson, Steve Toms, Sagi Traybel, Sabine Uhde, Stephanie Ulrich, Philippos Vakalakis, Erik van Veenendaal, Marianne Vesterdal, Ernst von Düring, Salinda Wickramasinghe, Marie Walsh, Søren Wassard, Hans Weiberg, Paul Weymouth, Hyungjin Yoon, John Young, Surong Yuan, Ester Zabar, and Karolina Zmitrowicz.

Kelompok Kerja Dewan Kualifikasi Pengujian Perangkat Lunak Internasional Tingkat Dasar (Edisi 2018): Klaus Olsen (ketua), Tauhida Parveen (wakil ketua), Rex Black (manajer proyek), Dani Almog, Debra Friedenberg, Rashed Karim, Johan Klintin, Vipul Kocher, Corne Kruger, Sunny Kwon, Judy McKay, Thomas Müller, Igal Levi, Ebbe Munk, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh, Eshraka Zakaria, dan Stevan Zivanovic. Inti tim berterima kasih kepada tim peninjau dan semua Dewan Anggota atas saran mereka.

Dewan Kualifikasi Pengujian Perangkat Lunak Internasional Tingkat Dasar Kelompok Kerja (Edisi 2011): Thomas Müller (ketua), Debra Friedenberg. Tim inti berterima kasih kepada tim peninjau (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquier Hans Schaefer, Stephanie Ulrich, Erik van Veenendaal), dan semua Dewan Anggota atas saran mereka.

Kelompok Kerja Dewan Kualifikasi Pengujian Perangkat Lunak Internasional Tingkat Yayasan (Edisi 2010): Thomas Müller (ketua), Rahul Verma, Martin Klink dan Armin Beer. Tim inti berterima kasih atas tinjauannya tim (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Judy McKay, Tuula Pääkkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veenendaal), dan semua Dewan Anggota atas saran mereka.

Kelompok Kerja Dewan Kualifikasi Pengujian Perangkat Lunak Internasional Tingkat Dasar (Edisi 2007): Thomas Müller (ketua), Dorothy Graham, Debra Friedenberg, dan Erik van Veenendaal. Tim inti terima kasih kepada tim peninjau (Hans Schaefer,

Stephanie Ulrich, Meile Posthuma, Anders Pettersson, dan Wonil Kwon) dan semua Dewan Anggota atas saran mereka.

Kelompok Kerja Dewan Kualifikasi Pengujian Perangkat Lunak Internasional Tingkat Dasar (Edisi 2005): Thomas Müller (ketua), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson dan Erik van Veenendaal. Tim inti berterima kasih kepada tim peninjau dan semua Dewan Anggota atas saran mereka.

# 0 Pengantar

## 0.1 Tujuan Silabus

Silabus ini menjadi dasar untuk kualifikasi Kualifikasi Pengujian Perangkat Lunak Internasional untuk Tingkat Dasar. ISTQB® menyediakan silabus ini sebagai berikut:

1. Untuk anggota dewan, untuk menerjemahkan ke dalam bahasa lokal mereka dan untuk mengakreditasi penyedia pelatihan. Dewan anggota dapat menyesuaikan silabus dengan kebutuhan bahasa khusus mereka dan menambahkan referensi untuk beradaptasi dengan publikasi lokal mereka.

2. Kepada lembaga sertifikasi, untuk memperoleh pertanyaan ujian dalam bahasa lokal mereka yang disesuaikan dengan tujuan pembelajaran untuk silabus ini.

3. Kepada penyelenggara pelatihan, untuk menghasilkan perperalatanan kursus dan menentukan metode pengajaran yang tepat.

4. Untuk calon sertifikasi, untuk mempersiapkan ujian sertifikasi (baik sebagai bagian dari kursus pelatihan atau mandiri).

5. Kepada komunitas rekayasa perangkat lunak dan sistem internasional, untuk memajukan profesi perangkat lunak dan pengujian sistem, dan sebagai dasar untuk buku dan artikel.

ISTQB® dapat mengizinkan entitas lain untuk menggunakan silabus ini untuk tujuan lain, asalkan mereka mencari dan mendapatkan izin tertulis sebelumnya dari ISTQB®.

## 0.2 Certified Tester Tingkat Dasar dalam Pengujian Perangkat Lunak

Tingkat Dasar ditujukan untuk siapa saja yang terlibat dalam pengujian perangkat lunak. Ini termasuk orang-orang dalam peran seperti penguji, analis pengujian, insinyur pengujian, konsultan pengujian, manajer pengujian, penguji penerimaan pengguna, dan pengembang perangkat lunak. Tingkat Dasar ini juga cocok untuk siapa saja yang menginginkan pemahaman dasar tentang pengujian perangkat lunak, seperti pemilik produk, manajer proyek, manajer kualitas, manajer pengembangan perangkat lunak, analis bisnis, direktur TI, dan konsultan manajemen. Pemegang Sertifikat Tingkat Dasar akan dapat melanjutkan ke sertifikasi pengujian perangkat lunak tingkat yang lebih tinggi.

Gambaran ISTQB® 2018 Tingkat Dasar adalah dokumen terpisah yang masih berlaku untuk Silabus Dasar 2018 V3.1 ini dan mencakup informasi berikut:

- Hasil bisnis untuk silabus
- Matriks yang menunjukkan ketertelusuran antara hasil bisnis dan tujuan pembelajaran
- Ringkasan silabus ini

## 0.3 Tujuan Pembelajaran yang Dapat Diperiksa dan Tingkat Pengetahuan Kognitif

Tujuan pembelajaran mendukung hasil bisnis dan digunakan untuk membuat ujian Sertifikat Tingkat Dasar.

Secara umum, semua isi silabus ini dapat diuji pada tingkat K1, kecuali Pendahuluan dan Lampiran. Artinya, kandidat mungkin diminta untuk mengenali, mengingat, atau mengingat kata kunci atau konsep yang disebutkan dalam salah satu dari enam bab. Tingkat pengetahuan dari tujuan pembelajaran khusus ditunjukkan pada awal setiap bab, dan diklasifikasikan sebagai berikut:

- K1: ingat
- K2: mengerti
- K3: terapkan

Rincian lebih lanjut dan contoh tujuan pembelajaran diberikan di Lampiran B.

Definisi semua istilah yang tercantum sebagai kata kunci tepat di bawah judul bab harus diingat (K1), meskipun tidak disebutkan secara eksplisit dalam tujuan pembelajaran.

## 0.4 Ujian Sertifikasi Tingkat Dasar

Ujian Sertifikat Tingkat Dasar akan didasarkan pada silabus ini. Jawaban atas pertanyaan ujian mungkin memerlukan penggunaan materi berdasarkan lebih dari satu bagian dari silabus ini. Semua bagian dari silabus dapat diuji, kecuali untuk Pendahuluan dan Lampiran. Standar, buku, dan silabus ISTQB® lainnya dimasukkan sebagai referensi, tetapi isinya tidak dapat diuji, di luar apa yang dirangkum dalam silabus ini sendiri dari standar, buku, dan silabus ISTQB® lainnya.

Format ujian adalah pilihan ganda. Ada 40 soal. Untuk lulus ujian, setidaknya 65% dari pertanyaan (yaitu, 26 pertanyaan) harus dijawab dengan benar.

Ujian dapat diambil sebagai bagian dari kursus pelatihan terakreditasi atau diambil secara independen (misalnya, di pusat ujian atau dalam ujian publik). Menyelesaikan kursus di tempat pelatihan yang terakreditasi bukanlah prasyarat untuk ujian.

## 0.5 Akreditasi

Dewan Anggota ISTQB® dapat mengakreditasi penyedia pelatihan yang materi kursusnya mengikuti silabus ini. Penyelenggara pelatihan harus mendapatkan pedoman akreditasi dari Badan Anggota atau badan yang melakukan akreditasi. Kursus terakreditasi diakui sesuai dengan silabus ini, dan diizinkan untuk mengikuti ujian ISTQB® sebagai bagian dari kursus.

## 0.6 Tingkat Detail

Tingkat detail dalam silabus ini memungkinkan kursus dan ujian yang konsisten secara internasional. Untuk mencapai tujuan tersebut, silabus ini terdiri dari:

- Tujuan instruksional umum yang menjelaskan maksud dari tingkat dasar
- Daftar istilah yang harus dapat diingat siswa
- Tujuan pembelajaran untuk setiap bidang pengetahuan, menggambarkan hasil belajar kognitif yang ingin dicapai
- Denaskahsi konsep kunci, termasuk referensi ke sumber seperti literatur dan standar yang diterima

Isi silabus bukanlah denaskahsi dari seluruh area pengetahuan pengujian perangkat lunak; itu mencerminkan tingkat detail yang akan dicakup dalam kursus pelatihan Tingkat Dasar. Silabus ini berfokus pada konsep dan teknik pengujian yang dapat diterapkan ke semua proyek perangkat lunak, termasuk proyek Agile. Silabus ini tidak berisi tujuan pembelajaran khusus yang terkait dengan siklus hidup atau metode pengembangan perangkat lunak tertentu, tetapi membahas bagaimana konsep ini berlaku dalam proyek Agile, jenis siklus hidup iteratif dan inkremental lainnya, dan dalam siklus hidup berurutan.

## 0.7 Bagaimana Silabus ini Diorganisir

Ada enam bab dengan konten yang dapat diuji. Judul tingkat atas untuk setiap bab menentukan waktu untuk bab tersebut; waktu tidak disediakan di bawah tingkat bab. Untuk kursus pelatihan terakreditasi, silabus membutuhkan minimal 16,75 jam pengajaran, didistribusikan di enam bab sebagai berikut:

- Bab 1: 175 menit tentang Dasar-dasar Pengujian
- Bab 2: 100 menit tentang Pengujian diseluruh Siklus Hidup Pengembangan Perangkat Lunak (SDLC)
- Bab 3: 135 menit tentang Pengujian Statis
- Bab 4: 330 menit tentang Teknik Tes
- Bab 5: 225 menit tentang Manajemen Tes
- Bab 6: 40 menit untuk Peralatan-peralatan Dukungan yang digunakan untuk Tes

| 1 Dasar – dasar Pengujian | 175 menit |
|---|---|

**Kata Kunci**

cakupan, *debugging*, cacat, kesalahan, kegagalan, kualitas, jaminan kualitas, akar penyebab, analisis pengujian, dasar pengujian, kasus pengujian, penyelesaian pengujian, kondisi pengujian, kontrol pengujian, data pengujian, desain pengujian, pelaksanaan pengujian, implementasi pengujian, pemantauan pengujian , objek uji, tujuan pengujian, tes oracle, perencanaan pengujian, prosedur pengujian, proses pengujian, rangkaian tes, pengujian, testware, ketertelusuran, validasi, verifikasi

## Tujuan Pembelajaran untuk Dasar-dasar Pengujian:

### 1.1 Apa itu pengujian?

FL-1.1.1 (K1) Identifikasi tujuan umum pengujian

FL-1.1.2 (K2) Membedakan pengujian dari *debugging*

### 1.2 Mengapa Pengujian Diperlukan?

FL-1.2.1 (K2) Berikan contoh mengapa pengujian diperlukan

FL-1.2.2 (K2) Jelaskan hubungan antara pengujian dan jaminan kualitas dan berikan contoh bagaimana pengujian berkontribusi pada kualitas yang lebih tinggi

FL-1.2.3 (K2) Membedakan antara kesalahan, cacat, dan kegagalan

FL-1.2.4 (K2) Membedakan antara akar penyebab cacat dan efeknya

### 1.3 Tujuh Prinsip Pengujian

FL-1.3.1 (K2) Jelaskan tujuh prinsip pengujian

### 1.4 Proses Pengujian

FL-1.4.1 (K2) Jelaskan dampak konteks pada proses pengujian

FL-1.4.2 (K2) Jelaskan aktivitas pengujian dan tugas masing-masing dalam proses pengujian

FL-1.4.3 (K2) Membedakan produk kerja yang mendukung proses pengujian

FL-1.4.4 (K2) Jelaskan nilai pemeliharaan ketertelusuran antara basis pengujian dan produk kerja uji

### 1.5 Psikologi Pengujian

FL-1.5.1 (K1) Mengidentifikasi faktor psikologis yang mempengaruhi keberhasilan pengujian

FL-1.5.2 (K2) Jelaskan perbedaan antara pola pikir yang diperlukan untuk aktivitas pengujian dan pola pikir yang diperlukan untuk aktivitas pengembangan

## 1.1 Apa itu Pengujian?

Sistem perangkat lunak merupakan bagian integral dari kehidupan, dari aplikasi bisnis (misalnya, perbankan) hingga produk konsumen (misalnya, mobil). Kebanyakan orang memiliki pengalaman dengan perangkat lunak yang tidak bekerja seperti yang diharapkan. Perangkat lunak yang tidak berfungsi dengan benar dapat menyebabkan banyak masalah, termasuk kehilangan uang, waktu, atau reputasi bisnis, dan bahkan cedera atau kematian. Pengujian perangkat lunak adalah cara untuk menilai kualitas perangkat lunak dan untuk mengurangi risiko kegagalan perangkat lunak dalam pengoperasiannya.

Kesalahpahaman umum tentang pengujian adalah pengujian hanya terdiri dari menjalankan pengujian, yaitu, menjalankan perangkat lunak dan memeriksa hasilnya. Seperti yang dijelaskan di bagian 1.4, pengujian perangkat lunak adalah proses yang mencakup banyak aktivitas berbeda; pelaksanaan tes (termasuk pemeriksaan hasil) hanyalah salah satu dari aktivitas ini. Proses pengujian juga mencakup aktivitas seperti perencanaan pengujian, menganalisis, merancang, dan menerapkan pengujian, melaporkan kemajuan dan hasil pengujian, dan mengevaluasi kualitas objek pengujian.

Beberapa pengujian memang melibatkan eksekusi komponen atau sistem yang sedang diuji; pengujian tersebut disebut pengujian dinamis. Pengujian lainnya tidak melibatkan pelaksanaan komponen atau sistem yang sedang diuji; pengujian

tersebut disebut pengujian statis. Jadi, pengujian juga mencakup peninjauan produk kerja seperti persyaratan, cerita pengguna, dan sumber kode.

Kesalahpahaman umum lainnya tentang pengujian adalah bahwa hal itu sepenuhnya berfokus pada verifikasi persyaratan, cerita pengguna, atau spesifikasi lainnya. Sementara pengujian tidak hanya melibatkan pemeriksaan apakah sistem memenuhi persyaratan yang ditentukan, tapi juga melibatkan validasi, yaitu memeriksa apakah sistem akan memenuhi kebutuhan pengguna dan pemangku kepentingan lainnya dalam lingkungan operasionalnya.

Aktivitas pengujian diatur dan dilakukan secara berbeda dalam siklus hidup yang berbeda (lihat bagian 2.1).

### 1.1.1 Tujuan Umum Pengujian

Untuk setiap proyek tertentu, tujuan pengujian dapat mencakup:

- Untuk mencegah cacat dengan mengevaluasi produk kerja seperti persyaratan, cerita pengguna, desain, dan kode
- Untuk memverifikasi apakah semua persyaratan yang ditentukan telah dipenuhi
- Untuk memeriksa apakah objek uji selesai dan memvalidasi apakah berfungsi seperti yang diharapkan pengguna dan pemangku kepentingan lainnya
- Untuk membangun kepercayaan pada tingkat kualitas benda uji
- Untuk menemukan cacat dan kegagalan dengan mengurangi tingkat risiko kualitas perangkat lunak yang tidak memadai
- Untuk memberikan informasi yang cukup kepada para pemangku kepentingan agar mereka dapat membuat keputusan yang tepat, terutama mengenai tingkat kualitas objek uji
- Untuk mematuhi persyaratan atau standar kontrak, hukum, atau peraturan, dan/atau untuk memverifikasi kepatuhan objek uji dengan persyaratan atau standar tersebut

Tujuan pengujian dapat bervariasi, bergantung pada konteks komponen atau sistem yang sedang diuji, tingkat pengujian, dan model siklus hidup pengembangan perangkat lunak. Perbedaan ini dapat mencakup, misalnya:

- Selama pengujian komponen, salah satu tujuannya mungkin untuk menemukan sebanyak mungkin kegagalan sehingga cacat yang mendasarinya dapat diidentifikasi dan diperbaiki lebih awal. Tujuan lain mungkin untuk meningkatkan cakupan kode dari pengujian komponen.

- Selama pengujian penerimaan, satu tujuan mungkin untuk memastikan bahwa sistem bekerja seperti yang diharapkan dan memenuhi persyaratan. Tujuan lain dari pengujian ini mungkin untuk memberikan informasi kepada para pemangku kepentingan tentang risiko merilis sistem pada waktu tertentu.

### 1.1.2 Pengujian dan Debug

Pengujian dan debug itu berbeda. Melakukan pengujian dapat menunjukkan kegagalan yang disebabkan oleh cacat pada perangkat lunak. Debug adalah aktivitas pengembangan yang menemukan, menganalisis, dan memperbaiki cacat tersebut. Pengujian konfirmasi selanjutnya memeriksa apakah perbaikan mengatasi cacat. Dalam beberapa kasus, penguji bertanggung jawab atas pengujian awal dan pengujian konfirmasi akhir, sementara pengembang melakukan pengujian debug, komponen terkait, dan pengujian integrasi komponen (integrasi berkelanjutan). Namun, dalam pengembangan Agile dan dalam beberapa siklus pengembangan perangkat lunak lainnya, penguji mungkin terlibat dalam debug dan pengujian komponen.
Standar ISO (ISO/IEC/IEEE 29119-1) memiliki informasi lebih lanjut tentang konsep pengujian perangkat lunak.

## 1.2 Mengapa Pengujian Diperlukan?

Pengujian komponen dan sistem yang ketat, dan dokumentasi terkait, dapat membantu mengurangi risiko kegagalan yang terjadi selama pengoperasian. Ketika cacat terdeteksi, dan kemudian diperbaiki, ini berkontribusi pada kualitas komponen atau sistem. Selain itu, pengujian perangkat lunak mungkin juga diperlukan untuk memenuhi persyaratan kontrak atau hukum atau standar khusus industri.

### 1.2.1 Kontribusi Pengujian terhadap Kesuksesan

Sepanjang sejarah komputasi, sangat umum untuk perangkat lunak dan sistem yang akan dikirimkan ke dalam operasi dan, karena adanya cacat, kemudian menyebabkan kegagalan atau tidak memenuhi kebutuhan pemangku kepentingan. Namun, menggunakan teknik pengujian yang tepat dapat mengurangi frekuensi pengiriman bermasalah tersebut, ketika teknik tersebut diterapkan dengan tingkat keahlian pengujian yang sesuai, pada tingkat pengujian yang sesuai, dan pada titik yang sesuai dalam siklus hidup pengembangan perangkat lunak. Contohnya meliputi:

- Memiliki penguji yang terlibat dalam tinjauan persyaratan atau penyempurnaan cerita pengguna dapat mendeteksi cacat pada produk kerja ini. Identifikasi dan penghapusan cacat persyaratan mengurangi risiko pengembangan fitur yang salah atau tidak dapat diuji.

- Memiliki penguji yang bekerja sama dengan perancang sistem saat sistem sedang dirancang dapat meningkatkan pemahaman masing-masing pihak tentang desain dan cara mengujinya. Peningkatan pemahaman ini dapat mengurangi risiko cacat desain mendasar dan memungkinkan pengujian untuk diidentifikasi pada tahap awal.

- Memiliki penguji yang bekerja sama dengan pengembang saat kode sedang dikembangkan dapat meningkatkan pemahaman masing-masing pihak tentang kode dan cara mengujinya. Peningkatan pemahaman ini dapat mengurangi risiko cacat dalam kode dan pengujian.
- Memiliki penguji memverifikasi dan memvalidasi perangkat lunak sebelum rilis dapat mendeteksi kegagalan yang mungkin terlewatkan, dan mendukung proses menghilangkan cacat yang menyebabkan kegagalan (yaitu, debug). Hal ini meningkatkan kemungkinan bahwa perangkat lunak memenuhi kebutuhan pemangku kepentingan dan memenuhi persyaratan.

Selain contoh-contoh ini, pencapaian tujuan pengujian yang ditentukan (lihat bagian 1.1.1) berkontribusi pada keberhasilan pengembangan dan pemeliharaan perangkat lunak secara keseluruhan.

### 1.2.2 Jaminan Kualitas dan Pengujian

Sementara orang sering menggunakan istilah jaminan kualitas (atau hanya QA) untuk merujuk pada pengujian, jaminan kualitas dan pengujian tidak sama, tetapi mereka terkait. Sebuah konsep yang lebih besar, manajemen kualitas, mengikat mereka bersama-sama. Manajemen mutu mencakup semua aktivitas yang mengarahkan dan mengendalikan organisasi yang berkaitan dengan kualitas. Di antara aktivitas lainnya, manajemen mutu mencakup penjaminan mutu dan pengendalian mutu. Jaminan kualitas biasanya difokuskan pada kepatuhan terhadap proses yang tepat, untuk memberikan keyakinan bahwa tingkat kualitas yang sesuai akan tercapai. Ketika proses dilakukan dengan benar, produk kerja yang dibuat oleh proses tersebut umumnya berkualitas lebih tinggi, yang berkontribusi pada pencegahan cacat. Selain itu, penggunaan analisis akar masalah untuk mendeteksi dan menghilangkan penyebab cacat, bersama dengan penerapan yang tepat dari temuan pertemuan retrospektif untuk meningkatkan proses, penting untuk jaminan kualitas yang efektif.

Pengendalian kualitas melibatkan berbagai aktivitas, termasuk aktivitas pengujian, yang mendukung pencapaian tingkat kualitas yang sesuai. Aktivitas pengujian adalah bagian dari keseluruhan proses pengembangan atau pemeliharaan perangkat lunak. Karena jaminan kualitas berkaitan dengan pelaksanaan yang tepat dari seluruh proses, jaminan kualitas mendukung pengujian yang tepat. Seperti yang dijelaskan dalam bagian 1.1.1 dan 1.2.1, pengujian berkontribusi pada pencapaian kualitas dalam berbagai cara.

### 1.2.3 *Eror*, Cacat, dan Kegagalan

Seseorang dapat membuat eror (kesalahan), yang dapat menyebabkan pengenalan cacat (bug) dalam kode perangkat lunak atau di beberapa produk kerja terkait lainnya. Eror yang mengarah pada pengenalan cacat pada satu produk kerja dapat memicu eror yang mengarah pada pengenalan cacat pada produk kerja terkait. Misalnya, kesalahan elisitasi persyaratan dapat menyebabkan cacat persyaratan, yang kemudian menghasilkan kesalahan pemrograman yang menyebabkan cacat dalam kode.

Jika cacat dalam kode dijalankan, ini dapat menyebabkan kegagalan, tetapi tidak harus dalam semua keadaan. Misalnya, beberapa cacat memerlukan input atau prasyarat yang sangat spesifik untuk memicu kegagalan, yang mungkin jarang atau tidak pernah terjadi.

Eror dapat terjadi karena berbagai alasan, seperti:

- Tekanan waktu
- Kesalahan manusia
- Peserta proyek yang tidak berpengalaman atau tidak cukup terampil
- Miskomunikasi antara peserta proyek, termasuk miskomunikasi tentang persyaratan dan desain
- Kompleksitas kode, desain, arsitektur, masalah mendasar yang harus dipecahkan, dan/atau teknologi yang digunakan
- Kesalahpahaman tentang antarmuka intra-sistem dan antar-sistem, terutama ketika interaksi intrasistem dan antar-sistem tersebut jumlahnya besar
- Teknologi baru yang tidak dikenal

Selain kegagalan yang disebabkan karena cacat pada kode, kegagalan juga dapat disebabkan oleh kondisi lingkungan. Misalnya, radiasi, medan elektromagnetik, dan polusi dapat menyebabkan cacat pada *firmware* atau mempengaruhi eksekusi perangkat lunak dengan mengubah kondisi perangkat keras.

Tidak semua hasil tes yang tidak terduga adalah kegagalan. Positif palsu dapat terjadi karena kesalahan dalam cara pengujian dijalankan, atau karena cacat pada data pengujian, lingkungan pengujian, atau perangkat uji lainnya, atau karena alasan lain. Situasi terbalik juga dapat terjadi, di mana kesalahan atau cacat serupa menyebabkan negatif palsu. Negatif palsu adalah tes yang tidak mendeteksi cacat yang seharusnya terdeteksi; positif palsu dilaporkan sebagai cacat, tetapi sebenarnya bukan cacat.

### 1.2.4 Cacat, Akar Penyebab dan Akibat

Akar penyebab cacat adalah tindakan atau kondisi paling awal yang berkontribusi untuk menciptakan cacat. Cacat dapat dianalisis untuk mengidentifikasi akar penyebabnya, sehingga dapat mengurangi terjadinya cacat serupa di masa mendatang. Dengan berfokus pada akar penyebab yang paling signifikan, analisis akar masalah dapat mengarah pada perbaikan proses yang mencegah munculnya sejumlah besar cacat di masa depan.

Misalnya, pembayaran bunga yang salah, karena satu baris kode yang salah, mengakibatkan keluhan pelanggan. Kode yang rusak ditulis untuk cerita pengguna yang ambigu, karena kesalahpahaman pemilik produk tentang cara menghitung bunga. Jika ada persentase besar cacat dalam perhitungan bunga, dan cacat ini memiliki akar penyebab kesalahpahaman yang sama, pemilik produk dapat dilatih dalam topik perhitungan bunga untuk mengurangi cacat tersebut di masa depan.

Dalam contoh ini, keluhan pelanggan adalah efek. Pembayaran bunga yang salah adalah kegagalan. Perhitungan yang tidak tepat dalam kode adalah cacat, dan itu dihasilkan dari cacat asli, ambiguitas dalam cerita pengguna. Akar penyebab cacat asli adalah kurangnya pengetahuan dari pemilik produk, yang mengakibatkan pemilik produk membuat kesalahan saat menulis cerita pengguna. Proses analisis akar masalah dibahas dalam ISTQB-CTEL-TM dan ISTQB-CTEL-ITP.

## 1.3 Tujuh Prinsip Pengujian

Sejumlah prinsip pengujian telah disarankan selama 50 tahun terakhir dan menawarkan pedoman umum untuk semua pengujian.

**1. Pengujian menunjukkan adanya cacat, bukan ketidakhadirannya**

Pengujian dapat menunjukkan bahwa ada cacat, tetapi tidak dapat membuktikan bahwa tidak ada cacat. Pengujian mengurangi kemungkinan cacat yang belum ditemukan yang mungkin masih tersisa dalam perangkat lunak. Jika tidak ada cacat yang ditemukan, pengujian bukanlah bukti kebenaran.

**2. Pengujian menyeluruh itu tidak mungkin**

Menguji semuanya (semua kombinasi input dan prasyarat) tidak mungkin dilakukan kecuali untuk kasus-kasus sepele. Daripada mencoba menguji secara mendalam, analisis risiko, teknik pengujian, dan prioritas harus digunakan untuk memfokuskan upaya pengujian.

**3. Pengujian di awal menghemat waktu dan uang**

Untuk menemukan cacat lebih awal, aktivitas pengujian statis dan dinamis harus dimulai sedini mungkin dalam siklus hidup pengembangan perangkat lunak. Pengujian di awal kadang-kadang disebut sebagai shift kiri. Pengujian di awal siklus hidup pengembangan perangkat lunak membantu mengurangi atau menghilangkan perubahan yang mahal (lihat bagian 3.1).

**4. Cacat mengelompok bersama-sama**

Sejumlah kecil modul biasanya berisi sebagian besar cacat yang ditemukan selama pengujian pra-rilis, atau bertanggung jawab atas sebagian besar kegagalan operasional. Cluster cacat yang diprediksi, dan cluster cacat yang diamati sebenarnya dalam pengujian atau operasi, merupakan masukan penting ke dalam analisis risiko yang digunakan untuk memfokuskan upaya pengujian (sebagaimana disebutkan dalam prinsip 2).

**5. Waspadalah terhadap paradoks pestisida**

Jika pengujian yang sama dilakukan berulang-ulang, akhirnya pengujian tersebut tidak lagi menemukan cacat baru. Untuk mendeteksi cacat baru, pengujian yang ada dan data pengujian mungkin perlu diubah, dan pengujian baru mungkin perlu ditulis. (Pengujian tidak lagi efektif untuk menemukan cacat, sama seperti pestisida tidak lagi efektif membunuh serangga setelah beberapa saat.) Dalam beberapa kasus, seperti pengujian regresi otomatis, paradoks pestisida memiliki hasil yang menguntungkan, yaitu jumlah yang relatif rendah cacat regresi.

**6. Pengujian bergantung pada konteks**

Pengujian dilakukan secara berbeda dalam konteks yang berbeda. Misalnya, perangkat lunak kontrol industri yang kritis terhadap keselamatan diuji secara berbeda dari aplikasi seluler e-niaga. Sebagai contoh lain, pengujian dalam proyek *Agile* dilakukan secara berbeda dari pengujian dalam proyek siklus hidup pengembangan perangkat lunak sekuensial (lihat bagian 2.1).

**7. Tidak adanya kesalahan adalah kekeliruan**

Beberapa organisasi berharap bahwa penguji dapat menjalankan semua tes yang memungkinkan dan menemukan semua kemungkinan cacat. Akan tetapi seperti disebut di prinsip 1 dan 2 bahwa ini pengertian ini tidak mungkin. Lebih lanjut lagi, adalah suatu kekeliruan (keyakinan yang salah) untuk mengharapkan bahwa hanya dengan menemukan dan memperbaiki sejumlah besar cacat akan memastikan keberhasilan suatu sistem. Misalnya, menguji produk secara menyeluruh, menyelesaikan semua persyaratan yang ditentukan, dan memperbaiki semua cacat yang ditemukan kemungkinan masih dapat menghasilkan sistem yang sulit digunakan, tidak memenuhi kebutuhan dan harapan pengguna, atau kualitas yang lebih rendah dibandingkan dengan sistem pesaing lainnya.

Lihat Myers 2011, Kaner 2002, Weinberg 2008, dan Beizer 1990 untuk contoh dan prinsip pengujian lainnya.

## 1.4 Proses Pengujian

Tidak ada satupun proses pengujian perangkat lunak yang universal, akan tetapi ada serangkaian aktivitas pengujian umum yang tanpanya pengujian akan cenderung tidak mencapai tujuan yang ditetapkan. Serangkaian aktivitas pengujian ini adalah proses pengujian. Proses pengujian perangkat lunak yang tepat dan spesifik dalam situasi tertentu bergantung pada banyak faktor. Aktivitas pengujian mana yang terlibat dalam proses pengujian ini, bagaimana aktivitas ini diterapkan, dan kapan aktivitas ini terjadi dapat didiskusikan dalam strategi pengujian organisasi.

### 1.4.1 Proses Pengujian dalam Konteks

Faktor kontekstual yang mempengaruhi proses pengujian untuk suatu organisasi, termasuk, tetapi tidak terbatas pada:

- Model siklus hidup pengembangan perangkat lunak dan metodocatatani proyek yang digunakan
- Tingkat pengujian dan jenis pengujian yang sedang dipertimbangkan
- Risiko produk dan proyek
- Domain bisnis
- Kendala operasional, termasuk namun tidak terbatas pada:
    - Anggaran dan sumber daya
    - Rentang waktu
    - Kompleksitas

    - Persyaratan kontrak dan regulasi
- Kebijakan dan praktik organisasi
- Standar internal dan eksternal yang diperlukan

Bagian berikut menjelaskan aspek umum dari proses pengujian organisasi dalam hal berikut:

- Aktivitas dan tugas pengujian
- Uji produk kerja
- Ketertelusuran antara basis uji dan produk kerja uji

Hal ini sangat berguna jika dasar pengujian (untuk setiap tingkat atau jenis pengujian yang sedang dipertimbangkan) memiliki kriteria cakupan terukur yang ditentukan. Kriteria cakupan dapat bertindak secara efektif sebagai indikator kinerja utama (IKU) untuk mendorong aktivitas yang menunjukkan pencapaian tujuan pengujian perangkat lunak (lihat bagian 1.1.1).

Misalnya, untuk aplikasi seluler, dasar pengujian dapat mencakup daftar persyaratan dan daftar perangkat seluler yang didukung. Setiap persyaratan merupakan elemen dasar pengujian. Setiap perangkat yang didukung juga merupakan elemen dasar pengujian. Kriteria cakupan mungkin memerlukan setidaknya satu kasus uji untuk setiap elemen dasar pengujian. Setelah dijalankan, hasil tes ini memberi tahu pemangku kepentingan apakah persyaratan yang ditentukan terpenuhi dan apakah kegagalan diamati pada perangkat yang didukung.

Standar ISO (ISO/IEC/IEEE 29119-2) memiliki informasi lebih lanjut tentang proses pengujian.

### 1.4.2 Aktivitas dan Tugas Pengujian

Sebuah proses pengujian terdiri dari kelompok-kelompok utama aktivitas berikut:

- Perencanaan tes
- Pemantauan dan pengendalian tes
- Analisis pengujian
- Desain tes
- Implementasi tes
- Pelaksanaan pengujian
- Penyelesaian pengujian

Setiap kelompok utama aktivitas terdiri dari aktivitas-aktivitas penyusunnya, yang akan dijelaskan pada sub-bab di bawah ini. Setiap aktivitas konstituen terdiri dari beberapa tugas individu, yang akan bervariasi dari satu proyek atau rilis ke proyek lainnya.

Lebih lanjut, meskipun banyak dari grup aktivitas utama ini mungkin tampak berurutan secara logis, mereka sering diimplementasikan secara iteratif. Misalnya, pengembangan Agile melibatkan iterasi kecil dari desain, pembuatan, dan pengujian perangkat lunak yang terjadi secara terus menerus, didukung oleh perencanaan yang sedang berjalan. Jadi aktivitas pengujian juga terjadi secara berulang dan berkelanjutan dalam pendekatan pengembangan perangkat lunak ini. Bahkan dalam pengembangan perangkat lunak berurutan, urutan logis bertahap dari kelompok aktivitas utama akan melibatkan tumpang tindih, kombinasi, konkurensi, atau penghilangan, sehingga penyesuaian kelompok utama aktivitas ini dalam konteks sistem dan proyek biasanya diperlukan.

**Perencanaan tes**

Perencanaan tes melibatkan aktivitas yang menentukan tujuan pengujian dan pendekatan untuk memenuhi tujuan pengujian dalam batasan yang ditentukan oleh konteks (misalnya, menentukan teknik dan tugas pengujian yang sesuai, dan merumuskan jadwal pengujian untuk memenuhi tenggat waktu). Rencana pengujian dapat ditinjau kembali berdasarkan umpan balik dari aktivitas pemantauan dan pengendalian. Perencanaan pengujian dijelaskan lebih lanjut di bagian 5.2.

**Pemantauan dan pengendalian tes**

Pemantauan tes melibatkan perbandingan kemajuan aktual yang sedang berlangsung terhadap kemajuan yang direncanakan menggunakan metrik pemantauan tes yang ditentukan dalam rencana pengujian. Kontrol tes melibatkan pengambilan tindakan yang diperlukan untuk memenuhi tujuan rencana pengujian (yang dapat diperbarui dari waktu ke waktu). Pemantauan dan kontrol pengujian didukung oleh evaluasi kriteria keluar, yang dirujuk sebagai definisi yang dilakukan dalam beberapa model siklus hidup pengembangan perangkat lunak (lihat ISTQB-CTFL-AT). Misalnya, evaluasi kriteria keluar untuk pelaksanaan pengujian sebagai bagian dari tingkat pengujian yang diberikan dapat mencakup:

- Memeriksa hasil tes dan catatan terhadap kriteria cakupan yang ditentukan
- Menilai tingkat kualitas komponen atau sistem berdasarkan hasil pengujian dan catatan
- Menentukan apakah pengujian lebih lanjut diperlukan (misalnya, jika pengujian yang semula dimaksudkan untuk mencapai tingkat cakupan risiko
- produk tertentu gagal dilakukan, diperlukan pengujian tambahan untuk ditulis dan dijalankan)

Kemajuan pengujian terhadap rencana dikomunikasikan kepada pemangku kepentingan dalam laporan kemajuan pengujian, termasuk penyimpangan dari rencana dan informasi untuk mendukung setiap keputusan untuk menghentikan pengujian.

Pemantauan dan pengendalian tes dijelaskan lebih lanjut di bagian 5.3.

**Analisis pengujian**

Selama analisis pengujian, dasar pengujian dianalisis untuk mengidentifikasi fitur yang dapat diuji dan menentukan kondisi pengujian terkait. Dengan kata lain, analisis pengujian menentukan "apa yang akan di tes" dalam hal kriteria cakupan yang terukur.

Analisis pengujian mencakup aktivitas utama berikut:

- Menganalisis dasar pengujian yang sesuai dengan tingkat pengujian yang sedang dipertimbangkan, misalnya:
    - Spesifikasi persyaratan, seperti persyaratan bisnis, persyaratan fungsional, persyaratan sistem, cerita pengguna, *epic*, kasus penggunaan, atau produk kerja serupa yang menentukan komponen fungsional dan non-fungsional yang diinginkan atau perilaku sistem
    - Informasi desain dan implementasi, seperti diagram atau dokumen arsitektur sistem atau perangkat lunak, spesifikasi desain, *call flow graph*, diagram pemodelan (misalnya, UML atau diagram hubungan entitas), spesifikasi antarmuka, atau produk kerja serupa yang menentukan komponen atau struktur sistem
    - Implementasi komponen atau sistem itu sendiri, termasuk kode, metadata database dan kueri, dan antarmuka
    - Laporan analisis risiko, yang dapat mempertimbangkan aspek fungsional, non-fungsional, dan struktural dari komponen atau sistem
- Mengevaluasi dasar pengujian dan item pengujian untuk mengidentifikasi berbagai jenis cacat, seperti:
    - Ambiguitas
    - Kelalaian
    - Inkonsistensi
    - Ketidakakuratan
    - Kontradiksi

    - Pernyataan berlebihan
- Mengidentifikasi fitur dan kumpulan fitur yang akan diuji
- Mendefinisikan dan memprioritaskan kondisi pengujian untuk setiap fitur berdasarkan analisis dasar pengujian, dan mempertimbangkan karakteristik fungsional, non-fungsional, dan struktural, faktor bisnis dan teknis lainnya, dan tingkat risiko
- Menangkap ketertelusuran dua arah antara setiap elemen dasar pengujian dan kondisi pengujian terkait (lihat bagian 1.4.3 dan 1.4.4)

Penerapan teknik pengujian kotak hitam, kotak putih, dan berbasis pengalaman dapat berguna dalam proses analisis pengujian (lihat bab 4) untuk mengurangi kemungkinan mengabaikan kondisi pengujian yang penting dan untuk menentukan kondisi pengujian yang lebih tepat dan akurat.

Dalam beberapa kasus, analisis pengujian menghasilkan kondisi pengujian yang akan digunakan sebagai tujuan pengujian dalam piagam pengujian. Piagam uji adalah produk kerja tipikal dalam beberapa jenis pengujian berbasis pengalaman (lihat bagian 4.4.2). Ketika tujuan pengujian ini dapat dilacak ke dasar pengujian, cakupan yang dicapai selama pengujian berbasis pengalaman tersebut dapat diukur.

Identifikasi cacat selama analisis pengujian merupakan manfaat potensial yang penting, terutama jika tidak ada proses peninjauan lain yang digunakan dan/atau proses pengujian terkait erat dengan proses peninjauan. Aktivitas analisis pengujian tersebut tidak hanya memverifikasi apakah persyaratan konsisten, diungkapkan dengan benar, dan lengkap, tetapi juga memvalidasi apakah persyaratan benar menangkap pelanggan, pengguna, dan kebutuhan pemangku kepentingan lainnya. Misalnya, teknik seperti pengembangan yang digerakkan oleh perilaku (BDD) dan pengembangan yang didorong oleh uji penerimaan (ATDD), yang melibatkan pembuatan kondisi pengujian dan kasus uji dari cerita pengguna dan kriteria penerimaan sebelum pengkodean. Teknik ini juga memverifikasi, memvalidasi, dan mendeteksi cacat dalam cerita pengguna dan kriteria penerimaan (lihat ISTQB-CTFL-AT).

**Desain tes**

Selama desain tes, kondisi pengujian dijabarkan ke dalam kasus uji tingkat tinggi, set kasus uji tingkat tinggi, dan perangkat uji lainnya. Jadi, analisis pengujian menjawab pertanyaan “apa yang harus diuji?” sementara desain tes menjawab pertanyaan "bagaimana cara menguji?"

Desain tes mencakup aktivitas utama berikut:

- Merancang dan memprioritaskan kasus uji dan set kasus uji
- Mengidentifikasi data pengujian yang diperlukan untuk mendukung kondisi pengujian dan kasus pengujian
- Merancang lingkungan pengujian dan mengidentifikasi infrastruktur dan peralatan yang diperlukan
- Menangkap ketertelusuran dua arah antara basis pengujian, kondisi pengujian, dan kasus uji (lihat bagian 1.4.4)

Elaborasi kondisi pengujian ke dalam kasus uji dan set kasus uji selama desain tes sering melibatkan penggunaan teknik pengujian (lihat bab 4).

Seperti halnya analisis pengujian, desain tes juga dapat menghasilkan identifikasi jenis cacat yang serupa pada dasar pengujian. Juga, seperti analisis pengujian, identifikasi cacat selama desain tes merupakan manfaat potensial yang penting.

### Implementasi tes

Selama implementasi pengujian, perangkat uji yang diperlukan untuk pelaksanaan pengujian dibuat dan/atau diselesaikan, termasuk mengurutkan kasus uji ke dalam prosedur pengujian. Jadi, desain pengujian menjawab pertanyaan “bagaimana cara menguji?” sementara implementasi tes menjawab pertanyaan "apakah kita sekarang memiliki segalanya untuk menjalankan tes?"

Pelaksanaan tes meliputi aktivitas utama berikut:

- Mengembangkan dan memprioritaskan prosedur pengujian dan berpotensi untuk membuat naskah pengujian otomatis
- Membuat rangkaian pengujian dari prosedur pengujian dan (jika ada) naskah pengujian otomatis
- Mengatur rangkaian pengujian dalam jadwal eksekusi pengujian sedemikian rupa sehingga menghasilkan eksekusi pengujian yang efisien (lihat bagian 5.2.4)
- Membangun lingkungan pengujian (termasuk, berpotensi seperti peralatan pengujian, virtualisasi layanan, simulator, dan item infrastruktur lainnya) dan memverifikasi bahwa semua yang diperlukan telah disiapkan dengan benar
- Mempersiapkan data pengujian dan memastikan data tersebut sudah dimuat dengan benar di lingkungan pengujian
- Memverifikasi dan memperbarui ketertelusuran dua arah antara basis pengujian, kondisi pengujian, kasus pengujian, prosedur pengujian, dan rangkaian pengujian (lihat bagian 1.4.4)

Desain tes dan tugas implementasi tes sering digabungkan.

Dalam pengujian eksplorasi dan jenis pengujian berbasis pengalaman lainnya, desain dan implementasi pengujian dapat terjadi, dan dapat didokumentasikan, sebagai bagian dari pelaksanaan pengujian. Pengujian eksplorasi dapat didasarkan pada piagam pengujian (diproduksi sebagai bagian dari analisis pengujian), dan pengujian eksplorasi dilaksanakan segera saat dirancang dan diimplementasikan (lihat bagian 4.4.2).

### Pelaksanaan pengujian

Selama pelaksanaan pengujian, rangkaian pengujian dijalankan sesuai dengan jadwal eksekusi pengujian.

Pelaksanaan pengujian mencakup aktivitas utama berikut:

- Merekam ID dan versi item uji atau objek uji, peralatan pengujian, dan perangkat uji
- Menjalankan tes baik secara manual atau dengan menggunakan peralatan pelaksanaan pengujian
- Membandingkan hasil aktual dengan hasil yang diharapkan
- Menganalisis anomali untuk menentukan kemungkinan penyebabnya (misalnya, kegagalan dapat terjadi karena cacat pada kode, tetapi positif palsu juga dapat terjadi (lihat bagian 1.2.3)
- Melaporkan cacat berdasarkan kegagalan yang diamati (lihat bagian 5.6)
- Mencatat hasil pelaksanaan pengujian (misalnya: lulus, gagal, diblokir)
- Mengulangi aktivitas pengujian baik sebagai akibat dari tindakan yang diambil untuk suatu anomali, atau sebagai bagian dari pengujian yang direncanakan (misalnya, pelaksanaan pengujian yang dikoreksi, pengujian konfirmasi, dan/atau pengujian regresi)

- Memverifikasi dan memperbarui ketertelusuran dua arah antara dasar pengujian, kondisi pengujian, kasus pengujian, prosedur pengujian, dan hasil pengujian.

**Penyelesaian pengujian**

Aktivitas penyelesaian pengujian mengumpulkan data dari aktivitas pengujian yang diselesaikan untuk mengkonsolidasikan pengalaman, perangkat uji, dan informasi relevan lainnya. Aktivitas penyelesaian pengujian terjadi pada tonggak proyek seperti ketika sistem perangkat lunak dirilis, proyek pengujian selesai (atau dibatalkan), iterasi proyek Agile selesai, tingkat pengujian selesai, atau rilis pemeliharaan telah selesai.

Penyelesaian pengujian mencakup aktivitas utama berikut:

- Memeriksa apakah semua laporan cacat ditutup, memasukkan permintaan perubahan atau item jaminan simpanan produk untuk setiap cacat yang tetap tidak terselesaikan pada akhir pelaksanaan pengujian
- Membuat laporan ringkasan pengujian untuk dikomunikasikan kepada pemangku kepentingan
- Menyelesaikan dan mengarsipkan lingkungan pengujian, data pengujian, infrastruktur pengujian, dan perangkat uji lainnya untuk digunakan kembali nanti
- Menyerahkan peralatan-peralatan pengujian kepada tim pemeliharaan, tim proyek lain, dan/atau pemangku kepentingan lain yang dapat memperoleh manfaat dari penggunaannya
- Menganalisis pelajaran yang dipetik dari aktivitas pengujian yang telah diselesaikan untuk menentukan perubahan yang diperlukan untuk iterasi, rilis, dan proyek di masa mendatang
- Menggunakan informasi yang dikumpulkan untuk meningkatkan kematangan proses pengujian

### 1.4.3 Produk Uji Kerja

Produk kerja pengujian dibuat sebagai bagian dari proses pengujian. Sama seperti ada variasi yang signifikan dalam cara organisasi menerapkan proses pengujian, ada juga variasi yang signifikan dalam jenis produk kerja yang dibuat selama proses tersebut, dalam cara produk kerja tersebut diatur dan dikelola, dan dalam nama yang digunakan untuk produk tersebut. Silabus ini mengikuti proses pengujian yang diuraikan di atas, dan produk kerja yang dijelaskan dalam silabus ini dan di Daftar Istilah ISTQB®. Standar ISO (ISO/IEC/IEEE 29119-3) juga dapat berfungsi sebagai pedoman untuk produk kerja uji.

Banyak produk pekerjaan pengujian yang dijelaskan di bagian ini dapat ditangkap dan dikelola menggunakan peralatan manajemen pengujian dan peralatan manajemen cacat (lihat bab 6).

**Produk kerja perencanaan pengujian**

Produk kerja perencanaan pengujian biasanya mencakup satu atau lebih rencana pengujian. Rencana pengujian mencakup informasi tentang dasar pengujian, yang akan dihubungkan dengan produk pekerjaan pengujian lainnya melalui informasi ketertelusuran (lihat di bawah dan bagian 1.4.4), serta kriteria keluar (atau definisi

selesai) yang akan digunakan selama pengujian pemantauan dan pengendalian. Rencana pengujian dijelaskan di bagian 5.2.

**Produk kerja pemantauan dan kontrol pengujian**

Produk kerja pemantauan dan kontrol pengujian biasanya mencakup berbagai jenis laporan pengujian, termasuk laporan kemajuan pengujian yang dihasilkan secara berkelanjutan dan/atau secara teratur, dan laporan ringkasan pengujian yang dihasilkan pada berbagai pencapaian penyelesaian. Semua laporan pengujian harus memberikan perincian yang relevan

dengan hadirin tentang kemajuan pengujian pada tanggal laporan, termasuk meringkas hasil pelaksanaan pengujian setelah tersedia.

Pengujian pemantauan dan pengendalian produk kerja juga harus mengatasi masalah manajemen proyek, seperti penyelesaian tugas, alokasi dan penggunaan sumber daya, dan upaya.

Pemantauan dan kontrol pengujian, dan produk kerja yang dibuat selama aktivitas ini, dijelaskan lebih lanjut di bagian 5.3 silabus ini.

**Produk kerja analisis pengujian**

Produk kerja analisis pengujian mencakup kondisi pengujian yang ditentukan dan diprioritaskan, yang masing-masing idealnya dapat dilacak secara dua arah ke elemen spesifik dari dasar pengujian yang dicakupnya. Untuk pengujian eksplorasi, analisis pengujian mungkin melibatkan pembuatan piagam pengujian. Analisis pengujian juga dapat menghasilkan penemuan dan pelaporan cacat dalam basis pengujian.

**Produk kerja desain pengujian**

Hasil desain pengujian dalam kasus uji dan set kasus uji untuk menjalankan kondisi pengujian yang ditentukan dalam analisis pengujian. Seringkali merupakan praktik yang baik untuk merancang kasus uji tingkat tinggi, tanpa nilai konkret untuk data masukan dan hasil yang diharapkan. Kasus uji tingkat tinggi tersebut dapat digunakan kembali di beberapa siklus pengujian dengan data konkret yang berbeda, sambil tetap mendokumentasikan ruang lingkup kasus uji secara memadai. Idealnya, setiap kasus uji dapat dilacak secara dua arah ke kondisi pengujian yang dicakupnya.

Desain pengujian juga menghasilkan:

- desain dan/atau identifikasi data uji yang diperlukan
- desain lingkungan pengujian
- identifikasi infrastruktur dan peralatan-peralatan

Meskipun sejauh mana hasil ini didokumentasikan bervariasi secara signifikan.

**Produk kerja pelaksanaan pengujian**

Produk kerja pelaksanaan pengujian meliputi:

- Prosedur pengujian dan urutan prosedur pengujian tersebut
- Rangkain tes
- Jadwal pelaksanaan tes

Idealnya, setelah implementasi pengujian selesai, pencapaian kriteria cakupan yang ditetapkan dalam rencana pengujian dapat ditunjukkan melalui ketertelusuran dua arah

antara prosedur pengujian dan elemen spesifik dari dasar pengujian, melalui kasus pengujian dan kondisi pengujian.

Dalam beberapa kasus, implementasi pengujian melibatkan pembuatan produk kerja menggunakan atau digunakan oleh peralatan, seperti virtualisasi layanan dan naskah pengujian otomatis.

Pelaksanaan pengujian juga dapat mengakibatkan pembuatan dan verifikasi data pengujian dan lingkungan pengujian. Kelengkapan dokumentasi data dan/atau hasil verifikasi lingkungan dapat berbeda secara signifikan.

Data uji berfungsi untuk menetapkan nilai konkret pada masukan dan hasil yang diharapkan dari kasus uji. Nilai konkret seperti itu, bersama dengan arahan eksplisit tentang penggunaan nilai konkret, mengubah kasus uji tingkat tinggi menjadi kasus uji tingkat rendah yang dapat dieksekusi. Kasus uji tingkat tinggi yang sama dapat menggunakan data uji yang berbeda saat dieksekusi pada rilis objek uji yang berbeda. Hasil konkret yang diharapkan yang terkait dengan data uji konkret diidentifikasi dengan menggunakan oracle uji.

Dalam pengujian eksplorasi, beberapa desain pengujian dan produk kerja implementasi dapat dibuat selama pelaksanaan pengujian, meskipun sejauh mana pengujian eksplorasi (dan ketertelusurannya ke elemen dasar pengujian tertentu) didokumentasikan dapat bervariasi secara signifikan.

Kondisi pengujian yang ditentukan dalam analisis pengujian dapat disempurnakan lebih lanjut dalam implementasi pengujian.

**Produk kerja pelaksanaan pengujian**

Produk kerja pelaksanaan pengujian meliputi:

- Dokumentasi status kasus uji individu atau prosedur pengujian (misalnya, siap dijalankan, lulus, gagal, diblokir, sengaja dilewati, dll.)
- Laporan cacat (lihat bagian 5.6)
- Dokumentasi tentang item uji, objek uji, peralatan pengujian, dan perangkat uji mana yang terlibat dalam pengujian

Idealnya, setelah pelaksanaan pengujian selesai, status setiap elemen dari basis pengujian dapat ditentukan dan dilaporkan melalui ketertelusuran dua arah dengan prosedur pengujian yang terkait. Misalnya, kita dapat mengatakan persyaratan mana yang telah lulus semua pengujian yang direncanakan, persyaratan mana yang gagal dalam pengujian dan/atau memiliki cacat yang terkait dengannya, dan persyaratan mana yang telah direncanakan pengujian yang masih menunggu untuk dijalankan. Hal ini memungkinkan verifikasi bahwa kriteria cakupan telah terpenuhi, dan memungkinkan pelaporan hasil pengujian dalam istilah yang dapat dimengerti oleh pemangku kepentingan.

**Produk kerja penyelesaian pengujian**

Produk kerja penyelesaian pengujian mencakup laporan ringkasan pengujian, item tindakan untuk peningkatan proyek atau iterasi berikutnya, permintaan perubahan atau item jaminan simpanan produk, dan perangkat pengujian yang diselesaikan.

### 1.4.4 Ketertelusuran antara Basis Pengujian dan Produk Uji Kerja

Sebagaimana disebutkan dalam bagian 1.4.3, produk kerja uji dan nama produk kerja tersebut sangat bervariasi. Terlepas dari variasi ini, untuk menerapkan pemantauan dan

kontrol pengujian yang efektif, penting untuk menetapkan dan memelihara ketertelusuran selama proses pengujian antara setiap elemen dasar pengujian dan berbagai produk kerja pengujian yang terkait dengan elemen tersebut, seperti dijelaskan di atas. Selain evaluasi cakupan pengujian, kemampuan penelusuran yang
baik mendukung:

- Menganalisis dampak perubahan
- Membuat pengujian dapat diaudit
- Memenuhi kriteria tata kelola TI
- Meningkatkan pemahaman laporan kemajuan pengujian dan laporan ringkasan pengujian untuk memasukkan status elemen dasar pengujian (misalnya, persyaratan yang lulus pengujian, persyaratan yang gagal dalam pengujian, dan persyaratan yang menunggu pengujian)
- Menghubungkan aspek teknis pengujian dengan pemangku kepentingan dalam istilah yang dapat mereka pahami

- Menyediakan informasi untuk menilai kualitas produk, kapabilitas proses, dan kemajuan proyek terhadap tujuan bisnis

Beberapa peralatan manajemen pengujian menyediakan model produk kerja pengujian yang cocok dengan sebagian atau seluruh produk kerja pengujian yang diuraikan di bagian ini. Beberapa organisasi membangun sistem manajemen mereka sendiri untuk mengatur produk kerja dan menyediakan ketertelusuran informasi yang mereka butuhkan.

## 1.5 Psikologi Pengujian

Pengembangan perangkat lunak, termasuk pengujian perangkat lunak, melibatkan manusia. Oleh karena itu, psikologi manusia memiliki efek penting pada pengujian perangkat lunak.

### 1.5.1 Psikologi Manusia dan Pengujian

Mengidentifikasi cacat selama pengujian statis seperti tinjauan persyaratan atau sesi penyempurnaan cerita pengguna, atau mengidentifikasi kegagalan selama pelaksanaan pengujian dinamis, dapat dianggap sebagai kritik terhadap produk dan pembuatnya. Unsur psikologi manusia yang disebut bias konfirmasi dapat mempersulit penerimaan informasi yang tidak sesuai dengan keyakinan yang dianut saat ini. Misalnya, karena pengembang mengharapkan kode mereka benar, mereka memiliki bias konfirmasi yang menyulitkan untuk menerima bahwa kode tersebut salah. Selain bias konfirmasi, bias kognitif lainnya dapat mempersulit orang untuk memahami atau menerima informasi yang dihasilkan oleh pengujian. Lebih jauh, adalah sifat umum manusia untuk menyalahkan pembawa berita buruk, dan informasi yang dihasilkan oleh pengujian sering kali berisi berita buruk.

Sebagai akibat dari faktor psikologis ini, beberapa orang mungkin menganggap pengujian sebagai aktivitas yang merusak, meskipun hal itu berkontribusi besar terhadap kemajuan proyek dan kualitas produk (lihat bagian 1.1 dan 1.2). Untuk mencoba mengurangi persepsi ini, informasi tentang cacat dan kegagalan harus dikomunikasikan dengan cara yang konstruktif. Dengan cara ini, ketegangan antara penguji dan analis, pemilik produk, desainer, dan pengembang dapat dikurangi. Ini berlaku selama pengujian statis dan dinamis.

Penguji dan manajer pengujian perlu memiliki keterampilan interpersonal yang baik untuk dapat berkomunikasi secara efektif tentang cacat, kegagalan, hasil pengujian,

kemajuan pengujian, dan risiko, dan untuk membangun hubungan positif dengan rekan kerja. Cara berkomunikasi yang baik meliputi contoh-contoh berikut:

- Mulailah dengan kolaborasi daripada pertempuran. Ingatkan semua orang tentang tujuan bersama untuk mendapatkan sistem kualitas yang lebih baik.
- Tekankan manfaat pengujian. Misalnya, bagi penulis, informasi cacat dapat membantu mereka meningkatkan produk kerja dan keterampilan mereka. Bagi organisasi, cacat yang ditemukan dan diperbaiki selama pengujian akan menghemat waktu dan uang serta mengurangi risiko keseluruhan terhadap kualitas produk.
- Komunikasikan hasil tes dan temuan lainnya dengan cara yang netral dan berfokus pada fakta tanpa mengkritik orang yang membuat item yang cacat. Tulis laporan cacat yang objektif dan faktual dan tinjau temuannya.
- Cobalah untuk memahami bagaimana perasaan orang lain dan alasan mereka bereaksi negatif terhadap informasi tersebut.
- Konfirmasikan bahwa orang lain telah memahami apa yang telah dikatakan dan sebaliknya.

Tujuan tes yang khas telah dibahas sebelumnya (lihat bagian 1.1). Mendefinisikan secara jelas rangkaian tujuan tes yang tepat memiliki implikasi psikologis yang penting. Kebanyakan orang cenderung menyelaraskan rencana dan perilaku mereka dengan tujuan yang ditetapkan oleh tim, manajemen, dan pemangku kepentingan lainnya. Penting juga bahwa penguji mematuhi tujuan ini dengan bias pribadi yang minimal.

### 1.5.2 Pola Pikir Penguji dan Pengembang

Pengembang dan penguji sering kali berpikir secara berbeda. Tujuan utama pengembangan adalah merancang dan membangun produk. Seperti dibahas sebelumnya, tujuan pengujian termasuk memverifikasi dan memvalidasi produk, menemukan cacat sebelum rilis, dan sebagainya. Ini adalah rangkaian tujuan yang berbeda yang membutuhkan pola pikir yang berbeda. Menyatukan pola pikir ini membantu mencapai tingkat kualitas produk yang lebih tinggi.

Pola pikir mencerminkan asumsi individu dan metode pilihan untuk pengambilan keputusan dan pemecahan masalah. Pola pikir seorang penguji harus mencakup rasa ingin tahu, pesimisme profesional, pandangan kritis, perhatian terhadap detail, dan motivasi untuk komunikasi dan hubungan yang baik dan positif. Pola pikir penguji cenderung tumbuh dan matang saat penguji memperoleh pengalaman.

Pola pikir pengembang mungkin mencakup beberapa elemen pola pikir penguji, tetapi pengembang yang sukses sering kali lebih tertarik untuk merancang dan membangun solusi daripada memikirkan apa yang mungkin salah dengan solusi tersebut. Selain itu, bias konfirmasi membuat sulit untuk menyadari kesalahan yang dilakukan oleh mereka sendiri.

Dengan pola pikir yang benar, pengembang dapat menguji kode mereka sendiri. Model siklus hidup pengembangan perangkat lunak yang berbeda sering kali memiliki cara yang berbeda dalam mengatur penguji dan aktivitas pengujian. Memiliki beberapa aktivitas pengujian yang dilakukan oleh penguji independen meningkatkan efektivitas deteksi cacat, yang sangat penting untuk sistem yang besar, kompleks, atau kritis terhadap keselamatan. Penguji independen membawa perspektif yang berbeda dari penulis produk kerja (yaitu, analis bisnis, pemilik produk, desainer, dan pengembang), karena mereka memiliki bias kognitif yang berbeda dari penulis.

# 2 Pengujian Seluruh Perangkat Lunak Pengembangan Siklus Hidup — 100 menit

**Kata Kunci**

pengujian penerimaan, pengujian alfa, pengujian beta, pengujian terkait perubahan, komersial lepas dari rak (*commercial off-the-shelf*, COTS), pengujian integrasi komponen, pengujian komponen, pengujian konfirmasi, pengujian penerimaan kontraktual, pengujian fungsional, analisis dampak, pengujian integrasi, pengujian pemeliharaan, pengujian non-fungsional, pengujian penerimaan operasional, pengujian regresi, pengujian penerimaan peraturan, model pengembangan berurutan, pengujian integrasi sistem, pengujian sistem, dasar pengujian, kasus pengujian, lingkungan pengujian, tingkat pengujian, pengujian objek, tujuan pengujian, jenis pengujian, pengujian penerimaan pengguna, pengujian kotak-putih

## Tujuan Pembelajaran untuk Pengujian Sepanjang Siklus Hidup Pengembangan Perangkat Lunak

### 2.1 Model Siklus Hidup Pengembangan Perangkat Lunak

FL-2.1.1 (K2) Jelaskan hubungan antara aktivitas pengembangan perangkat lunak dan aktivitas pengujian dalam siklus hidup pengembangan perangkat lunak

FL-2.1.2 (K1) Mengidentifikasi alasan mengapa model siklus hidup pengembangan perangkat lunak harus disesuaikan dengan konteks proyek dan karakteristik produk

### 2.2 Tingkatan Pengujian

FL-2.2.1 (K2) Bandingkan perbedaan tingkatan tes dari perspektif tujuan, dasar tes, tes objek, cacat dan kegagalan tipikal, serta pendekatan dan tanggung jawab

### 2.3 Tipe-tipe Pengujian

FL-2.3.1 (K2) Bandingkan pengujian fungsional, non-fungsional, dan kotak-putih.

FL-2.3.2 (K1) Kenali pengujian fungsional, non-fungsional, dan pengujian kotak-putih terjadi pada setiap tingkat pengujian.

FL-2.3.3 (K2) Bandingkan tujuan dari pengujian konfirmasi dan pengujian regresi.

### 2.4 Pengujian Pemeliharaan

FL-2.4.1 (K2) Ringkasan pemicu untuk pengujian pemeliharaan.

FL-2.4.2 (K2) Jelaskan peran analisis dampak dalam pengujian pemeliharaan.

# 2.1 Model - Model Pengembangan Siklus Hidup Perangkat Lunak

Suatu model siklus hidup pengembangan perangkat lunak yang menggambarkan jenis aktivitas yang dilakukan pada setiap tahap dalam sebuah proyek pengembangan perangkat lunak, dan bagaimana aktivitas tersebut berhubungan satu sama lain secara logis dan kronologis. Terdapat sejumlah model siklus hidup pengembangan perangkat lunak yang berlainan, masing-masing juga membutuhkan model pendekatan yang berbeda untuk pengujian.

### 2.1.1 Pengembangan Perangkat Lunak dan Pengujian Perangkat Lunak

Merupakan bagian penting dari peran penguji untuk terbiasa dengan model siklus hidup pengembangan perangkat lunak yang umum sehingga aktivitas pengujian yang sesuai dapat dilakukan.

Dalam setiap model siklus hidung pengembangan perangkat lunak, terdapat beberapa karakteristik pengujian yang baik:

- Untuk setiap aktivitas pengembangan, terdapat aktivitas pengujian yang sesuai
- Setiap tingkat tes memiliki tujuan tes khusus untuk tingkat tersebut
- Analisis pengujian dan desain pengujian untuk tingkat pengujian tertentu dimulai selama pengembangan aktivitas yang sesuai
- Para penguji berpartisipasi dalam diskusi untuk menentukan dan menyempurnakan persyaratan dan desain, serta terlibat dalam meninjau produk kerja (misalnya, persyaratan, desain, cerita pengguna, dll.) segera setelah konsepnya tersedia

Apapun model siklus hidup pengembangan perangkat lunak yang dipilih, aktivitas pengujian harus dimulai sejak awal tahapan siklus hidup, mengikuti prinsip pengujian yakni pengujian dari tahap awal.

Silabus ini mengkategorikan model umum siklus hidup pengembangan perangkat lunak sebagai berikut:

- Model pengembangan berurutan
- Model pengembangan berulang dan inkremental

Model pengembangan berurutan yang menggambarkan proses pengembangan perangkat lunak sebagai linier, aliran aktivitas yang berurutan. Ini berarti bahwa setiap fase dalam proses pengembangan harus dimulai ketika fase sebelumnya telah selesai. Secara teori, tidak ada fase yang tumpang tindih, tetapi dalam praktiknya, ini bermanfaat untuk memilih lebih awal umpan balik dari fase berikutnya.

Dalam Model *Waterfall* (air terjun), aktivitas pengembangan (misalnya, analisis kebutuhan, desain, pengkodean, pengujian) diselesaikan satu demi satu. Dalam model ini, aktivitas pengujian hanya terjadi setelah semua aktivitas pengembangan lainnya telah selesai.

Berbeda dengan model *Waterfall,* model-V mengintegrasikan proses pengujian di seluruh proses pengembangan, menerapkan prinsip pengujian awal. Selanjutnya, model-V mencakup tingkat pengujian yang terkait dengan masing-masing fase pengembangan yang sesuai, yang selanjutnya mendukung pengujian awal (lihat bagian 2.2 untuk diskusi tingkatan tes). Dalam model ini, pelaksanaan tes yang terkait dengan setiap tingkat tes berlangsung secara berurutan, tetapi dalam beberapa kasus terjadi tumpang tindih.

Model pengembangan berurutan menghasilkan perangkat lunak yang berisi serangkaian fitur lengkap, tetapi biasanya membutuhkan bulan atau tahun untuk diserahkan ke pemangku kepentingan dan pengguna.

Pengembangan inkremental melibatkan penetapan persyaratan, perancangan, pembangunan, dan pengujian sistem yang berarti bahwa fitur perangkat lunak tumbuh secara bertahap. Ukuran peningkatan fitur ini bervariasi, beberapa metode memiliki potongan yang lebih besar dan

beberapa memiliki potongan yang lebih kecil. Penambahan fitur dapat berupa perubahan terkecil pada layar antarmuka pengguna atau opsi kueri baru.

Pengembangan iteratif (berulang) terjadi ketika sekelompok fitur ditentukan, dirancang, dibangun, dan diuji bersama dalam serangkaian siklus, seringkali dengan durasi yang tetap. Iterasi mungkin melibatkan perubahan fitur yang dikembangkan diiterasi sebelumnya, bersama dengan perubahan dalam lingkup proyek. Setiap iterasi menghasilkan perangkat lunak yang berfungsi serta *subset* yang berkembang dari keseluruhan set fitur hingga perangkat lunak akhir dikirimkan atau pengembangannya berhenti.

Contoh - contoh yang termasuk:

- *Rational Unified Process (RUP)*: Setiap iterasi cenderung relatif lama ( misalnya, dua hingga tiga bulan) serta peningkatan fitur juga besar, seperti dua atau tiga kelompok fitur terkait
- *Scrum*: Setiap iterasi cenderung relatif singkat (misalnya, jam, hari, atau beberapa minggu), dan peningkatan fitur juga kecil, seperti beberapa peningkatan dan/atau dua hingga tiga fitur baru
- *Kanban*: Diimplementasikan dengan atau tanpa iterasi yang panjang-tetap, yang mana dapat menghasilkan satu peningkatan atau fitur setelah selesai, atau dapat mengelompokkan fitur bersama untuk dirilis sekaligus
- *Spiral*: Melibatkan pembuatan peningkatan eksperimental, beberapa diantaranya mungkin banyak dikerjakan ulang atau bahkan ditinggalkan dalam pekerjaan pengembangan selanjutnya

Komponen atau sistem yang dikembangkan menggunakan metode ini sering kali melibatkan uji tumpang tindih dan iterasi bertingkat di seluruh pengembangan. Idealnya, setiap fitur diuji pada beberapa tingkat pengujian saat bergerak menuju penyelesaiaan. Dalam beberapa kasus, para tim menggunakan pengiriman berkelanjutan atau penyebarannya berkelanjutan, yang keduanya melibatkan: otomasi signifikan dari beberapa tingkat pengujian sebagai bagian dari jalur pengiriman mereka. Banyak upaya pengembangan menggunakan metode ini juga mencakup konsep tim mengatur-diri sendiri, yang dapat mengubah cara kerja pengujian terorganisir serta hubungan antara penguji dan pengembang.

Metode ini membentuk sistem yang berkembang, yang dapat dirilis ke pengguna akhir berdasarkan fitur per fitur, iterasi per iterasi, atau dengan cara tradisional rilis yang lebih besar. Terlepas dari apakah peningkatan perangkat lunak dirilis ke pengguna akhir, pengujian regresi semakin penting sebagai pertumbuhan sistem.

Berbeda dengan model sekuensial, model berulang dan inkremental dapat membawakan perangkat lunak yang digunakan dalam beberapa minggu atau bahkan berhari-hari, tetapi hanya dapat menghasilkan set lengkap persyaratan produk selama periode bulan atau bahkan bertahun-tahun.

Untuk info lebih jelas mengenai pengujian perangkat lunak dalam konteks *Agile development*. (*lihat ISTQB-CTFL-AT, Black 2017, Crispin 2008, and Gregory 2015*).

### 2.1.2 Model Siklus Hidup Pengembangan Perangkat Lunak dalam Konteks

Model siklus hidup pengembangan perangkat lunak harus dipilih dan disesuaikan dengan konteks proyek dan karakteristik produk. Model siklus hidup pengembangan perangkat lunak yang sesuai harus dipilih dan disesuaikan berdasarkan tujuan proyek, jenis produk yang dikembangkan, prioritas bisnis (misalnya, waktu peluncuran), dan risiko produk dan proyek yang teridentifikasi. Contohnya, pengembangan dan pengujian sistem administrasi internal minor harus berbeda dari pengembangan dan pengujian sistem yang kritis terhadap keselamatan seperti sistem kontrol rem mobil. Sebagai contoh lain, dalam beberapa kasus organisasi dan masalah budaya, hal ini dapat menghambat komunikasi antara anggota tim, sehingga dapat menghambat pengembangan iteratif.

Tergantung pada konteks proyek, mungkin perlu untuk menggabungkan atau mengatur ulang tingkat pengujian dan/atau aktivitas pengujian. Contohnya, untuk integrasi produk perangkat

lunak *commercial off-the-shelf* (COTS) ke dalam sistem yang lebih besar, pembeli dapat melakukan pengujian interoperabilitas pada tingkat pengujian integrasi sistem (misalnya, integrasi ke infrastruktur dan sistem lainnya) dan pada tingkat tes penerimaan (fungsional dan

non fungsional, bersama dengan pengujian penerimaan pengguna dan pengujian penerimaan operasional). Lihat bagian 2.2 untuk diskusi dari tingkatan tes dan bagian 2.3 untuk diskusi dari tipe tipe tes.

Selain itu, model siklus hidup pengembangan perangkat lunak itu sendiri dapat digabungkan. Misalnya, model-V dapat digunakan untuk pengembangan dan pengujian sistem bagian belakang dan integrasinya, sementara model pengembangan *Agile* dapat digunakan untuk mengembangkan dan menguji antarmuka pengguna *(UI) front-end* dan kegunaannya. membuat prototipe dapat digunakan di awal proyek, dengan mengadopsi model pengembangan tambahan setelah fase percobaan selesai.

Sistem *Internet of Things* (IoT), yang terdiri dari banyak objek yang berbeda, seperti perangkat, produk, dan layanan, biasanya menerapkan model siklus hidup pengembangan perangkat lunak yang terpisah untuk setiap objek. Ini menjadikan tantangan khusus untuk pengembangan versi sistem *Internet of Things*. Selain itu siklus hidup pengembangan perangkat lunak objek tersebut menempatkan penekanan yang lebih kuat pada fase selanjutnya dari siklus ini setelah diperkenalkan ke penggunaan operasional (misalnya, mengoperasikan, memperbarui, dan fase penghentian).

Alasan mengapa model pengembangan perangkat lunak harus disesuaikan dengan konteks proyek dan produk karakteristik dapat berupa yakni:

- Perbedaan dalam sistem resiko produk (proyek kompleks atau sederhana)
- Banyak unit bisnis yang dapat menjadi bagian dari suatu proyek atau program (kombinasi dari *sequential* (berurutan) dan pengembangan *agile*)
- Waktu yang singkat untuk mengirimkan produk ke pasar (penggabungan tingkat pengujian dan/atau integrasi jenis pengujian di tingkat tes)

## 2.2 Tingkatan Tes

Tingkatan tes adalah sekelompok aktivitas tes yang diatur dan dikelola bersama. Setiap tingkat tes mempunyai contoh proses pengujian, yang terdiri dari aktivitas yang dijelaskan dalam bagian 1.4, yang dilakukan berkaitan dengan perangkat lunak pada tingkat pengembangan tertentu, dari unit atau komponen individual hingga ke sistem yang lengkap, atau jika dapat diterapkan *systems of systems (SoS)*. Tingkat pengujian berkaitan dengan aktivitas lain dalam siklus hidup pengembangan perangkat lunak. Tingkatan tes yang digunakan dalam silabus ini adalah:

- Pengujian Komponen
- Pengujian Integrasi
- Pengujian Sistem
- Pengujian Penerimaan *(Acceptance)*

Tingkatan tes dikategorikan oleh karakteristik berikut ini:

- Objek spesifik
- Dasar pengujian, direferensikan untuk menurunkan kasus pengujian
- Objek tes (yakni, apa yang sedang di uji)
- Cacat dan kegagalan tipikal/khusus
- Pendekatan dan tanggung jawab khusus

Untuk setiap tingkat pengujian, diperlukan lingkungan pengujian yang sesuai. Dalam pengujian penerimaan (acceptance), misalnya sebuah produksi lingkungan pengujian yang ideal,

sementara dalam pengujian komponen, pengembang biasanya menggunakan lingkungan pengembangan sendiri.

### 2.2.1 Pengujian Komponen

**Tujuan dari pengujian komponen**

Pengujian komponen (juga dikenal dengan pengujian unit atau modul) berfokus pada komponen komponen yang dapat diuji terpisah. Tujuan pengujian komponen meliputi:

- Mengurangi resiko
- Memverifikasi apakah perilaku fungsional dan non-fungsional dari komponen sesuai dengan yang telah didesain dan ditentukan
- Membangun kepercayaan dalam setiap kualitas komponen
- Menemukan kecacatan/kerusakan dalam komponen
- Mencegah kecacatan agar tidak lolos ke tingkat pengujian yang lebih tinggi

Dalam beberapa kasus, terutama dalam model pengembangan tambahan (*incremental*) dan iteratif (misalnya, *Agile*) di mana kode perubahan sedang berlangsung, tes regresi komponen otomatis memainkan peran kunci dalam membangun kepercayaan bahwa perubahan yang terjadi tidak merusak komponen yang ada.

Pengujian komponen sering dilakukan secara terpisah dari sistem lainnya, tergantung pada model siklus hidup pengembangan perangkat lunak dan sistemnya, yang mungkin memerlukan objek tiruan, virtualisasi layanan, pemanfaatan, potongan, dan penggerak. Pengujian komponen dapat mencakup fungsionalitas (contohnya, kebenaran dari perhitungan), karakteristik non-fungsional (misalnya, mencari kebocoran memori), dan sifat struktural (misalnya, pengujian keputusan).

**Dasar Pengujian**

Contoh produk kerja yang dapat digunakan sebagai dasar pengujian untuk pengujian komponen meliputi:

- Detail desain
- Kode
- Model data
- Spesifikasi komponen

**Objek pengujian**

Tipikal objek pengujian untuk pengujian komponen meliputi:

- Komponen - komponen, unit-unit dan modul - modul
- Kode dan struktur data
- Kelas-kelas
- Modul-modul basis data

**Tipikal cacat dan kegagalan**

Contoh dari tipikal cacat dan kegagalan untuk pengujian komponen meliputi:

- Fungsionalitas yang salah (misalnya, Tidak seperti yang dijelaskan dalam spesifikasi desain)
- Masalah aliran data
- Kode dan logika salah

Kecacatan biasanya diperbaiki segera setelah ditemukan, seringkali tanpa manajemen cacat formal. Namun, ketika pengembang melaporkan cacat, ini memberikan informasi penting untuk analisis akar masalah (*root cause analysis*) dan peningkatan proses.

**Pendekatan dan tanggung jawab khusus**

Pengujian komponen biasanya dilakukan oleh pengembang yang menulis kode, tetapi setidaknya memerlukan akses ke kode yang sedang diuji. Pengembang dapat mengganti pengembangan komponen dengan menemukan dan memperbaiki cacat. Pengembang akan sering menulis dan menjalankan tes setelah menulis kode untuk suatu komponen. Namun, khususnya dalam pengembangan *Agile*, menulis kasus uji komponen otomatis dapat mendahului penulisan kode aplikasi.

Contohnya, pertimbangkan pengembangan yang digerakkan oleh tes (*Test Driven Development - TDD*). Pengembangan yang didorong oleh pengujian sangat berulang dan berdasarkan siklus pengembangan kasus uji otomatis, kemudian membangun dan mengintegrasikan potongan-potongan kecil kode, lalu menjalankan pengujian komponen, memperbaiki masalah apa pun, dan memfaktorkan ulang kode. Proses ini berlanjut hingga komponen selesai dibuat dan semua pengujian komponen lulus. Didorong oleh tes pengembangan adalah contoh dari pendekatan uji-pertama. Sementara pengembangan yang didorong oleh pengujian berasal dari *eXtreme* Pemrograman (*eXtreme Programing - XP*), telah menyebar ke bentuk lain dari *Agile* dan juga siklus hidup berurutan (lihat ISTQB CTFL-AT).

### 2.2.2 Pengujian Integrasi

**Tujuan dari pengujian integrasi**

Pengujian integrasi berfokus pada interaksi antar komponen atau sistem. Tujuan integrasi pengujian meliputi:

- Mengurangi resiko
- Memverifikasi apakah perilaku fungsional dan non-fungsional antarmuka sesuai dengan yang dirancang dan ditentukan
- Membangun kepercayaan pada kualitas antarmuka
- Menemukan cacat (yang mungkin ada di antarmuka itu sendiri atau di dalam komponen atau sistem)
- Mencegah cacat agar tidak lolos ke tingkat pengujian yang lebih tinggi

Seperti pengujian komponen, dalam beberapa kasus, uji regresi integrasi otomatis memberikan keyakinan bahwa perubahan tidak merusak antarmuka, komponen, atau sistem yang ada.

Ada dua tingkat pengujian integrasi yang dijelaskan dalam silabus ini, yang dapat dilakukan pada objek uji dengan berbagai ukuran sebagai berikut:

- Pengujian integrasi komponen berfokus pada interaksi dan antarmuka antara komponen yang terintegrasi. Pengujian integrasi komponen dilakukan setelah pengujian komponen, dan umumnya otomatis. Dalam pengembangan iteratif dan inkremental, tes komponen integrasi biasanya merupakan bagian dari proses integrasi berkelanjutan.
- Pengujian integrasi sistem berfokus pada interaksi dan antarmuka antar sistem, paket, dan layanan mikro. Pengujian sistem integrasi juga dapat mencakup interaksi dengan, dan antarmuka yang disediakan oleh, organisasi eksternal (misalnya, layanan web). Dalam hal ini, organisasi yang

  berkembang tidak mengontrol antarmuka eksternal, yang dapat menciptakan berbagai tantangan untuk pengujian (misalnya, memastikan bahwa cacat uji-hambatan dalam kode organisasi eksternal diselesaikan, mengatur lingkungan pengujian, dll.). Pengujian sistem integrasi dapat dilakukan setelah sistem pengujian atau secara paralel dengan

aktivitas pengujian sistem yang sedang berlangsung (baik dalam pengembangan berurutan dan pengembangan iteratif dan inkremental).

**Dasar Tes**

Contoh produk kerja yang dapat digunakan sebagai dasar pengujian untuk pengujian integrasi meliputi:

- Perangkat lunak dan desain sistem
- Diagram berurutan
- Antarmuka dan Spesifikasi protokol komunikasi
- Kasus pengguna
- Arsitektur di tingkat komponen atau sistem
- Alur kerja
- Definisi antarmuka eksternal

**Objek Tes**

Tipikal objek tes untuk pengujian integrasi meliputi:

- Subsistem
- Basis data
- Infrastruktur
- Antarmuka
- *APIs*
- Layanan mikro

**Tipikal cacat dan kegagalan**

Contoh tipikal cacat dan kegagalan untuk pengujian integrasi komponen meliputi:

- Data salah, data hilang, atau penyandian data salah
- Urutan atau waktu panggilan antarmuka yang salah
- Ketidakcocokan antarmuka
- Kegagalan dalam komunikasi antar komponen
- Kegagalan komunikasi antar komponen yang tidak ditangani atau ditangani dengan tidak benar
- Asumsi yang salah tentang arti, unit, atau batas data yang dilewatkan antara komponen

Contoh tipikal cacat dan kegagalan untuk pengujian integrasi sistem meliputi:

- Struktur pesan yang tidak konsisten antar sistem
- Data salah, data hilang, atau penyandian data salah
- Ketidakcocokan antarmuka
- Kegagalan dalam komunikasi antar sistem
- Kegagalan komunikasi antar sistem yang tidak ditangani atau ditangani dengan tidak benar
- Asumsi yang salah tentang arti, unit, atau batas data yang dilewatkan antara sistem
- Kegagalan untuk mematuhi peraturan keamanan wajib

**Pendekatan dan tanggung jawab khusus**

Pengujian integrasi komponen dan pengujian integrasi sistem harus berkonsentrasi pada integrasi itu sendiri. Untuk contoh, jika mengintegrasikan modul A dengan modul B, pengujian harus fokus pada komunikasi antara modul, bukan fungsionalitas modul individual, seperti yang seharusnya dibahas selama pengujian komponen. Jika mengintegrasikan sistem X dengan sistem Y, pengujian harus fokus pada komunikasi antara sistem, bukan fungsionalitas sistem individual, seperti yang seharusnya dibahas selama pengujian sistem. Jenis uji fungsional, non-fungsional, dan struktural dapat diterapkan.

Pengujian integrasi komponen sering kali menjadi tanggung jawab pengembang. Sistem pengujian integrasi umumnya merupakan tanggung jawab penguji. Idealnya,

penguji yang melakukan pengujian sistem integrasi harus memahami arsitektur sistem, dan seharusnya mempengaruhi perencanaan integrasi.

Jika tes integrasi dan strategi integrasi direncanakan sebelum komponen atau sistem dibangun, komponen atau sistem dapat dibangun dalam urutan yang diperlukan untuk pengujian yang paling efisien. Strategi integrasi sistematis mungkin didasarkan pada arsitektur sistem (misalnya, atas-bawah dan bawah-atas), tugas-tugas fungsional, urutan pemrosesan transaksi, atau beberapa aspek lain dari sistem atau komponen. Untuk menyederhanakan isolasi cacat dan mendeteksi cacat lebih awal, integrasi

biasanya harus bertahap (yaitu, sejumlah komponen atau sistem tambahan pada suatu waktu) daripada *"big bang"* (yakni, mengintegrasikan semua komponen atau sistem dalam satu langkah). Analisis risiko dari antarmuka yang paling kompleks dapat membantu untuk memfokuskan pengujian integrasi.

Semakin besar cakupan integrasi, semakin sulit untuk mengisolasi cacat ke spesifik komponen atau sistem, yang dapat menyebabkan peningkatan risiko dan waktu tambahan untuk pemecahan masalah. Ini salah satu alasan integrasi berkelanjutan, di mana perangkat lunak terintegrasi pada komponen demi komponen dasar (yaitu, integrasi fungsional), telah menjadi praktik umum. Integrasi berkelanjutan seperti itu sering termasuk pengujian regresi otomatis, idealnya pada beberapa tingkat pengujian.

### 2.2.3 Pengujian Sistem

**Tujuan dari pengujian system**

Pengujian sistem berfokus pada perilaku dan kemampuan keseluruhan sistem atau produk, sering kali mempertimbangkan tugas ujung ke ujung (*end-to-end tasks*) yang dapat dilakukan sistem dan perilaku non-fungsional yang ditunjukkannya saat melakukan tugas-tugas itu. Tujuan dari pengujian sistem meliputi:

- Mengurangi risiko
- Memverifikasi apakah perilaku fungsional dan non-fungsional sistem sesuai dengan yang dirancang dan ditentukan
- Memvalidasi bahwa sistem telah selesai dan akan berfungsi seperti yang diharapkan
- Membangun kepercayaan pada kualitas sistem secara keseluruhan
- Menemukan cacat
- Mencegah cacat agar tidak lolos ke tingkat pengujian atau produksi yang lebih tinggi

Untuk sistem tertentu, memverifikasi kualitas data juga dapat menjadi tujuan. Seperti halnya pengujian komponen dan pengujian integrasi, dalam beberapa kasus, otomatisasi sistem pengujian regresi memberikan keyakinan bahwa perubahan belum merusak fitur yang ada atau kemampuan ujung ke ujung (*end-to-end*). Pengujian sistem seringkali menghasilkan informasi yang digunakan oleh pemangku kepentingan untuk membuat keputusan rilis. Pengujian sistem juga dapat memenuhi persyaratan hukum atau persyaratan atau standar peraturan.

Lingkungan pengujian idealnya harus sesuai dengan target akhir atau lingkungan produksi.

**Dasar tes**

Contoh produk kerja yang dapat digunakan sebagai dasar pengujian untuk pengujian sistem meliputi:

- Spesifikasi kebutuhan sistem dan perangkat lunak (fungsional dan non-fungsional)
- Laporan analisis risiko
- Kasus pengguna
- *Epics* dan cerita pengguna
- Model perilaku system
- Diagram keadaan
- Sistem dan manual pengguna

**Objek tes**

Tipikal objek uji untuk pengujian sistem meliputi:

- Aplikasi
- Sistem perangkat lunak/perangkat keras
- Sistem beroperasi
- Sistem dalam pengujian (*System Under Test - SUT*)
- Konfigurasi sistem dan konfigurasi data

**Tipikal cacat dan kegagalan**

Contoh tipikal cacat dan kegagalan untuk pengujian sistem meliputi:

- Kesalahan kalkulasi
- Perilaku fungsional atau non-fungsional sistem yang salah atau tidak terduga
- Kontrol dan/atau aliran data yang salah dalam sistem
- Kegagalan untuk melaksanakan tugas fungsional ujung ke ujung (*end-to-end*) dengan benar dan lengkap
- Kegagalan sistem untuk bekerja dengan baik di lingkungan sistem
- Kegagalan sistem untuk bekerja seperti yang dijelaskan dalam sistem dan manual pengguna

**Pendekatan dan tanggung jawab khusus**

Pengujian sistem harus fokus pada keseluruhan, perilaku ujung-ke-ujung (*end-to-end*) dari sistem secara keseluruhan, baik fungsional dan non-fungsional. Pengujian sistem harus menggunakan teknik yang paling tepat (lihat bab 4) untuk aspek dari sistem yang akan diuji. Misalnya, tabel keputusan dapat dibuat untuk memverifikasi apakah perilaku fungsional seperti yang dijelaskan dalam aturan bisnis.

Pengujian sistem biasanya dilakukan oleh penguji independen yang sangat bergantung pada spesifikasi. Cacat dalam spesifikasi (misalnya, cerita pengguna yang hilang, persyaratan bisnis yang dinyatakan secara tidak benar, dll.) dapat menyebabkan kurangnya pemahaman, atau ketidaksepakatan tentang, perilaku sistem yang diharapkan. Situasi seperti itu dapat menyebabkan positif palsu dan

negatif palsu, yang mana membuang waktu dan mengurangi efektivitas deteksi cacat dari masing-masing. Keterlibatan awal penguji dalam penyempurnaan cerita pengguna atau aktivitas pengujian statis, seperti: tinjauan, dapat membantu mengurangi kejadian situasi seperti itu.

### 2.2.4 Pengujian Penerimaan

**Tujuan dari pengujian penerimaan**

Pengujian penerimaan, seperti pengujian sistem, biasanya berfokus pada perilaku dan kemampuan keseluruhan sistem atau produk. Tujuan pengujian penerimaan meliputi:

- Membangun kepercayaan pada kualitas sistem secara keseluruhan
- Memvalidasi bahwa sistem telah selesai dan akan berfungsi seperti yang diharapkan
- Memverifikasi bahwa perilaku fungsional dan non-fungsional dari sistem adalah seperti yang ditentukan

Pengujian penerimaan dapat menghasilkan informasi untuk menilai kesiapan sistem untuk penyebaran dan digunakan oleh pelanggan (pengguna akhir - *end user*). Cacat dapat ditemukan selama pengujian penerimaan, tetapi sering menemukan cacat itu bukan tujuan, dan menemukan sejumlah besar cacat selama pengujian penerimaan dalam beberapa kasus dianggap sebagai risiko proyek utama. Pengujian penerimaan juga dapat memenuhi persyaratan hukum, peraturan atau standar.

Bentuk umum dari pengujian penerimaan meliputi:

- Pengujian penerimaan pengguna (*User Acceptance Testing - UAT*)
- Pengujian penerimaan operasional (*Operational acceptance testing - OAT*)
- Pengujian penerimaan kontraktual dan peraturan
- Pengujian alfa dan beta.

Masing-masing dijelaskan dalam empat sub bagian berikut.

**Pengujian penerimaan pengguna (*User Acceptance Testing - UAT*)**

Pengujian penerimaan pengguna dari sistem biasanya difokuskan pada validasi kesesuaian untuk penggunaan sistem oleh pengguna yang dituju dalam lingkungan

operasional yang nyata atau simulasi. Tujuan utamanya adalah membangun keyakinan bahwa pengguna dapat menggunakan sistem untuk memenuhi kebutuhan mereka, memenuhi persyaratan, dan melakukan proses bisnis dengan kesulitan, biaya, dan risiko yang minimal.

**Pengujian penerimaan operasional (*Operational Acceptance Testing - OAT*)**

Pengujian penerimaan sistem oleh pekerjaan atau staf administrasi sistem biasanya dilakukan dalam lingkungan produksi (simulasi). Tes berfokus pada aspek operasional, dan mungkin termasuk:

- Pengujian pencadangan dan pemulihan
- Memasang, menghapus, dan meningkatkan
- Pemulihan bencana

- Manajemen pengguna
- Pekerjaan pemeliharaan
- Pemuatan data dan tugas migrasi
- Memeriksa kerentanan keamanan
- Pengujian kinerja

Tujuan utama dari pengujian penerimaan operasional adalah membangun kepercayaan bahwa operator atau sistem administrator dapat menjaga sistem bekerja dengan baik untuk pengguna di lingkungan operasional, bahkan dalam kondisi luar biasa atau sulit.

**Pengujian penerimaan kontraktual dan peraturan**

Pengujian penerimaan kontraktual dilakukan terhadap kriteria penerimaan kontrak untuk memproduksi perangkat lunak yang dikembangkan khusus. Kriteria penerimaan harus ditentukan ketika para pihak menyetujui kontrak. Pengujian penerimaan kontraktual sering dilakukan oleh pengguna atau penguji independen.

Pengujian penerimaan peraturan dilakukan terhadap peraturan apa pun yang harus dipatuhi, seperti peraturan pemerintah, hukum, atau keselamatan. Pengujian penerimaan peraturan sering dilakukan oleh pengguna atau oleh penguji independen, terkadang dengan hasil yang disaksikan atau diaudit oleh badan pengatur.

Tujuan utama pengujian penerimaan kontraktual dan peraturan adalah membangun kepercayaan bahwa kepatuhan kontrak atau peraturan telah tercapai.

**Pengujian alfa dan beta**

Pengujian alfa dan beta biasanya digunakan oleh pengembang perangkat lunak *commercial off-the-shelf (COTS)* yang ingin mendapatkan umpan balik dari pengguna, pelanggan, dan/atau operator potensial atau yang sudah ada sebelum perangkat lunak produk ditempatkan di pasar. Pengujian alfa dilakukan di situs organisasi yang sedang berkembang, bukan oleh tim pengembangan, tetapi oleh calon pelanggan atau pelanggan yang sudah ada, dan/atau operator atau tim uji independen. Pengujian beta dilakukan oleh calon pelanggan atau pelanggan yang sudah ada, dan/atau operator di lokasi mereka sendiri. Pengujian beta dapat dilakukan setelah pengujian alfa, atau dapat terjadi tanpa pengujian alfa sebelumnya.

Salah satu tujuan pengujian alfa dan beta adalah membangun kepercayaan di antara calon pelanggan atau pelanggan yang sudah ada, dan/atau operator bahwa mereka dapat menggunakan sistem dalam kondisi normal, sehari-hari, dan dalam operasional lingkungan untuk mencapai tujuan mereka dengan kesulitan minimum, biaya, dan risiko. Tujuan lain mungkin menjadi deteksi cacat yang terkait dengan kondisi dan lingkungan di mana sistem akan digunakan, terutama ketika kondisi dan lingkungan tersebut sulit untuk ditiru oleh tim pengembangan.

**Dasar Tes**

Contoh produk kerja yang dapat digunakan sebagai dasar pengujian untuk segala bentuk pengujian penerimaan meliputi:

- Proses bisnis
- Persyaratan pengguna atau bisnis
- Peraturan, kontrak hukum dan standar
- Kasus pengguna dan atau cerita pengguna
- Persyaratan sistem
- Dokumentasi sistem atau pengguna

- Prosedur instalasi
- Laporan analisis resiko

Selain itu, sebagai dasar pengujian untuk menurunkan kasus uji untuk pengujian penerimaan operasional, berikut satu atau lebih dari produk kerja yang dapat digunakan:

- Prosedur pencadangan dan pemulihan
- Prosedur pemulihan bencana
- Persyaratan non-fungsional
- Dokumentasi pekerjaan
- Petunjuk pemasangan dan penyebaran
- Target performa
- Paket basis data
- Standar atau peraturan keamanan

**Tipikal objek tes**

Tipikal objek untuk segala bentuk pengujian penerimaan meliputi:

- Sistem sedang diuji
- Konfigurasi sistem dan data konfigurasi
- Proses bisnis untuk sistem yang terintegrasi penuh
- Sistem pemulihan dan situs populer (untuk kelangsungan bisnis dan pengujian pemulihan bencana)
- Proses operasional dan pemeliharaan
- Formulir
- Laporan
- Data produksi yang ada dan yang dikonversi

**Tipikal cacat dan kegagalan**

Contoh tipikal kecacatan untuk segala bentuk pengujian penerimaan meliputi:

- Alur kerja sistem tidak memenuhi kebutuhan bisnis atau pengguna
- Aturan bisnis tidak diterapkan dengan benar
- Sistem tidak memenuhi persyaratan kontraktual atau peraturan
- Kegagalan non-fungsional seperti kerentanan keamanan, efisiensi kinerja yang tidak memadai di bawah beban tinggi, atau operasi yang tidak tepat pada platform yang didukung

**Pendekatan dan tanggung jawab khusus**

Pengujian penerimaan sering kali menjadi tanggung jawab pelanggan, pengguna bisnis, pemilik produk, atau operator sistem, dan pemangku kepentingan lainnya yang mungkin terlibat juga.

Pengujian penerimaan sering dianggap sebagai tingkat pengujian terakhir dalam siklus hidup pengembangan berurutan, tetapi itu dapat juga terjadi pada waktu lain, misalnya:

- Pengujian penerimaan produk perangkat lunak *COTS (commercial off-the-shelf)* dapat terjadi saat pemasangan atau terintegrasi
- Pengujian penerimaan dari peningkatan fungsional baru dapat terjadi sebelum pengujian sistem

Dalam pengembangan berulang, tim proyek dapat menggunakan berbagai bentuk pengujian penerimaan selama dan diakhir setiap iterasi, seperti yang berfokus pada verifikasi fitur baru terhadap kriteria penerimaannya dan yang berfokus pada validasi bahwa fitur baru memenuhi kebutuhan pengguna. Selain itu, tes alfa dan tes beta dapat terjadi, baik pada akhir setiap iterasi, setelah selesainya setiap iterasi, atau setelah serangkaian dari iterasi. Tes penerimaan pengguna, tes penerimaan operasional, tes penerimaan peraturan, dan tes penerimaan kontraktual juga dapat terjadi, baik pada penutupan setiap iterasi, setelah selesainya setiap iterasi, atau setelah serangkaian iterasi.

## 2.3 Jenis - jenis Tes

Jenis pengujian adalah sekelompok aktivitas pengujian yang ditujukan untuk menguji karakteristik khusus dari sistem perangkat lunak, atau bagian dari sistem, berdasarkan tujuan pengujian tertentu. Tujuan tersebut dapat mencakup:

- Mengevaluasi karakteristik kualitas fungsional, seperti kelengkapan, kebenaran, dan kelayakan
- Mengevaluasi karakteristik kualitas non-fungsional, seperti keandalan, efisiensi kinerja, keamanan, kompatibilitas, dan kegunaan
- Mengevaluasi apakah struktur atau arsitektur komponen atau sistem sudah benar, lengkap, dan seperti yang ditentukan
- Mengevaluasi efek perubahan, seperti memastikan bahwa cacat telah diperbaiki (pengujian konfirmasi) dan mencari perubahan yang tidak diinginkan dalam perilaku yang dihasilkan dari perangkat lunak atau lingkungan perubahan (pengujian regresi)

### 2.3.1 Pengujian Fungsional

Pengujian fungsional dari sebuah sistem melibatkan pengujian yang mengevaluasi fungsi yang harus dilakukan sistem. Persyaratan fungsional dapat dijelaskan dalam produk kerja seperti spesifikasi persyaratan bisnis, epik, cerita pengguna, *kasus pengguna*, atau spesifikasi fungsional, atau mungkin tidak didokumentasikan. Fungsinya adalah "apa" yang harus dilakukan sistem.

Pengujian fungsional harus dilakukan pada semua tingkat pengujian (misalnya, pengujian untuk komponen mungkin didasarkan pada spesifikasi komponen), meskipun fokusnya berbeda pada setiap tingkat (lihat bagian 2.2).

Pengujian fungsional mempertimbangkan perilaku perangkat lunak, sehingga teknik kotak-hitam dapat digunakan untuk menurunkan kondisi pengujian dan kasus uji untuk fungsionalitas komponen atau sistem (lihat bagian 4.2).

Ketelitian pengujian fungsional dapat diukur melalui cakupan fungsional. Cakupan fungsional adalah sejauh mana beberapa fungsi telah dilakukan oleh tes, dan dinyatakan sebagai persentase jenis elemen yang dicakup. Misalnya, menggunakan ketertelusuran antara pengujian dan persyaratan fungsional, persentase persyaratan yang ditangani dengan pengujian dapat dihitung, berpotensi mengidentifikasi kesenjangan cakupan.

Desain dan pelaksanaan tes fungsional mungkin melibatkan keterampilan atau pengetahuan khusus, seperti pengetahuan tentang masalah bisnis tertentu yang dipecahkan oleh perangkat lunak (misalnya, perangkat lunak pemodelan geocatatani untuk minyak dan gas industri).

### 2.3.2 Pengujian Non-fungsional

Pengujian non-fungsional dari suatu sistem mengevaluasi karakteristik sistem dan perangkat lunak seperti kegunaan, efisiensi kinerja atau keamanan. Lihat standar ISO (ISO/IEC 25010) untuk klasifikasi perangkat lunak karakteristik kualitas produk. Pengujian non-fungsional adalah pengujian "seberapa baik" sistem berperilaku.

Berlawanan dengan kesalahan persepsi umum, pengujian non-fungsional dapat dan sering kali harus dilakukan pada semua tingkat pengujian, dan dilakukan sedini mungkin. Penemuan cacat non-fungsional yang terlambat bisa sangat berbahaya bagi keberhasilan suatu proyek.

Teknik kotak-hitam (lihat bagian 4.2) dapat digunakan untuk menurunkan kondisi pengujian dan kasus uji untuk pengujian nonfungsional. Misalnya, analisis nilai batas dapat digunakan untuk menentukan kondisi tegangan untuk tes kinerja.

Ketelitian pengujian non-fungsional dapat diukur melalui cakupan non-fungsional. Cakupan nonfungsional adalah sejauh mana beberapa jenis elemen non-fungsional telah dilaksanakan oleh pengujian, dan dinyatakan sebagai persentase dari jenis elemen yang dicakup. Misalnya, menggunakan ketelusuran antara pengujian dan perangkat yang didukung untuk aplikasi seluler, persentase perangkat yang ditangani oleh pengujian kompatibilitas dapat dihitung, berpotensi mengidentifikasi kesenjangan cakupan.

Desain dan pelaksanaan tes non-fungsional mungkin melibatkan keterampilan atau pengetahuan khusus, seperti pengetahuan tentang kelemahan yang melekat pada desain atau teknologi (misalnya, kerentanan keamanan yang terkait dengan bahasa pemrograman) atau basis pengguna tertentu (misalnya, persona pengguna fasilitas kesehatan sistem manajemen).

Lihat ISTQB-CTAL-TA, ISTQB-CTAL-TTA, ISTQB-CTAL-SEC, dan modul spesialis ISTQB® lainnya untuk rincian lebih lanjut mengenai pengujian karakteristik kualitas non-fungsional.

### 2.3.3 Pengujian Kotak Putih

Pengujian kotak putih memperoleh pengujian berdasarkan struktur atau implementasi internal sistem. Struktur internal dapat mencakup kode, arsitektur, alur kerja, dan/atau arus data dalam sistem (lihat bagian 4.3).

Ketelitian pengujian kotak putih dapat diukur melalui cakupan struktural. Cakupan struktural adalah sejauh mana beberapa jenis elemen struktur telah dilakukan pengujian, dan dinyatakan sebagai persentase jenis elemen yang dicakup.

Pada tingkat pengujian komponen, cakupan kode didasarkan pada persentase kode komponen yang telah diuji, dan dapat diukur dari segi aspek kode yang berbeda (item cakupan) seperti persentase pernyataan yang dapat dieksekusi diuji dalam komponen, atau persentase dari hasil keputusan yang diuji. Jenis cakupan ini secara kolektif disebut cakupan kode. Pada integrasi komponen tingkat pengujian, pengujian kotak putih mungkin didasarkan pada arsitektur sistem, seperti antarmuka antar komponen, dan cakupan struktural dapat diukur dalam persentase antarmuka yang dilakukan dengan tes.

Desain dan pelaksanaan tes kotak putih mungkin melibatkan keterampilan atau pengetahuan khusus, seperti cara kodenya dibangun, bagaimana data disimpan (misalnya, untuk mengevaluasi kemungkinan *queries* basis data), dan bagaimana menggunakan peralatan cakupan dan untuk menginterpretasikan hasil mereka dengan benar.

### 2.3.4 Pengujian terkait perubahan

Ketika perubahan dilakukan pada sistem, baik untuk memperbaiki cacat atau karena baru atau berubah fungsionalitas, pengujian harus dilakukan untuk mengonfirmasi bahwa perubahan telah memperbaiki cacat atau mengimplementasikan fungsionalitas dengan benar, dan tidak menyebabkan konsekuensi merugikan yang tidak terduga.

- Pengujian konfirmasi: Setelah cacat diperbaiki, perangkat lunak dapat diuji dengan semua kasus uji yang gagal karena cacat, yang harus dijalankan kembali pada versi perangkat lunak yang baru. Perangkat lunak juga dapat diuji dengan tes baru untuk menutupi perubahan yang diperlukan untuk memperbaiki cacat. Setidaknya, langkah-langkah untuk me-reproduksi kegagalan yang disebabkan oleh cacat harus dijalankan kembali pada perangkat lunak versi baru. Tujuan dari tes konfirmasi adalah untuk mengkonfirmasi apakah cacat asli telah berhasil diperbaiki.

- Pengujian regresi: Ada kemungkinan bahwa perubahan dibuat di salah satu bagian dari kode, apakah perbaikan atau jenis perubahan lain, mungkin secara tidak sengaja mempengaruhi perilaku bagian lain dari kode, baik dalam komponen yang sama, dalam komponen lain dari sistem yang sama, atau bahkan dalam sistem lain. Perubahan dapat mencakup perubahan pada lingkungan, seperti versi baru dari sistem operasi atau sistem manajemen basis data. Efek samping yang tidak diinginkan seperti itu disebut regresi. Pengujian regresi melibatkan pengujian yang berjalan untuk mendeteksi efek samping yang tidak diinginkan tersebut.

Pengujian konfirmasi dan pengujian regresi dilakukan di semua tingkat pengujian.
Terutama dalam siklus hidup pengembangan berulang dan inkremental (misalnya, Agile), fitur baru, perubahan pada fitur yang ada, dan pemfaktoran ulang kode menghasilkan perubahan yang sering pada kode, yang juga memerlukan pengujian terkait perubahan. Karena sifat sistem yang berkembang, konfirmasi dan pengujian regresi sangat penting. Ini sangat relevan untuk sistem ***Internet of Things*** di mana objek individual (misalnya, perangkat) sering diperbarui atau diganti.

Rangkaian uji regresi dijalankan berkali-kali dan umumnya berkembang perlahan, sehingga pengujian regresi merupakan kandidat kuat untuk otomatisasi. Otomatisasi pengujian ini harus dimulai sejak awal proyek (lihat bab 6).

### 2.3.5 Tipe dan Tingkat Tes

Dimungkinkan untuk melakukan salah satu jenis pengujian yang disebutkan di atas pada setiap tingkat pengujian. Sebagai ilustrasi, contoh pengujian fungsional, non-fungsional, kotak putih, dan terkait perubahan akan diberikan di semua tingkat pengujian, untuk aplikasi perbankan, dimulai dengan pengujian fungsional:

• Untuk pengujian komponen, pengujian dirancang berdasarkan cara komponen menghitung bunga majemuk.

• Untuk pengujian integrasi komponen, pengujian dirancang berdasarkan bagaimana informasi akun yang diambil pada antarmuka pengguna diteruskan ke catatanika bisnis.

• Untuk pengujian sistem, pengujian dirancang berdasarkan bagaimana pemegang rekening dapat mengajukan permohonan kredit pada rekening giro mereka.

• Untuk pengujian integrasi sistem, pengujian dirancang berdasarkan cara sistem menggunakan layanan mikro eksternal untuk memeriksa skor kredit pemegang akun.

• Untuk pengujian penerimaan, pengujian dirancang berdasarkan bagaimana bankir menangani persetujuan atau penolakan aplikasi kredit.

Berikut ini adalah contoh pengujian nonfungsional:

- Untuk pengujian komponen, pengujian kinerja dirancang untuk mengevaluasi jumlah siklus CPU yang diperlukan untuk melakukan penghitungan total bunga yang Kompleks.
- Untuk pengujian integrasi komponen, pengujian keamanan dirancang untuk kerentanan ***buffer overflow*** karena data yang diteruskan dari antarmuka pengguna ke catatanika bisnis.
- Untuk pengujian sistem, pengujian portabilitas dirancang untuk memeriksa apakah susunan presentasi bekerja pada semua browser dan perangkat seluler yang didukung.
- Untuk pengujian integrasi sistem, pengujian keandalan dirancang untuk mengevaluasi ketahanan sistem jika layanan mikro skor kredit gagal merespons.
- Untuk pengujian penerimaan, uji kegunaan dirancang untuk mengevaluasi aksesibilitas antarmuka pemrosesan kredit bankir bagi penyandang disabilitas.

Berikut ini adalah contoh pengujian kotak putih:

- Untuk pengujian komponen, pengujian dirancang untuk mencapai cakupan pernyataan dan keputusan yang lengkap (lihat bagian 4.3) untuk semua komponen yang melakukan perhitungan finansial.
- Untuk pengujian integrasi komponen, pengujian dirancang untuk melatih bagaimana setiap layar di antarmuka *browser* meneruskan data ke layar berikutnya dan ke catatanika bisnis.
- Untuk pengujian sistem, pengujian dirancang untuk mencakup urutan halaman web yang dapat terjadi selama aplikasi kredit.
- Untuk pengujian integrasi sistem, pengujian dirancang untuk menggunakan semua jenis pertanyaan yang mungkin dikirim ke layanan mikro skor kredit.
- Untuk pengujian penerimaan, pengujian dirancang untuk mencakup semua struktur file data keuangan yang didukung dan rentang nilai untuk transfer antar bank.

Akhirnya, berikut ini adalah contoh untuk tes terkait perubahan:

- Untuk pengujian komponen, pengujian regresi otomatis dibuat untuk setiap komponen dan disertakan dalam kerangka integrasi berkelanjutan.
- Untuk pengujian integrasi komponen, pengujian dirancang untuk mengonfirmasi perbaikan pada cacat terkait antarmuka saat perbaikan diperiksa ke dalam repositori kode.
- Untuk pengujian sistem, semua pengujian untuk alur kerja tertentu akan dijalankan kembali jika layar pada alur kerja tersebut berubah.
- Untuk pengujian integrasi sistem, pengujian aplikasi yang berinteraksi dengan layanan mikro penilaian kredit dijalankan ulang setiap hari sebagai bagian dari penerapan layanan mikro yang berkelanjutan.
- Untuk pengujian penerimaan, semua pengujian yang sebelumnya gagal dijalankan kembali setelah cacat yang ditemukan dalam pengujian penerimaan diperbaiki.

Meskipun bagian ini memberikan contoh setiap jenis pengujian di setiap tingkat, tidak perlu, untuk semua perangkat lunak, agar setiap jenis pengujian terwakili di setiap tingkat. Namun,

penting untuk dijalankan jenis pengujian yang berlaku di setiap tingkat, terutama tingkat paling awal di mana jenis pengujian tersebut terjadi.

## 2.4 Pengujian Pemeliharaan

Setelah digunakan untuk lingkungan produksi, perangkat lunak dan sistem perlu dipelihara. Perubahan dari berbagai macam hampir tak terelakkan dalam perangkat lunak dan sistem yang dikirimkan, baik untuk memperbaiki cacat yang ditemukan dalam penggunaan operasional, untuk menambah fungsionalitas baru, atau untuk menghapus atau mengubah fungsionalitas yang sudah disampaikan. Pemeliharaan juga diperlukan untuk menjaga atau meningkatkan karakteristik kualitas non-fungsional komponen atau sistem selama masa pakainya, terutama efisiensi kinerja, kompatibilitas, keandalan, keamanan, dan portabilitas.

Ketika ada perubahan yang dibuat sebagai bagian dari pemeliharaan, pengujian pemeliharaan harus dilakukan, baik untuk mengevaluasi keberhasilan perubahan yang dibuat dan untuk memeriksa kemungkinan efek samping (misalnya, regresi)
di bagian sistem yang tetap tidak berubah (yang biasanya sebagian besar sistem).
Pemeliharaan dapat melibatkan rilis yang direncanakan dan rilis yang tidak direncanakan (perbaikan terbaru).

Rilis pemeliharaan mungkin memerlukan pengujian pemeliharaan di berbagai tingkat pengujian, menggunakan berbagai jenis pengujian, berdasarkan cakupannya. Ruang lingkup pengujian pemeliharaan tergantung pada:

- Tingkat risiko perubahan, misalnya, sejauh mana area yang diubah dari perangkat lunak berkomunikasi dengan komponen atau sistem lain
- Besarnya sistem yang ada
- Besar kecilnya perubahan

### 2.4.1 Pemicu untuk Pemeliharaan

Ada beberapa alasan mengapa pemeliharaan perangkat lunak, dan dengan demikian pengujian pemeliharaan, dilakukan, baik: untuk perubahan yang direncanakan dan tidak direncanakan.

Kami dapat mengklasifikasikan pemicu untuk pemeliharaan sebagai berikut:

- Modifikasi, seperti peningkatan yang direncanakan (misalnya, berbasis rilis), perubahan korektif dan darurat, perubahan lingkungan operasional (seperti sistem operasi yang direncanakan atau upgrade database), upgrade perangkat lunak COTS, dan tambalan untuk cacat dan kerentanan
- Migrasi, seperti dari satu platform ke platform lain, yang dapat memerlukan pengujian operasional dari lingkungan baru serta perangkat lunak yang diubah, atau pengujian konversi data ketika data dari aplikasi lain akan dipindahkan ke sistem yang dipelihara
    - o Pensiun, seperti saat aplikasi mencapai akhir masa pakainya. Saat aplikasi atau sistem dihentikan, ini dapat memerlukan pengujian migrasi data atau pengarsipan jika periode penyimpanan data yang lama diperlukan.
    - o Pengujian prosedur pemulihan/pengambilan kembali setelah pengarsipan untuk periode penyimpanan yang lama mungkin juga diperlukan.
    - o pengujian regresi mungkin diperlukan untuk memastikan bahwa fungsionalitas apa pun yang tersisa dalam layanan masih berfungsi.

Untuk sistem ***Internet of Things***, pengujian pemeliharaan dapat dipicu oleh pengenalan hal-hal yang benar-benar baru atau yang dimodifikasi, seperti perangkat keras dan layanan perangkat lunak, ke dalam sistem secara keseluruhan. Itu pengujian pemeliharaan untuk sistem tersebut menempatkan penekanan khusus pada pengujian integrasi pada tingkat yang berbeda (misalnya, tingkat jaringan, tingkat aplikasi) dan pada aspek keamanan, khususnya yang berkaitan dengan pribadi data.

### 2.4.2 Analisis Dampak untuk Pemeliharaan

Analisis dampak mengevaluasi perubahan yang dibuat untuk rilis pemeliharaan untuk mengidentifikasi konsekuensi yang diinginkan serta efek samping yang diharapkan dan mungkin terjadi dari perubahan, dan untuk mengidentifikasi area dalam sistem yang akan terpengaruh oleh perubahan. Analisis dampak juga dapat membantu untuk mengidentifikasi dampak perubahan pada pengujian yang ada.

Efek samping dan area yang terpengaruh dalam sistem perlu diuji untuk regresi, mungkin setelah memperbarui semua pengujian yang ada yang terpengaruh oleh perubahan.

Analisis dampak dapat dilakukan sebelum perubahan dilakukan, untuk membantu memutuskan apakah perubahan harus dilakukan, berdasarkan potensi konsekuensi di area lain dari sistem.

Analisis dampak bisa menjadi sulit jika:

- Spesifikasi (misalnya, persyaratan bisnis, cerita pengguna, arsitektur) sudah ketinggalan zaman atau tidak ada
- Kasus uji tidak didokumentasikan atau ketinggalan zaman
- Ketertelusuran dua arah antara pengujian dan basis pengujian tidak dipertahankan
- Dukungan peralatan lemah atau tidak ada
- Orang yang terlibat tidak memiliki pengetahuan domain dan/atau sistem
- Perhatian yang kurang telah diberikan pada pemeliharaan perangkat lunak selama pengembangan

Sertifikat Yayasan yang ada dalam Pengujian Perangkat Lunak (misalnya, dari ISEB, ASQF atau dewan anggota yang diakui ISTQB) yang diberikan sebelum Sertifikat Internasional ini dirilis, akan dianggap setara dengan Sertifikat Internasional. Sertifikat Foundation tidak kedaluwarsa dan tidak perlu diperpanjang. Tanggal pemberiannya tertera pada Sertifikat.

# 3 Pengujian Statis — 135 menit

**Kata kunci**

Tinjauan *ad hoc*, tinjauan berbasis daftar periksa, pengujian dinamis, tinjauan formal, tinjauan informal, pemeriksaan, membaca berbasis perspektif, tinjauan, tinjauan berbasis peran, tinjauan berbasis skenario, analisis statis, statis pengujian, tinjauan teknis, penelusuran

**Tujuan Pembelajaran untuk Pengujian Statis**

### 3.1 Dasar Pengujian Statis

FL-3.1.1 (K1) Mengenali jenis produk kerja perangkat lunak yang dapat diperiksa oleh berbagai statis teknik pengujian

FL-3.1.2 (K2) Gunakan contoh untuk mendenaskahsikan nilai pengujian statis

FL-3.1.3 (K2) Jelaskan perbedaan antara teknik statis dan dinamis, dengan mempertimbangkan tujuan,jenis cacat yang harus diidentifikasi, dan peran teknik ini dalam siklus hidup perangkat lunak

### 3.2 Proses Peninjauan

FL-3.2.1 (K2) Meringkas aktivitas proses peninjauan produk kerja

FL-3.2.2 (K1) Mengenali peran dan tanggung jawab yang berbeda dalam tinjauan formal

FL-3.2.3 (K2) Jelaskan perbedaan antara berbagai jenis tinjauan: tinjauan informal,panduan, tinjauan teknis, dan inspeksi

FL-3.2.4 (K3) Menerapkan teknik peninjauan ke produk kerja untuk menemukan cacat

FL-3.2.5 (K2) Jelaskan faktor-faktor yang berkontribusi terhadap keberhasilan tinjauan

# 3.1 Dasar - dasar Pengujian Statis

Berbeda dengan pengujian dinamis, yang membutuhkan eksekusi perangkat lunak yang diuji, pengujian statis bergantung pada pemeriksaan manual terhadap produk kerja (yaitu, tinjauan) atau evaluasi kode yang digerakkan oleh peralatan atau produk kerja lainnya (yaitu, analisis statis). Kedua jenis pengujian statis ini menilai kode atau pekerjaan lainnya produk yang sedang diuji tanpa benar-benar mengeksekusi kode atau produk kerja yang sedang diuji.

Analisis statis penting untuk sistem komputer yang kritis terhadap keselamatan (misalnya, penerbangan, medis, atau nuklir perangkat lunak), tetapi analisis statis juga menjadi penting dan umum di pengaturan lain. Sebagai contoh, analisis statis merupakan bagian penting dari pengujian keamanan. Analisis statis juga sering dimasukkan ke dalam pembuatan perangkat lunak dan peralatan distribusi otomatis, misalnya dalam pengembangan Agile, pengiriman berkelanjutan, dan penyebaran yang berkelanjutan.

### 3.1.1 Produk Kerja yang Dapat Diperiksa dengan Pengujian Statis

Hampir semua produk kerja dapat diperiksa menggunakan pengujian statis (tinjauan dan/atau analisis statis), misalnya:

- Spesifikasi, termasuk persyaratan bisnis, persyaratan fungsional, dan persyarata Keamanan
- Epik, cerita pengguna, dan kriteria penerimaan
- Spesifikasi arsitektur dan desain
- Kode
- Perangkat pengujian, termasuk rencana pengujian, kasus pengujian, prosedur pengujian, dan naskah pengujian otomatis
- Panduan pengguna
- Halaman web
- Kontrak, rencana proyek, jadwal, dan perencanaan anggaran
- Penyiapan konfigurasi dan penyiapan infrastruktur
- Model, seperti diagram aktivitas, yang dapat digunakan untuk pengujian Berbasis Model (lihat ISTQB CTFL-MBT dan Kramer 2016)

Tinjauan dapat diterapkan pada produk kerja apa pun yang dapat dibaca dan dipahami oleh peserta. Analisis statis dapat diterapkan secara efisien untuk setiap produk kerja dengan struktur formal (biasanya kode atau model) yang memiliki peralatan analisis statis yang sesuai. Analisis statis bahkan dapat diterapkan dengan peralatan yang mengevaluasi produk kerja yang ditulis dalam bahasa alami seperti persyaratan (misalnya, memeriksa ejaan, tata bahasa, dan keterbacaan).

### 3.1.2 Manfaat Pengujian Statis

Teknik pengujian statis memberikan berbagai manfaat. Ketika diterapkan di awal pengembangan perangkat lunak siklus hidup, pengujian statis memungkinkan deteksi dini cacat sebelum pengujian dinamis dilakukan (misalnya, persyaratan atau tinjauan spesifikasi desain, penyempurnaan backcatatan, dll.). Cacat yang ditemukan lebih awal seringkali jauh lebih murah untuk dihilangkan daripada cacat yang ditemukan kemudian dalam siklus hidup, terutama dibandingkan dengan cacat yang ditemukan setelah perangkat lunak dikerahkan dan digunakan secara aktif. Menggunakan teknik pengujian statis untuk menemukan cacat dan kemudian memperbaiki cacat tersebut segera hampir selalu jauh lebih murah bagi organisasi daripada menggunakan dinamis pengujian untuk menemukan cacat pada objek uji dan kemudian memperbaikinya, terutama ketika mempertimbangkan tambahan anggaran yang terkait dengan memperbarui produk kerja lainnya dan melakukan konfirmasi dan pengujian regresi.

Manfaat tambahan pengujian statis dapat mencakup:

- Mendeteksi dan memperbaiki cacat secara lebih efisien, dan sebelum eksekusi pengujian dinamis
- Mengidentifikasi cacat yang tidak mudah ditemukan dengan pengujian dinamis
- Mencegah cacat dalam desain atau pengkodean dengan mengungkap inkonsistensi, ambiguitas, kontradiksi, kelalaian, ketidakakuratan, dan redundansi dalam persyaratan
- Meningkatkan produktivitas pengembangan (misalnya, karena desain yang lebih baik, kode yang lebih dapat dipelihara)
- Mengurangi biaya dan waktu pengembangan
- Mengurangi biaya dan waktu pengujian
- Mengurangi total biaya kualitas selama masa pakai perangkat lunak, karena lebih sedikit kegagalan di kemudian hari siklus hidup atau setelah pengiriman ke dalam operasi
- Meningkatkan komunikasi antar anggota tim selama berpartisipasi dalam tinjauan

### 3.1.3 Perbedaan antara Pengujian Statis dan Dinamis

Pengujian statis dan pengujian dinamis dapat memiliki tujuan yang sama (lihat bagian 1.1.1), seperti memberikan penilaian kualitas produk kerja dan mengidentifikasi cacat sedini mungkin. statis dan pengujian dinamis saling melengkapi dengan menemukan berbagai jenis cacat. Satu perbedaan utama adalah bahwa pengujian statis menemukan cacat pada produk kerja secara langsung daripada mengidentifikasi kegagalan yang disebabkan oleh cacat saat perangkat lunak dijalankan. Cacat dapat berada dalam produk kerja untuk waktu yang sangat lama tanpa menyebabkan kegagalan. Jalur di mana cacat terletak mungkin jarang dilakukan atau sulit dijangkau, sehingga tidak akan mudah untuk membangun dan menjalankan tes dinamis yang menghadapinya. Pengujian statis mungkin dapat menemukan cacat dengan sedikit usaha. Perbedaan lainnya adalah bahwa pengujian statis dapat digunakan untuk meningkatkan konsistensi dan kualitas internal produk kerja, sedangkan pengujian dinamis biasanya berfokus pada perilaku yang terlihat secara eksternal. Dibandingkan dengan pengujian dinamis, cacat umum yang lebih mudah dan lebih murah untuk ditemukan dan diperbaiki melalui pengujian statis meliputi:

- Cacat persyaratan (misalnya, inkonsistensi, ambiguitas, kontradiksi, kelalaian, ketidakakuratan,dan redundansi)
- Cacat desain (misalnya, algoritma atau struktur database yang tidak efisien, kopling tinggi, kohesi rendah)
- Cacat pengkodean (misalnya, variabel dengan nilai yang tidak ditentukan, variabel yang dideklarasikan tetapi tidak pernah digunakan, kode yang tidak dapat dijangkau, kode duplikat)
- Penyimpangan dari standar (misalnya, kurangnya kepatuhan terhadap standar pengkodean)
- Spesifikasi antarmuka salah (misalnya, unit pengukuran berbeda yang digunakan oleh sistem pemanggil daripada sistem yang dipanggil)
- Kerentanan keamanan (misalnya, kerentanan terhadap ***buffer overflows***)
- Kesenjangan atau ketidakakuratan dalam ketertelusuran atau cakupan basis pengujian (misalnya, tes yang hilang untuk kriteria penerimaan)

Selain itu, sebagian besar jenis kurangnya rawatan hanya dapat ditemukan dengan pengujian statis (misalnya, modularisasi yang tidak tepat, penggunaan kembali komponen yang buruk, kode yang sulit untuk dianalisis dan dimodifikasi tanpa memperkenalkan cacat baru).

## 3.2 Proses Peninjauan

Tinjauan bervariasi dari informal ke formal. Tinjauan informal dicirikan dengan tidak mengikuti proses yang ditentukan dan tidak memiliki keluaran yang didokumentasikan secara formal. Tinjauan formal ditandai oleh tim partisipasi, hasil tinjauan yang terdokumentasi, dan prosedur terdokumentasi untuk melakukan tinjauan. Formalitas proses peninjauan terkait dengan faktor-faktor seperti model siklus hidup pengembangan perangkat lunak, kematangan proses pengembangan, kompleksitas produk kerja yang akan ditinjau, persyaratan hukum atau peraturan, dan/atau kebutuhan akan jejak audit.

Fokus tinjauan tergantung pada tujuan tinjauan yang disepakati (misalnya, menemukan kekurangan, memperoleh pemahaman, mendidik peserta seperti penguji dan anggota tim baru, atau berdiskusi dan memutuskan melalui persetujuan umum).

Standar ISO (ISO/IEC 20246) berisi denaskahsi yang lebih mendalam tentang proses tinjauan produk kerja, termasuk peran dan teknik tinjauan.

### 3.2.1 Proses Tinjauan Produk Kerja

Proses peninjauan terdiri dari aktivitas utama berikut:

**Perencanaan**

- Menentukan ruang lingkup, yang meliputi tujuan tinjauan, dokumen atau bagian apa dokumen untuk ditinjau, dan karakteristik kualitas yang akan dievaluasi
- Memperkirakan usaha dan kerangka waktu
- Mengidentifikasi karakteristik tinjauan seperti jenis tinjauan dengan peran, aktivitas, dan daftar periksa
- Memilih orang untuk berpartisipasi dalam peninjauan dan mengalokasikan peran
- Menentukan kriteria masuk dan keluar untuk jenis tinjauan yang lebih formal (misalnya, inspeksi)
- Memeriksa apakah kriteria entri terpenuhi (untuk jenis tinjauan yang lebih formal)

**Mulai tinjauan**

- Mendistribusikan hasil kerja (secara fisik atau elektronik) dan materi lainnya, seperti: mengeluarkan formulir catatan, daftar periksa, dan produk kerja terkait
- Menjelaskan ruang lingkup, tujuan, proses, peran, dan produk kerja kepada peserta
- Menjawab pertanyaan apa pun yang mungkin dimiliki peserta tentang tinjauan

**Tinjauan individu (yaitu, persiapan individu)**

- Meninjau semua atau sebagian dari produk kerja
- Mencatat potensi cacat, rekomendasi, dan pertanyaan

**Masalah komunikasi dan analisis**

- Mengkomunikasikan potensi cacat yang teridentifikasi (misalnya, dalam rapat peninjauan)

- Menganalisis potensi cacat, menetapkan kepemilikan dan status kepada mereka
- Mengevaluasi dan mendokumentasikan karakteristik kualitas
- Mengevaluasi temuan tinjauan terhadap kriteria keluar untuk membuat keputusan tinjauan (menolak; mayor perubahan yang diperlukan; terima, mungkin dengan sedikit perubahan)

**Memperbaiki dan melaporkan**

- Membuat laporan cacat untuk temuan yang memerlukan perubahan pada produk kerja
- Memperbaiki cacat yang ditemukan (biasanya dilakukan oleh penulis) pada produk kerja yang ditinjau
- Mengkomunikasikan cacat kepada orang atau tim yang tepat (bila ditemukan dalam produk kerja yang terkait dengan produk kerja yang ditinjau)
- Merekam status terbaru dari cacat (dalam tinjauan formal), yang berpotensi termasuk persetujuan dari pencetus komentar
- Mengumpulkan metrik (untuk jenis tinjauan yang lebih formal)
- Memeriksa apakah kriteria keluar terpenuhi (untuk jenis tinjauan yang lebih formal)
- Menerima produk kerja ketika kriteria keluar tercapai

Hasil tinjauan produk kerja bervariasi, tergantung pada jenis tinjauan dan formalitas, seperti yang dijelaskan pada bagian 3.2.3.

### 3.2.2 Peran dan tanggung jawab dalam tinjauan formal

Tinjauan formal yang khas akan mencakup peran di bawah ini:

**Pengarang**

- Membuat produk kerja dalam peninjauan
- Memperbaiki cacat pada produk kerja yang sedang ditinjau (jika perlu)

**Pengelolaan**

- Bertanggung jawab untuk perencanaan peninjauan
- Memutuskan pelaksanaan tinjauan
- Menetapkan staf, anggaran, dan waktu
- Memantau keefektifan biaya yang sedang berlangsung
- Mengeksekusi keputusan kontrol jika terjadi hasil yang tidak memadai

**Fasilitator (sering disebut sebagai moderator)**

- Memastikan berjalannya rapat tinjauan secara efektif (bila diadakan)
- Menjadi penengah, jika perlu, di antara berbagai sudut pandang
- Seringkali orang yang menjadi sandaran keberhasilan tinjauan

**Pemimpin peninjau**

- Bertanggung jawab secara keseluruhan atas tinjauan tersebut
- Memutuskan siapa yang akan terlibat dan mengatur kapan dan di mana itu akan Dilakukan

**Pengulas**

- Mungkin ahli materi pelajaran, orang yang mengerjakan proyek, pemangku kepentingan yang berkepentingan dengan produk kerja, dan/atau individu dengan latar belakang teknis atau bisnis tertentu
- Identifikasi potensi cacat pada produk kerja yang sedang ditinjau
- Dapat mewakili perspektif yang berbeda (misalnya, penguji, pengembang, pengguna, operator, analis bisnis, pakar kegunaan, dll.)

**Penulis (atau perekam)**

- Mengumpulkan potensi cacat yang ditemukan selama aktivitas peninjauan individu
- Mencatat potensi cacat baru, poin terbuka, dan keputusan dari pertemuan tinjauan (bila diadakan)

Dalam beberapa jenis tinjauan, satu orang mungkin memainkan lebih dari satu peran, dan tindakan yang terkait dengan setiap peran juga dapat bervariasi berdasarkan jenis tinjauan. Selain itu,dengan munculnya peralatan untuk mendukung proses peninjauan, terutama pencatatan cacat, titik terbuka, dan keputusan, seringkali tidak perlu juru tulis. Selanjutnya, peran yang lebih rinci kemungkinan, seperti yang dijelaskan dalam standar ISO (ISO/IEC 20246).

### 3.2.3 Jenis Tinjauan

Meskipun tinjauan dapat digunakan untuk berbagai tujuan, salah satu tujuan utamanya adalah untuk mengungkap cacat. Semua jenis tinjauan dapat membantu dalam deteksi cacat, dan jenis tinjauan yang dipilih harus didasarkan pada kebutuhan proyek, sumber daya yang tersedia, jenis dan risiko produk, domain bisnis, dan budaya perusahaan, di antara kriteria pemilihan lainnya.

Satu produk kerja dapat menjadi subjek lebih dari satu jenis tinjauan. Jika lebih dari satu jenis tinjauan digunakan, urutannya mungkin berbeda. Misalnya, tinjauan informal

dapat dilakukan sebelum tinjauan, untuk memastikan produk kerja siap untuk tinjauan teknis.

Jenis tinjauan yang dijelaskan di bawah ini dapat dilakukan sebagai tinjauan sejawat, yaitu, dilakukan oleh rekan kerja yang memenuhi syarat untuk melakukan pekerjaan yang sama.

Jenis cacat yang ditemukan dalam tinjauan bervariasi, terutama tergantung pada produk kerja yang ditinjau. (Lihat bagian 3.1.3 untuk contoh cacat yang dapat ditemukan dengan tinjauan di produk kerja yang berbeda, dan Gilb 1993 untuk informasi tentang inspeksi formal). Tinjauan dapat diklasifikasikan menurut berbagai atribut. Berikut ini daftar empat jenis tinjauan yang paling umum dan atribut terkaitnya.

**Tinjauan informal (misalnya, cek teman, pemasangan, tinjauan pasangan)**

- Tujuan utama: mendeteksi cacat potensial
- Kemungkinan tujuan tambahan: menghasilkan ide atau solusi baru, memecahkan masalah kecil dengan cepat
- Tidak berdasarkan proses formal (terdokumentasi)
- Mungkin tidak melibatkan rapat peninjauan

- Bisa dilakukan oleh rekan penulis (*buddy check*) atau lebih banyak orang
- Hasil dapat didokumentasikan
- Bervariasi dalam kegunaan tergantung pada pengulas
- Penggunaan daftar periksa bersifat opsional
- Sangat sering digunakan dalam pengembangan *Agile*

**Panduan**

- Tujuan utama: menemukan cacat, meningkatkan produk perangkat lunak, mempertimbangkan implementasi alternatif, mengevaluasi kesesuaian dengan standar dan spesifikasi
- Kemungkinan tujuan tambahan: bertukar ide tentang teknik atau variasi gaya, pelatihan peserta, mencapai persetujuan umum
- Persiapan individu sebelum pertemuan tinjauan adalah opsional
- Rapat tinjauan biasanya dipimpin oleh penulis produk kerja
- Juru tulis adalah wajib
- Penggunaan daftar periksa adalah opsional
- Dapat berbentuk skenario, latihan, atau simulasi
- Catatan cacat potensial dan laporan tinjauan dihasilkan
- Mungkin berbeda dalam praktiknya dari yang cukup informal hingga yang sangat formal

**Tinjauan teknis**

- Tujuan utama: memperoleh persetujuan umum, mendeteksi potensi cacat
- Kemungkinan tujuan lebih lanjut: mengevaluasi kualitas dan membangun kepercayaan dalam produk kerja, menghasilkan ide-ide baru, memotivasi dan memungkinkan penulis untuk meningkatkan produk kerja di masa depan, mempertimbangkan implementasi alternatif
- Peninjau harus merupakan rekan teknis penulis, dan pakar teknis dalam disiplin ilmu yang sama atau lainnya
- Diperlukan persiapan individu sebelum rapat tinjauan
- Rapat tinjauan bersifat opsional, idealnya dipimpin oleh fasilitator terlatih (biasanya bukan penulis)
- Penulis wajib, idealnya bukan penulis
- Penggunaan daftar periksa adalah opsional
- Catatan cacat potensial dan laporan tinjauan dihasilkan

**Inspeksi**

- Tujuan utama: mendeteksi potensi cacat, mengevaluasi kualitas dan membangun kepercayaan pada produk kerja, mencegah cacat serupa di masa mendatang melalui pembelajaran penulis dan analisis akar masalah
- Kemungkinan tujuan lebih lanjut: memotivasi dan memungkinkan penulis untuk meningkatkan produk kerja masa depan dan proses pengembangan perangkat lunak, mencapai persetujuan umum

- Mengikuti proses yang ditentukan dengan keluaran terdokumentasi formal, berdasarkan aturan dan daftar periksa

- Menggunakan peran yang didefinisikan dengan jelas, seperti yang ditentukan dalam bagian 3.2.2 yang bersifat wajib, dan mungkin termasuk pembaca yang berdedikasi (yang membacakan produk kerja dengan keras sering memparafrasekan, yaitu menggambarkannya dengan kata-kata sendiri, selama pertemuan tinjauan)

- Persiapan individu sebelum pertemuan tinjauan diperlukan

- Peninjau adalah rekan penulis atau ahli dalam disiplin ilmu lain yang relevan dengan produk karya

- Digunakan kriteria masuk dan keluar yang ditentukan

- Juru tulis adalah wajib

- Rapat tinjauan dipimpin oleh fasilitator terlatih (bukan penulis)

- Penulis tidak dapat bertindak sebagai pemimpin tinjauan, pembaca, atau juru tulis

- Catatan cacat potensial dan laporan tinjauan diproduksi

- Metrik dikumpulkan dan digunakan untuk meningkatkan proses pengembangan perangkat lunak secara keseluruhan, termasuk proses pemeriksaan

### 3.2.4 Menerapkan Teknik Tinjauan

Ada sejumlah teknik tinjauan yang dapat diterapkan selama aktivitas tinjauan individu (yaitu, persiapan individu) untuk mengungkap cacat. Teknik-teknik ini dapat digunakan di seluruh jenis tinjauan yang dijelaskan di atas. Efektivitas teknik mungkin berbeda tergantung pada jenis tinjauan yang digunakan. Contoh teknik tinjauan individu yang berbeda untuk berbagai jenis tinjauan tercantum di bawah ini.

***Ad hoc***

Dalam tinjauan *ad hoc*, peninjau diberikan sedikit atau tidak sama sekali panduan tentang bagaimana tugas ini harus dilakukan. Peninjau sering membaca produk kerja secara berurutan, mengidentifikasi dan mendokumentasikan masalah yang mereka hadapi. Peninjauan *ad hoc* adalah teknik umum yang membutuhkan sedikit persiapan. Teknik ini sangat bergantung pada keterampilan peninjau dan dapat menyebabkan banyak masalah duplikat yang dilaporkan oleh pengulas yang berbeda.

**Berdasarkan Pemeriksaan**

Tinjauan berdasarkan pemeriksaan adalah teknik sistematis, di mana peninjau mendeteksi masalah berdasarkan pemeriksaan yang didistribusikan pada inisiasi tinjauan (misalnya, oleh fasilitator). pemeriksaan tinjauan terdiri dari serangkaian pertanyaan berdasarkan potensi cacat, yang mungkin berasal dari pengalaman. Pemeriksaan harus spesifik untuk jenis produk kerja yang ditinjau dan harus dipelihara secara teratur untuk mencakup jenis masalah yang terlewatkan dalam tinjauan sebelumnya. Keuntungan utama dari teknik berdasarkan pemeriksaan adalah cakupan sistematis jenis cacat yang khas. Perhatian harus diberikan untuk tidak hanya mengikuti pemeriksaan dalam tinjauan individu, tetapi juga untuk mencari cacat di luar pemeriksaan.

**Skenario dan Latihan**

Dalam tinjauan berbasis skenario, peninjau diberikan panduan terstruktur tentang cara membaca produk kerja. Tinjauan berbasis skenario mendukung peninjau dalam melakukan "latihan" pada produk kerja berdasarkan penggunaan yang diharapkan dari produk kerja (jika produk kerja didokumentasikan dalam format yang sesuai seperti kasus penggunaan). Skenario ini memberikan peninjau panduan yang lebih baik tentang cara mengidentifikasi jenis cacat tertentu daripada entri daftar periksa sederhana. Seperti tinjauan berbasis daftar periksa, agar tidak melewatkan jenis cacat lainnya (misalnya, fitur yang hilang), peninjau tidak boleh dibatasi pada skenario yang didokumentasikan.

**Berdasarkan perspektif**

Dalam membaca berdasarkan perspektif, mirip dengan tinjauan berdasarkan peran, peninjau mengambil sudut pandang pemangku kepentingan yang berbeda dalam tinjauan individu. Sudut pandang pemangku kepentingan yang umum termasuk pengguna akhir, pemasaran, perancang,
penguji, atau operasi. Menggunakan sudut pandang pemangku kepentingan yang berbeda mengarah ke tinjauan individu yang lebih mendalam dengan lebih sedikit duplikasi masalah di antara pengulas. Selain itu, pemmembaca berbasis perspektif juga mengharuskan peninjau untuk mencoba menggunakan produk kerja yang ditinjau untuk menghasilkan produk yang akan mereka dapatkan darinya. Misalnya, seorang penguji akan mencoba untuk menghasilkan rencana tes penerimaan jika melakukan pemmembaca berbasis perspektif pada suatu persyaratan spesifikasi untuk melihat apakah semua informasi yang diperlukan telah disertakan. Selanjutnya, dalam membaca berbasis perspektif, daftar periksa diharapkan dapat digunakan.

Studi empiris telah menunjukkan membaca berbasis perspektif menjadi teknik umum yang paling efektif untuk meninjau persyaratan dan produk kerja teknis. Faktor kunci keberhasilan adalah memasukkan dan menimbang sudut pandang pemangku kepentingan yang berbeda secara tepat, berdasarkan risiko. Lihat Shul 2000 untuk rincian tentang membaca perspektif, dan Sauer 2000 untuk efektivitas teknik tinjauan yang berbeda.

**Berbasis peran**

Tinjauan berdasarkan peran adalah teknik di mana peninjau penerapan produk kerja dari perspektif peran pemangku kepentingan individu. Peran umum termasuk jenis pengguna akhir tertentu (berpengalaman, tidak berpengalaman, senior, anak, dll.), dan peran khusus dalam organisasi (administrator pengguna, administrator sistem, penguji kinerja, dll.). Prinsip-prinsip yang sama berlaku seperti dalam membaca berdasarkan perspektif karena berbasis serupa.

### 3.2.5 Faktor Keberhasilan untuk Tinjauan

Agar tinjauan berhasil, jenis tinjauan yang tepat dan teknik yang digunakan harus dipertimbangkan. Selain itu, ada sejumlah faktor lain yang akan mempengaruhi hasil tinjauan.

Faktor keberhasilan organisasi untuk tinjauan meliputi:

- Setiap tinjauan memiliki tujuan yang jelas, ditentukan selama perencanaan tinjauan, dan digunakan sebagai kriteria keluar yang terukur
- Jenis tinjauan diterapkan yang sesuai untuk mencapai tujuan dan sesuai dengan jenis dan tingkat produk kerja perangkat lunak dan peserta
- Teknik peninjauan apa pun yang digunakan, seperti peninjauan berdasarkan daftar periksa atau berdasarkan peran, cocok untuk identifikasi cacat yang efektif pada produk kerja yang akan ditinjau
- Daftar periksa apa pun yang digunakan membahas risiko utama dan mutakhir
- Dokumen besar ditulis dan ditinjau dalam potongan kecil, sehingga kontrol kualitas dilakukan dengan memberikan umpan balik awal dan sering kepada penulis tentang cacat
- Peserta memiliki waktu yang cukup untuk mempersiapkan
- Peninjauan dijadwalkan dengan pemberitahuan yang memadai

- Manajemen mendukung proses peninjauan (misalnya, dengan memasukkan waktu yang cukup untuk aktivitas peninjauan dalam jadwal proyek)

- Tinjauan terintegrasi dalam kualitas dan/atau kebijakan pengujian perusahaan.

Faktor keberhasilan terkait orang untuk tinjauan meliputi:

- Orang yang tepat dilibatkan untuk memenuhi tujuan tinjauan, misalnya, orang dengan keahlian atau perspektif yang berbeda, yang mungkin menggunakan dokumen sebagai masukan kerja

- Penguji dipandang sebagai pengulas yang dihargai yang berkontribusi pada tinjauan dan belajar tentang produk kerja, yang memungkinkan mereka untuk mempersiapkan pengujian yang lebih efektif, dan mempersiapkan pengujian tersebut lebih awal

- Peserta mendedikasikan waktu dan perhatian yang cukup terhadap detail

- Tinjauan dilakukan dalam potongan-potongan kecil, sehingga peninjau tidak kehilangan konsentrasi saat peninjauan individu dan/atau pertemuan peninjau (saat diadakan)

- Cacat yang ditemukan diakui, dihargai, dan ditangani secara objektif

- Rapat dikelola dengan baik, sehingga peserta menganggapnya sebagai penggunaan waktu yang berharga

- Peninjau dilakukan dalam suasana saling percaya hasilnya tidak akan digunakan untuk evaluasi peserta

- Peserta menghindari bahasa tubuh dan perilaku yang mungkin menunjukkan kebosanan kejengkelan, atau permusuhan dengan peserta lain

- Pelatihan yang memadai diberikan, terutama untuk jenis tinjauan yang lebih formal seperti inspeksi

- Budaya pembelajaran dan peningkatan proses dipromosikan

(Lihat Gilb 1993, Wiegers 2002, dan van Veenendaal 2004 untuk lebih banyak tinjauan sukses.)

| 4 Teknik Pengujian | 330 menit |
|---|---|

## Kata kunci

teknik pengujian kotak hitam, analisis nilai batas, pengujian berbasis daftar periksa, cakupan, pengujian keputusan, pengujian tabel keputusan, tebakan kesalahan, partisi ekuivalensi, teknik pengujian berbasis pengalaman,pengujian eksplorasi, pengujian transisi status, cakupan pernyataan, teknik pengujian, kasus ujipenggunaan, teknik pengujian kotak putih

### Tujuan Pembelajaran Teknik Tes

### 4.1 Kategori Teknik Tes

FL-4.1.1 (K2) Jelaskan ciri-ciri, persamaan, dan perbedaan antara uji kotak hitam teknik, teknik tes kotak putih, dan teknik tes berbasis pengalaman

### 4.2 Teknik Uji Kotak Hitam

FL-4.2.1 (K3) Terapkan partisi ekuivalensi untuk menurunkan kasus uji dari persyaratan yang diberikan

FL-4.2.2 (K3) Menerapkan analisis nilai batas untuk menurunkan kasus uji dari persyaratan yang diberikan

FL-4.2.3 (K3) Terapkan pengujian tabel keputusan untuk menurunkan kasus uji dari persyaratan yang diberikan

FL-4.2.4 (K3) Terapkan pengujian transisi status untuk mendapatkan kasus uji dari persyaratan yang diberikan

FL-4.2.5 (K2) Jelaskan cara menurunkan kasus uji dari kasus penggunaan

### 4.3 Teknik Uji Kotak Putih

FL-4.3.1 (K2) Jelaskan cakupan pernyataan

FL-4.3.2 (K2) Menjelaskan cakupan keputusan

FL-4.3.3 (K2) Menjelaskan nilai pernyataan dan cakupan keputusan

### 4.4 Teknik Tes Berdasarkan Pengalaman

FL-4.4.1 (K2) Jelaskan kesalahan dalam menebak

FL-4.4.2 (K2) Jelaskan pengujian eksplorasi

FL-4.4.3 (K2) Jelaskan pengujian berbasis daftar periksa

## 4.1 Kategori Teknik Pengujian

Tujuan dari teknik pengujian, termasuk yang dibahas dalam bagian ini, adalah untuk membantu dalam mengidentifikasi kondisi pengujian, kasus pengujian, dan data pengujian.

Pilihan teknik tes yang akan digunakan bergantung pada sejumlah faktor, termasuk:

- Kompleksitas komponen atau sistem
- Standar regulasi
- Persyaratan pelanggan atau kontrak
- Tingkat dan jenis risiko
- Dokumentasi yang tersedia
- Pengetahuan dan keterampilan penguji
- Peralatan yang tersedia
- Waktu dan anggaran
- Model siklus hidup pengembangan perangkat lunak
- Jenis cacat yang diharapkan pada komponen atau sistem

Beberapa teknik lebih dapat diterapkan pada situasi dan tingkat pengujian tertentu; yang lain berlaku untuk semua tingkat pengujian. Saat membuat kasus uji, penguji umumnya menggunakan kombinasi teknik pengujian untuk mencapai hasil terbaik dari upaya pengujian.

Penggunaan teknik tes dalam analisis pengujian, desain tes, dan aktivitas implementasi tes dapat berkisar dari sangat informal (sedikit atau tanpa dokumentasi) hingga sangat formal. Tingkat formalitas yang sesuai tergantung pada konteks pengujian, termasuk kematangan proses pengujian dan pengembangan, batasan waktu, persyaratan keselamatan atau peraturan, pengetahuan dan keterampilan orang-orang yang terlibat, dan model siklus hidup pengembangan perangkat lunak yang diikuti.

### 4.1.1 Kategori Teknik Tes dan Karakteristiknya

Dalam silabus ini, teknik tes diklasifikasikan sebagai kotak hitam, kotak putih, atau berbasis pengalaman.

Teknik pengujian kotak hitam (juga disebut teknik berbasis perilaku atau perilaku) didasarkan pada analisis dasar pengujian yang sesuai (misalnya, dokumen persyaratan formal, spesifikasi, kasus penggunaan, cerita pengguna, atau proses bisnis). Teknik-teknik ini dapat diterapkan untuk pengujian fungsional dan nonfungsional. Teknik pengujian kotak hitam berkonsentrasi pada input dan output dari objek uji tanpa mengacu pada struktur internalnya.

Teknik pengujian kotak putih (juga disebut teknik struktural atau berbasis struktur) didasarkan pada analisis arsitektur, desain detail, struktur internal, atau kode objek pengujian. Tidak seperti teknik uji kotak hitam, teknik uji kotak putih berkonsentrasi pada struktur dan pemrosesan di dalam objek uji.

Teknik pengujian berbasis pengalaman memanfaatkan pengalaman pengembang, penguji, dan pengguna untuk merancang, menerapkan, dan menjalankan pengujian. Teknik ini sering dikombinasikan dengan teknik pengujian kotak hitam dan kotak putih.

Karakteristik umum dari teknik pengujian kotak hitam adalah sebagai berikut:

- Kondisi pengujian, kasus pengujian, dan data pengujian berasal dari basis pengujian yang dapat mencakup persyaratan perangkat lunak, spesifikasi, kasus penggunaan, dan cerita pengguna
- Uji kasus dapat digunakan untuk mendeteksi kesenjangan antara persyaratan dan implementasi dari persyaratan, serta penyimpangan dari persyaratan
- Cakupan diukur berdasarkan item yang diuji dalam dasar tes dan teknik yang diterapkan untuk dasar tes

Karakteristik umum dari teknik uji kotak putih meliputi:

- Kondisi pengujian, kasus pengujian, dan data pengujian berasal dari basis pengujian yang dapat mencakup kode, arsitektur perangkat lunak, desain detail, atau sumber informasi lainnya mengenai struktur perangkat lunak

- Cakupan diukur berdasarkan item yang diuji dalam struktur yang dipilih (misalnya, kode atau antarmuka) dan teknik yang diterapkan pada dasar pengujian

Karakteristik umum dari teknik tes berdasarkan pengalaman meliputi:

- Kondisi pengujian, kasus pengujian, dan data pengujian berasal dari dasar pengujian yang mungkin mencakup pengetahuan dan pengalaman penguji, pengembang, pengguna, dan pemangku kepentingan lainnya

Pengetahuan dan pengalaman ini mencakup penggunaan yang diharapkan dari perangkat lunak, lingkungannya, kemungkinan cacat, dan distribusi cacat tersebut

Standar internasional (ISO/IEC/IEEE 29119-4) berisi denaskahsi teknik pengujian dan ukuran cakupan yang sesuai (lihat *Craig 2002* dan *Copeland 2004* untuk lebih lanjut tentang teknik).

## 4.2 Teknik Uji Kotak Hitam

### 4.2.1 Ekuivalensi Partisi

Partisi ekuivalen membagi data menjadi beberapa partisi (juga dikenal sebagai kelas ekuivalensi) sedemikian rupa sehingga semua anggota partisi tertentu diharapkan diproses dengan cara yang sama (lihat Kaner 2013 dan Jorgensen 2014). Ada partisi ekuivalensi untuk nilai valid dan tidak valid.

- Nilai-nilai yang valid adalah nilai-nilai yang harus diterima oleh komponen atau sistem.Sebuah partisi ekuivalensi yang berisi nilai-nilai yang valid disebut "partisi ekuivalensi yang valid."

- Nilai-nilai yang tidak valid adalah nilai-nilai yang harus ditolak oleh komponen atau sistem. Partisi ekuivalensi yang berisi nilai yang tidak valid disebut "partisi ekuivalensi tidak valid".

- Partisi dapat diidentifikasi untuk elemen data apa pun yang terkait dengan objek uji, termasuk input, output, nilai internal, nilai terkait waktu (misalnya, sebelum atau sesudah suatu peristiwa) dan untuk parameter antarmuka (misalnya, komponen terintegrasi yang diuji selama pengujian integrasi).

- Setiap partisi dapat dibagi menjadi sub-partisi jika diperlukan.

- Setiap nilai harus dimiliki oleh satu dan hanya satu partisi ekivalen.

- Ketika partisi ekuivalensi yang tidak valid digunakan dalam kasus uji, partisi tersebut harus diuji satu per satu, yaitu, tidak digabungkan dengan partisi ekuivalensi tidak valid lainnya, untuk memastikan bahwa kegagalan tidak ditutupi.

Kegagalan dapat ditutup-tutupi ketika beberapa kegagalan terjadi pada waktu yang sama tetapi hanya satu yang terlihat, menyebabkan kegagalan lainnya tidak terdeteksi.
Untuk mencapai cakupan 100% dengan teknik ini, kasus pengujian harus mencakup semua partisi yang diidentifikasi (termasuk partisi yang tidak valid) dengan menggunakan minimal satu nilai dari setiap partisi. Cakupan diukur sebagai jumlah partisi ekuivalensi yang diuji dengan setidaknya satu nilai, dibagi dengan jumlah total partisi ekuivalensi yang teridentifikasi, biasanya dinyatakan sebagai persentase. Partisi ekuivalensi dapat diterapkan di semua tingkat pengujian.

### 4.2.2 Analisis Nilai Batas

Analisis nilai batas (BVA) adalah perpanjangan dari partisi ekuivalensi, tetapi hanya dapat digunakan ketika partisi dipesan, yang terdiri dari data numerik atau berurutan. Nilai minimum dan maksimum (atau nilai pertama dan terakhir) dari sebuah partisi adalah nilai batasnya (lihat *Beizer 1990*).

Sebagai contoh, anggaplah sebuah field input menerima satu nilai bilangan bulat sebagai input, menggunakan keypad untuk membatasi input sehingga input bukan bilangan bulat tidak mungkin dilakukan. Rentang yang valid adalah dari 1 hingga 5, inklusif. Jadi, ada tiga partisi ekuivalensi: tidak valid (terlalu rendah); sah; tidak valid (terlalu tinggi). Untuk partisi ekivalen yang valid, nilai batasnya adalah 1 dan 5. Untuk partisi yang tidak valid (terlalu tinggi), nilai batasnya adalah 6. partisi tidak valid (terlalu rendah), hanya ada satu nilai batas, 0, karena ini adalah partisi yang hanya memiliki satu anggota.

Pada contoh di atas, kita mengidentifikasi dua nilai batas per batas. Batas antara tidak valid (terlalu rendah) dan valid memberikan nilai pengujian 0 dan 1. Batas antara valid dan tidak valid (terlalu tinggi) memberikan nilai uji 5 dan 6. Beberapa variasi teknik ini mengidentifikasi tiga nilai batas per batas: nilai sebelum, pada, dan tepat di atas batas. Dalam contoh sebelumnya, menggunakan nilai batas tiga titik, nilai uji batas bawah adalah 0, 1, dan 2, dan nilai uji batas atas adalah 4, 5, dan 6 (lihat Jorgensen 2014).

Perilaku pada batas-batas partisi ekuivalensi lebih mungkin salah daripada perilaku di dalam partisi. Penting untuk diingat bahwa batas-batas yang ditentukan dan yang diterapkan dapat dipindahkan ke posisi di atas atau di bawah posisi yang dimaksudkan, dapat dihilangkan sama sekali, atau dapat dilengkapi dengan batas tambahan yang tidak diinginkan. Analisis dan pengujian nilai batas akan mengungkapkan hampir semua cacat seperti itu dengan memaksa perangkat lunak untuk menunjukkan perilaku dari partisi selain dari partisi yang seharusnya menjadi milik nilai batas.

Analisis nilai batas dapat diterapkan di semua tingkat pengujian. Teknik ini umumnya digunakan untuk menguji persyaratan yang membutuhkan rentang angka (termasuk tanggal dan waktu). Cakupan batas untuk partisi diukur sebagai jumlah nilai batas yang diuji, dibagi dengan jumlah total nilai uji batas yang diidentifikasi, biasanya dinyatakan sebagai persentase.

### 4.2.3 Pengujian Tabel Keputusan

Tabel keputusan adalah cara yang baik untuk merekam aturan bisnis yang kompleks yang harus diterapkan oleh sistem. Saat membuat tabel keputusan, penguji mengidentifikasi kondisi (seringkali input) dan tindakan yang dihasilkan (seringkali merupakan keluaran) dari sistem. Ini membentuk baris tabel, biasanya dengan kondisi di atas dan tindakan di bawah. Setiap kolom sesuai dengan aturan keputusan yang mendefinisikan kombinasi unik dari kondisi yang menghasilkan eksekusi tindakan yang terkait dengan aturan itu. Nilai kondisi dan tindakan biasanya ditampilkan sebagai nilai Boolean (benar atau salah) atau nilai diskrit (misalnya, merah, hijau, biru), tetapi juga

dapat berupa angka atau rentang angka. Jenis kondisi dan tindakan yang berbeda ini dapat ditemukan bersama dalam tabel yang sama.

Notasi yang umum pada tabel keputusan adalah sebagai berikut:
Untuk kondisi:

- Y berarti kondisinya benar (mungkin juga ditampilkan sebagai T atau 1)
- N berarti kondisinya salah (mungkin juga ditampilkan sebagai F atau 0)
- — berarti nilai kondisi tidak menjadi masalah (mungkin juga ditampilkan sebagai N/A) Untuk tindakan:
- X berarti aksi harus terjadi (mungkin juga ditampilkan sebagai Y atau T atau 1)
- Kosong berarti tindakan tersebut tidak boleh terjadi (juga dapat ditampilkan sebagai – atau N atau F atau 0)

Tabel keputusan lengkap memiliki kolom yang cukup (kasus uji) untuk mencakup setiap kombinasi kondisi. Dengan menghapus kolom yang tidak mempengaruhi hasil, jumlah kasus uji dapat berkurang secara signifikan. Misalnya dengan menghapus kombinasi kondisi yang mustahil. Untuk informasi selengkapnya tentang cara mengecilkan tabel keputusan. (lihat ISTQB-CTAL-AT).

Standar cakupan minimum umum untuk pengujian tabel keputusan adalah memiliki setidaknya satu kasus pengujian per aturan keputusan dalam tabel. Ini biasanya mencakup semua kombinasi kondisi. Cakupan diukur sebagai jumlah aturan keputusan yang diuji oleh setidaknya satu kasus uji, dibagi dengan jumlah total aturan keputusan, biasanya dinyatakan sebagai persentase.

Kekuatan pengujian tabel keputusan adalah membantu mengidentifikasi semua kombinasi penting dari kondisi, beberapa di antaranya mungkin diabaikan. Hal ini juga membantu dalam menemukan kesenjangan dalam persyaratan. Ini dapat diterapkan pada semua situasi di mana perilaku perangkat lunak bergantung pada kombinasi kondisi, pada tingkat pengujian apa pun.

### 4.2.4 Pengujian Peralihan Status

Komponen atau sistem dapat merespons secara berbeda terhadap suatu peristiwa tergantung pada kondisi saat ini atau riwayat sebelumnya (misalnya, peristiwa yang telah terjadi sejak sistem diinisialisasi). Sejarah sebelumnya dapat diringkas menggunakan konsep negara. Diagram transisi status menunjukkan kemungkinan status perangkat lunak, serta bagaimana perangkat lunak masuk, keluar, dan transisi antar status. Transisi dimulai oleh suatu peristiwa (misalnya, input pengguna dari suatu nilai ke dalam bidang). Peristiwa tersebut menghasilkan transisi.Kejadian yang sama dapat menghasilkan dua atau lebih transisi yang berbeda dari keadaan yang sama. Perubahan status dapat mengakibatkan perangkat lunak mengambil tindakan (misalnya, mengeluarkan perhitungan atau pesan kesalahan).

Tabel transisi status menunjukkan semua transisi yang valid dan transisi yang berpotensi tidak valid antar status, serta peristiwa, dan tindakan yang dihasilkan untuk transisi yang valid. Diagram transisi status biasanya hanya menampilkan transisi yang valid dan mengecualikan transisi yang tidak valid.

Pengujian dapat dirancang untuk mencakup urutan keadaan tertentu, untuk melatih semua keadaan, untuk melatih setiap transisi, untuk menjalankan urutan transisi tertentu, atau untuk menguji transisi yang tidak valid.

Pengujian transisi status digunakan untuk aplikasi berbasis menu dan digunakan secara luas dalam industri perangkat lunak tertanam. Teknik ini juga cocok untuk membuat model skenario bisnis yang memiliki status tertentu atau untuk menguji

navigasi layar. Konsep negara bersifat abstrak – mungkin mewakili beberapa baris kode atau keseluruhan proses bisnis.

Cakupan biasanya diukur sebagai jumlah status atau transisi yang diidentifikasi yang diuji, dibagi dengan jumlah total status atau transisi yang diidentifikasi dalam objek uji, biasanya dinyatakan dalam persentase. Untuk informasi selengkapnya tentang kriteria cakupan untuk pengujian transisi status, (lihat ISTQB-CTAL-AT).

### 4.2.5 Kasus Uji Penggunaan

Pengujian dapat diturunkan dari kasus penggunaan, yang merupakan cara spesifik untuk merancang interaksi dengan item perangkat lunak. Mereka memasukkan persyaratan untuk fungsi-fungsi perangkat lunak. Kasus pengguna diasosiasikan dengan aktor (pengguna manusia, perangkat keras eksternal, atau komponen atau sistem lain) dan subyek (komponen atau sistem yang digunakan untuk kasus pengguna).

Setiap kasus penggunaan menentukan beberapa perilaku yang dapat dilakukan subjek dalam kolaborasi dengan satu atau lebih aktor (UML 2.5.1 2017). Sebuah kasus pengguna dapat didenaskahsikan melalui interaksi dan aktivitas, serta prakondisi, pascakondisi dan bahasa alami jika sesuai. Interaksi antara aktor dan subjek dapat menyebabkan perubahan pada keadaan subjek. Interaksi dapat diwakili secara grafis dengan alur kerja, diagram aktivitas, atau model proses bisnis.

Sebuah kasus pengguna dapat mencakup kemungkinan variasi dari perilaku dasarnya, termasuk perilaku yang luar biasa dan penanganan kesalahan (respon sistem dan pemulihan dari kesalahan pemrograman, aplikasi dan komunikasi, misalnya, menghasilkan pesan kesalahan). Tes dirancang untuk melatih perilaku yang ditentukan (dasar, luar biasa atau alternatif, dan penanganan kesalahan). Cakupan dapat diukur dengan jumlah perilaku kasus pengguna yang diuji dibagi dengan jumlah total perilaku kasus pengguna, yang biasanya dinyatakan dalam persentase. Untuk informasi selengkapnya tentang kriteria cakupan untuk kasus ujipengguna. (lihat ISTQB-CTAL-AT).

## 4.3 Teknik Pengujian Kotak Putih

Pengujian kotak putih didasarkan pada struktur internal objek uji. Teknik uji kotak putih dapat digunakan di semua tingkat pengujian, tetapi dua teknik terkait kode yang dibahas dalam bagian ini paling sering digunakan pada tingkat pengujian komponen. Ada teknik yang lebih maju yang digunakan di beberapa lingkungan kritis keselamatan, kritis misi, atau integritas tinggi untuk mencapai cakupan yang lebih menyeluruh, tetapi itu tidak dibahas di sini. Untuk informasi lebih lanjut tentang teknik semacam itu, lihat ISTQB-CTAL-TTA.

### 4.3.1 Pengujian Pernyataan dan Cakupan

Pengujian pernyataan melatih pernyataan potensial yang dapat dieksekusi dalam kode. Cakupan diukur sebagai jumlah pernyataan yang dieksekusi oleh tes dibagi dengan jumlah total pernyataan yang dapat dieksekusi dalam objek tes, biasanya dinyatakan sebagai persentase.

### 4.3.2 Pengujian Keputusan dan Cakupan

Pengujian keputusan melatih keputusan dalam kode dan menguji kode yang dieksekusi berdasarkan hasil keputusan. Untuk melakukan ini, kasus uji mengikuti aliran kontrol yang terjadi dari titik keputusan (misalnya, untuk pernyataan *IF*, satu untuk hasil yang benar dan satu untuk hasil yang salah; untuk pernyataan KASUS, kasus uji akan diperlukan untuk semua kemungkinan hasil, termasuk hasil standar).

Cakupan diukur sebagai jumlah hasil keputusan yang dilaksanakan oleh tes dibagi dengan jumlah total hasil keputusan dalam objek tes, biasanya dinyatakan sebagai persentase.

### 4.3.3 Nilai Pernyataan dan Pengujian Keputusan

Ketika cakupan pernyataan 100% tercapai, ini memastikan bahwa semua pernyataan yang dapat dieksekusi dalam kode telah diuji setidaknya satu kali, tetapi itu tidak memastikan bahwa semua catatanika keputusan telah diuji. Dari dua teknik kotak putih yang dibahas dalam silabus ini, pengujian pernyataan dapat memberikan cakupan yang lebih sedikit daripada keputusan pengujian.

Ketika cakupan keputusan 100% tercapai, itu mengeksekusi semua hasil keputusan, yang mencakup pengujian hasil yang benar dan juga hasil yang salah, bahkan ketika tidak ada pernyataan palsu yang eksplisit (misalnya, dalam kasus pernyataan *IF* tanpa yang lain dalam kode). Cakupan pernyataan membantu menemukan cacat dalam kode yang tidak dilakukan oleh pengujian lain. Cakupan keputusan membantu untuk menemukan cacat dalam kode di mana tes lain belum mengambil hasil yang benar dan salah.

Mencapai cakupan keputusan 100% menjamin cakupan pernyataan 100% (tetapi tidak sebaliknya).

## 4.4 Teknik Tes Berdasarkan Pengalaman

Saat menerapkan teknik pengujian berbasis pengalaman, kasus pengujian diturunkan dari keterampilan dan intuisi penguji, serta pengalaman mereka dengan aplikasi dan teknologi serupa. Teknik-teknik ini dapat membantu dalam mengidentifikasi tes yang tidak mudah diidentifikasi oleh teknik lain yang lebih sistematis. Bergantung pada pendekatan dan pengalaman penguji, teknik-teknik ini dapat mencapai tingkat cakupan dan efektivitas yang sangat bervariasi. Cakupan bisa sulit untuk dinilai dan mungkin tidak dapat diukur dengan teknik ini.

Teknik-teknik berbasis pengalaman yang umum digunakan dibahas di bagian berikut.

### 4.4.1 Kesalahan Menebak

Kesalahan menebak adalah teknik yang digunakan untuk mengantisipasi terjadinya kesalahan, cacat, dan kegagalan, berdasarkan pengetahuan penguji, meliputi:

- Bagaimana aplikasi bekerja di masa lalu
- Kesalahan seperti apa yang cenderung dilakukan?
- Kegagalan yang terjadi di aplikasi lain

Pendekatan metodis untuk teknik tebakan kesalahan adalah dengan membuat daftar kemungkinan kesalahan, cacat, dan kegagalan, dan pengujian desain yang akan mengungkapkan kegagalan tersebut dan cacat yang menyebabkannya.

kesalahan ini, cacat, daftar kegagalan dapat dibuat berdasarkan pengalaman, data cacat dan kegagalan, atau dari pengetahuan umum tentang mengapa perangkat lunak gagal.

### 4.4.2 Pengujian Eksplorasi

Dalam pengujian eksplorasi, pengujian informal (tidak ditentukan sebelumnya) dirancang, dijalankan, dicatat, dan dievaluasi secara dinamis selama pelaksanaan pengujian. Hasil tes digunakan untuk mempelajari lebih lanjut tentang komponen atau sistem, dan untuk membuat pengujian untuk area yang mungkin memerlukan pengujian lebih lanjut.

Pengujian eksplorasi terkadang dilakukan menggunakan pengujian berbasis sesi untuk menyusun aktivitas. Dalam pengujian berbasis sesi, pengujian eksplorasi dilakukan dalam kotak waktu yang ditentukan, dan penguji menggunakan piagam pengujian yang berisi tujuan pengujian untuk memandu pengujian. Penguji dapat menggunakan lembar sesi uji untuk mendokumentasikan langkah-langkah yang diikuti dan penemuan yang dibuat.

Pengujian eksplorasi paling berguna ketika spesifikasinya sedikit atau tidak memadai atau waktu yang signifikan tekanan pada pengujian. Pengujian eksplorasi juga berguna untuk melengkapi teknik pengujian formal lainnya. Pengujian eksplorasi sangat terkait dengan strategi pengujian reaktif (lihat bagian 5.2.2). Pengujian eksplorasi dapat menggabungkan penggunaan teknik kotak hitam, kotak putih, dan pengalaman lainnya.

### 4.4.3 Pengujian Berbasis Daftar periksa

Dalam pengujian berbasis daftar periksa, penguji merancang, menerapkan, dan menjalankan pengujian untuk mencakup kondisi pengujian yang ditemukan dalam daftar periksa. Sebagai bagian dari analisis, penguji membuat daftar periksa baru atau memperluas daftar periksa yang ada, tetapi penguji juga dapat menggunakan daftar periksa yang ada tanpa modifikasi. Daftar periksa tersebut dapat dibangun berdasarkan pengalaman, pengetahuan tentang apa yang penting bagi pengguna, atau pemahaman tentang mengapa dan bagaimana perangkat lunak gagal.

Daftar periksa dapat dibuat untuk mendukung berbagai jenis pengujian, termasuk pengujian fungsional dan non-fungsional. Dengan tidak adanya kasus uji yang terperinci, pengujian berbasis daftar periksa dapat memberikan pedoman dan tingkat konsistensi. Karena ini adalah daftar tingkat tinggi, beberapa variabilitas dalam pengujian yang sebenarnya mungkin terjadi, menghasilkan cakupan yang berpotensi lebih besar tetapi pengulangan yang lebih sedikit.

# 5 Manajemen Tes — 225 menit

**Kata Kunci**

manajemen konfigurasi, manajemen cacat, laporan cacat, kriteria masuk, kriteria keluar, risiko produk, risiko proyek, risiko, tingkat risiko, pengujian berbasis risiko, pendekatan pengujian, kontrol pengujian, estimasi pengujian, manajer pengujian, pemantauan pengujian, rencana pengujian, pengujian perencanaan, laporan kemajuan pengujian, strategi pengujian, laporan ringkasan pengujian, penguji

## Tujuan Pembelajaran untuk Manajemen Tes

### 5.1 Organisasi Tes

FL-5.1.1 (K2) Jelaskan keuntungan dan kerugian dari pengujian independen

FL-5.1.2 (K1) Identifikasi tugas manajer pengujian dan penguji

### 5.2 Perencanaan dan Estimasi Tes

FL-5.2.1 (K2) Meringkas tujuan dan isi rencana pengujian

FL-5.2.2 (K2) Bedakan antara berbagai strategi pengujian

FL-5.2.3 (K2) Berikan contoh kriteria masuk dan keluar yang potensial

FL-5.2.4 (K3) Menerapkan pengetahuan tentang prioritas, dan dependensi teknis dan logis, untuk menjadwalkan eksekusi pengujian untuk serangkaian kasus pengujian tertentu

FL-5.2.5 (K1) Mengidentifikasi faktor-faktor yang mempengaruhi upaya yang berkaitan dengan pengujian

FL-5.2.6 (K2) Jelaskan perbedaan antara dua teknik estimasi: teknik berbasis metrik dan teknik berbasis pakar

### 5.3 Pemantauan dan Kontrol Uji

FL-5.3.1 (K1) Mengingat metrik yang digunakan untuk pengujian

FL-5.3.2 (K2) Ringkas tujuan, isi, dan audiens untuk laporan pengujian

### 5.4 Manajemen Konfigurasi

FL-5.4.1 (K2) Rangkum bagaimana manajemen konfigurasi mendukung pengujian

### 5.5 Risiko dan Pengujian

FL-5.5.1 (K1) Tentukan tingkat risiko dengan menggunakan kemungkinan dan dampak

FL-5.5.2 (K2) Bedakan antara risiko proyek dan produk

FL-5.5.3 (K2) Jelaskan, dengan menggunakan contoh, bagaimana analisis risiko produk dapat mempengaruhi ketelitian dan cakupan pengujian

### 5.6 Manajemen Cacat

FL-5.6.1 (K3) Tulis laporan cacat, yang mencakup cacat yang ditemukan selama pengujian

## 5.1 Organisasi Tes

### 5.1.1 Pengujian Independen

Tugas pengujian dapat dilakukan oleh orang-orang dalam peran pengujian tertentu, atau oleh orang-orang dalam peran lain (misalnya, pelanggan). Tingkat independensi tertentu sering kali membuat penguji lebih efektif dalam menemukan cacat karena perbedaan antara bias kognitif penulis dan penguji (lihat bagian 1.5). Akan tetapi, independensi bukanlah pengganti keakraban, dan pengembang dapat secara efisien menemukan banyak cacat dalam kode mereka sendiri.

Derajat independensi dalam pengujian meliputi hal-hal berikut (dari tingkat independensi rendah hingga tingkat tinggi):

- Tidak ada penguji independen; satu-satunya bentuk pengujian yang tersedia adalah pengembang menguji kode mereka sendiri
- Pengembang atau penguji independen dalam tim pengembangan atau tim proyek; ini bisa jadi pengembang menguji produk rekan mereka
- Tim atau grup penguji independen dalam organisasi, melapor ke manajemen proyek atau manajemen eksekutif
- Penguji independen dari organisasi bisnis atau komunitas pengguna, atau dengan spesialisasi dalam jenis pengujian tertentu seperti kegunaan, keamanan, kinerja, peraturan/kepatuhan, atau portabilitas
- Penguji independen di luar organisasi, baik yang bekerja di lokasi (*in-house*) atau di luar lokasi (*outsourcing*)

Untuk sebagian besar jenis proyek, biasanya yang terbaik adalah memiliki beberapa tingkat pengujian, dengan beberapa tingkat ini ditangani oleh penguji independen. Pengembang harus berpartisipasi dalam pengujian, terutama di tingkat yang lebih rendah, untuk melakukan kontrol atas kualitas pekerjaan mereka sendiri.

Cara penerapan independensi pengujian bervariasi tergantung pada model siklus hidup pengembangan perangkat lunak. Misalnya, dalam pengembangan *Agile*, penguji dapat menjadi bagian dari tim pengembangan. di beberapa organisasi yang menggunakan metode *Agile*, penguji ini juga dapat dianggap sebagai bagian dari tim uji independen yang lebih besar. Selain itu, di organisasi tersebut, pemilik produk dapat melakukan pengujian penerimaan untuk memvalidasi cerita pengguna di akhir setiap iterasi.

Potensi manfaat dari independensi tes meliputi:

- Penguji independen cenderung mengenali berbagai jenis kegagalan dibandingkan dengan pengembang karena latar belakang mereka yang berbeda, perspektif teknis, dan bias
- Penguji independen dapat memverifikasi, menantang, atau menyangkal asumsi yang dibuat oleh pemangku kepentingan selama spesifikasi dan implementasi system
- Penguji independen dari vendor dapat melaporkan secara jujur dan objektif tentang sistem yang diuji tanpa tekanan (politik) dari perusahaan yang mempekerjakan mereka

Potensi kelemahan dari independensi tes meliputi:

- Isolasi dari tim pengembangan, dapat menyebabkan kurangnya kolaborasi, keterlambatan dalam memberikan umpan balik kepada tim pengembangan, atau hubungan yang bertentangan dengan tim pengembangan
- Pengembang mungkin kehilangan rasa tanggung jawab terhadap kualitas
- Penguji independen dapat dilihat sebagai hambatan

- Penguji independen mungkin kekurangan beberapa informasi penting (misalnya tentang objek uji)

Banyak organisasi berhasil mencapai manfaat dari independensi tes sambil menghindari kekurangannya.

### 5.1.2 Tugas Manajer Tes dan Penguji

Dalam silabus ini, dua peran tes tercakup, manajer tes dan penguji. Aktivitas dan tugas yang dilakukan oleh kedua peran ini bergantung pada konteks proyek dan produk, keterampilan orang-orang dalam peran tersebut, dan organisasi.

Manajer tes ditugaskan dengan tanggung jawab keseluruhan untuk proses tes dan kepemimpinan yang sukses dari aktivitas tes. Peran manajemen pengujian dapat dilakukan oleh manajer pengujian profesional, atau oleh manajer proyek, manajer pengembangan, atau manajer jaminan kualitas. Dalam proyek atau organisasi yang lebih besar, beberapa tim pengujian dapat melapor ke manajer pengujian, pelatih pengujian, atau koordinator pengujian, masing-masing tim dipimpin oleh pemimpin pengujian atau penguji utama.

Tugas manajer tes yang umum dapat mencakup:

- Mengembangkan atau meninjau kebijakan pengujian dan strategi pengujian untuk organisasi
- Merencanakan aktivitas pengujian dengan mempertimbangkan konteks, dan pahami tujuan dan risiko pengujian. Ini mungkin termasuk memilih pendekatan pengujian, memperkirakan waktu pengujian, upaya dan biaya, memperoleh sumber daya, menentukan tingkat pengujian dan siklus pengujian, dan merencanakan manajemen cacat.
- Menulis dan memperbarui rencana pengujian
- Mengkoordinasikan rencana pengujian dengan manajer proyek, pemilik produk, dan lainnya
- Membagikan perspektif pengujian dengan aktivitas proyek lainnya, seperti perencanaan integrasi
- Memulai analisis, desain, implementasi, dan pelaksanaan pengujian, memantau kemajuan dan hasil pengujian, dan memeriksa status kriteria keluar (atau definisi selesai) dan memfasilitasi aktivitas penyelesaian pengujian
- Menyiapkan dan menyampaikan laporan kemajuan pengujian dan laporan ringkasan pengujian berdasarkan informasi yang dikumpulkan
- Menyesuaikan perencanaan berdasarkan hasil dan kemajuan pengujian (terkadang didokumentasikan dalam laporan kemajuan pengujian, dan/atau dalam laporan ringkasan pengujian untuk pengujian lain yang telah diselesaikan pada proyek) dan mengambil tindakan yang diperlukan untuk pengendalian pengujian
- Mendukung pengaturan sistem manajemen cacat dan manajemen konfigurasi peralatan pengujian yang memadai
- Memperkenalkan metrik yang sesuai untuk mengukur kemajuan pengujian dan mengevaluasi kualitas pengujian dan produk
- Mendukung pemilihan dan implementasi peralatan untuk mendukung proses pengujian, termasuk merekomendasikan anggaran untuk pemilihan peralatan (dan mungkin pembelian dan/atau dukungan), mengalokasikan waktu dan upaya untuk proyek percontohan, dan memberikan dukungan berkelanjutan dalam penggunaan peralatan.)
- Memutuskan tentang penerapan lingkungan pengujian
- Mempromosikan dan mengadvokasi penguji, tim penguji, dan profesi penguji di dalam organisasi

- Mengembangkan keterampilan dan karir penguji (misalnya, melalui rencana pelatihan, evaluasi kinerja, pembinaan, dll.)

Cara menjalankan peran manajer pengujian bervariasi tergantung pada siklus hidup pengembangan perangkat lunak. Misalnya, dalam pengembangan Agile, beberapa tugas yang disebutkan di atas ditangani oleh tim Agile, terutama tugas-tugas yang berkaitan dengan pengujian sehari-hari yang dilakukan di dalam tim, seringkali oleh penguji yang bekerja di dalam tim. Beberapa tugas yang mencakup beberapa tim atau seluruh organisasi, atau yang berkaitan dengan manajemen personalia, dapat dilakukan oleh manajer pengujian di luar tim pengembangan, yang terkadang disebut pelatih pengujian. Lihat *Black* 2009 untuk lebih lanjut tentang mengelola proses pengujian.

Tipe tugas penguji dapat mencakup:

- Tinjau dan kontribusi untuk rencana pengujian
- Menganalisis, meninjau, dan menilai persyaratan, cerita pengguna dan kriteria penerimaan, spesifikasi, dan model untuk dapat diuji (yaitu, dasar pengujian)
- Mengidentifikasi dan mendokumentasikan kondisi pengujian, dan menangkap ketertelusuran antara kasus uji, kondisi pengujian, dan dasar pengujian
- Merancang, mengatur, dan memverifikasi lingkungan pengujian, sering kali berkoordinasi dengan administrasi sistem dan manajemen jaringan
- Merancang dan mengimplementasikan kasus uji dan prosedur pengujian
- Mempersiapkan dan memperoleh data uji
- Buat jadwal pelaksanaan tes yang terperinci
- Jalankan tes, evaluasi hasil, dan dokumentasikan penyimpangan dari hasil yang diharapkan
- Gunakan peralatan yang sesuai untuk memfasilitasi proses pengujian
- Mengotomatiskan pengujian sesuai kebutuhan (mungkin didukung oleh pengembang atau ahli otomatisasi pengujian)
- Evaluasi karakteristik non-fungsional seperti efisiensi kinerja, keandalan, kegunaan, keamanan, kompatibilitas, dan portabilitas
- Tinjau tes yang dikembangkan oleh orang lain

Orang yang bekerja pada analisis pengujian, desain pengujian, jenis pengujian tertentu, atau otomatisasi pengujian mungkin merupakan spesialis dalam peran ini. Bergantung pada risiko yang terkait dengan produk dan proyek, dan model siklus hidup pengembangan perangkat lunak yang dipilih, orang yang berbeda dapat mengambil alih peran penguji pada tingkat pengujian yang berbeda. Misalnya, pada tingkat pengujian komponen dan tingkat pengujian integrasi komponen, peran seorang tester sering dilakukan oleh pengembang. Pada tingkat tes penerimaan, peran penguji sering dilakukan oleh analis bisnis, pakar materi pelajaran, dan pengguna. Pada tingkat pengujian sistem dan tingkat pengujian integrasi sistem, peran penguji sering dilakukan oleh tim penguji independen. Pada tingkat uji penerimaan operasional, peran penguji sering dilakukan oleh staf administrasi operasi dan/atau sistem.

## 5.2 Perencanaan dan Estimasi Tes

### 5.2.1 Tujuan dan Isi Rencana Uji

Rencana pengujian menguraikan aktivitas pengujian untuk proyek pengembangan dan pemeliharaan. Perencanaan dipengaruhi oleh kebijakan pengujian dan strategi pengujian organisasi, siklus hidup pengembangan dan metode yang digunakan (lihat bagian 2.1), ruang lingkup pengujian, tujuan, risiko, kendala, kekritisan, kemampuan pengujian, dan ketersediaan sumber daya.

Seiring dengan kemajuan proyek dan perencanaan pengujian, semakin banyak informasi yang tersedia dan lebih banyak detail dapat dimasukkan dalam rencana pengujian. Perencanaan pengujian adalah aktivitas yang berkesinambungan dan dilakukan di sepanjang siklus hidup produk. (Perhatikan bahwa siklus hidup produk dapat melampaui lingkup proyek untuk memasukkan fase pemeliharaan.) Umpan balik dari aktivitas pengujian harus digunakan untuk mengenali risiko yang berubah sehingga perencanaan dapat disesuaikan. Perencanaan dapat didokumentasikan dalam rencana pengujian induk dan dalam rencana pengujian terpisah untuk tingkat pengujian, seperti pengujian sistem dan pengujian penerimaan, atau untuk jenis pengujian terpisah, seperti pengujian kegunaan dan pengujian kinerja.

Aktivitas perencanaan pengujian dapat mencakup hal-hal berikut dan beberapa di antaranya dapat didokumentasikan dalam rencana pengujian:

- Menentukan ruang lingkup, tujuan, dan risiko pengujian
- Mendefinisikan pendekatan pengujian secara keseluruhan
- Mengintegrasikan dan mengkoordinasikan aktivitas pengujian ke dalam aktivitas siklus hidup perangkat lunak
- Membuat keputusan tentang apa yang akan diuji, orang-orang, dan sumber daya lain yang diperlukan untuk melakukan berbagai aktivitas pengujian, dan bagaimana aktivitas pengujian akan dilakukan
- Penjadwalan analisis pengujian, desain, implementasi, pelaksanaan, dan aktivitas evaluasi, baik pada tanggal tertentu (misalnya, dalam pengembangan berurutan) atau dalam konteks setiap iterasi (misalnya, dalam pengembangan berulang)
- Memilih metrik untuk pemantauan dan kontrol pengujian
- Penganggaran untuk aktivitas pengujian
- Menentukan tingkat detail dan struktur untuk dokumentasi pengujian (misalnya, dengan menyediakan template atau contoh dokumen)

Isi dari rencana pengujian bervariasi, dan dapat melampaui topik yang telah diidentifikasi di atas. Struktur rencana pengujian sampel dan rencana pengujian sampel dapat ditemukan di standar ISO (ISO/IEC/IEEE 29119-3).

### 5.2.2 Strategi dan Pendekatan Pengujian

Strategi pengujian memberikan gambaran umum tentang proses pengujian, biasanya pada tingkat produk atau organisasi. Jenis umum strategi pengujian meliputi:

- **Analitis:** Jenis strategi pengujian ini didasarkan pada analisis beberapa faktor (misalnya, persyaratan atau risiko). Pengujian berbasis risiko adalah contoh pendekatan analitis, di mana pengujian dirancang dan diprioritaskan berdasarkan tingkat risiko.
- **Berbasis Model:** Dalam jenis strategi pengujian ini, pengujian dirancang berdasarkan beberapa model dari beberapa aspek produk yang diperlukan, seperti fungsi, proses bisnis, struktur internal, atau karakteristik non-fungsional (misalnya, keandalan). Contoh model tersebut meliputi model proses bisnis, model keadaan, dan model pertumbuhan keandalan.

- **Metodis:** Jenis strategi pengujian ini bergantung pada penggunaan sistematis dari beberapa rangkaian pengujian atau kondisi pengujian yang telah ditentukan sebelumnya, seperti taksonomi jenis kegagalan yang umum atau mungkin terjadi, daftar karakteristik kualitas penting, atau standar tampilan dan nuansa perusahaan untuk aplikasi seluler atau halaman web.

- **Sesuai dengan proses (atau sesuai standar):** Jenis strategi pengujian ini melibatkan analisis, perancangan, dan penerapan pengujian berdasarkan aturan dan standar eksternal, seperti yang ditentukan oleh standar khusus industri, dengan dokumentasi proses, dengan identifikasi yang ketat dan penggunaan dasar tes, atau oleh setiap proses atau standar yang diberlakukan pada atau oleh organisasi.

- **Diarahkan (atau konsultatif):** Jenis strategi pengujian ini terutama didorong oleh saran, bimbingan, atau instruksi dari pemangku kepentingan, pakar domain bisnis, atau pakar teknologi, yang mungkin berada di luar tim pengujian atau di luar organisasi itu sendiri.

- **Penolakan regresi:** Jenis strategi pengujian ini dimotivasi oleh keinginan untuk menghindari regresi kemampuan yang ada. Strategi pengujian ini mencakup penggunaan kembali perangkat uji yang ada (khususnya kasus uji dan data pengujian), otomatisasi pengujian regresi yang ekstensif, dan rangkaian pengujian standar.

- **Reaktif:** Dalam jenis strategi pengujian ini, pengujian bersifat reaktif terhadap komponen atau sistem yang sedang diuji, dan peristiwa yang terjadi selama pelaksanaan pengujian, daripada yang telah direncanakan sebelumnya (seperti strategi sebelumnya). Tes dirancang dan diimplementasikan, dan dapat segera dilaksanakan sebagai tanggapan atas pengetahuan yang diperoleh dari hasil tes sebelumnya. Pengujian eksplorasi adalah teknik umum yang digunakan dalam strategi reaktif.

Sebuah strategi pengujian yang tepat sering dibuat dengan menggabungkan beberapa jenis strategi pengujian ini.Misalnya, pengujian berbasis risiko (strategi analitis) dapat dikombinasikan dengan pengujian eksplorasi (strategi reaktif); mereka melengkapi satu sama lain dan dapat mencapai pengujian yang lebih efektif bila digunakan bersama-sama.

Sementara strategi pengujian memberikan gambaran umum dari proses pengujian, pendekatan pengujian menyesuaikan strategi pengujian untuk proyek atau rilis tertentu. Pendekatan tes adalah titik awal untuk memilih teknik tes, tingkat tes, dan jenis tes, dan untuk menentukan kriteria masuk dan kriteria keluar (atau definisi siap dan definisi selesai, masing-masing). Penyesuaian strategi didasarkan pada keputusan yang dibuat sehubungan dengan kompleksitas dan tujuan proyek, jenis produk yang dikembangkan, dan analisis risiko produk. Pendekatan yang dipilih bergantung pada konteks dan dapat mempertimbangkan faktor-faktor seperti risiko,
keselamatan, sumber daya dan keterampilan yang tersedia, teknologi, sifat sistem (misalnya, dibuat khusus *versus COTS*), tujuan pengujian, dan peraturan.

### 5.2.3 Kriteria Masuk dan Keluar (Definisi Siap dan Definisi Selesai)

Untuk melakukan kontrol yang efektif atas kualitas perangkat lunak, dan pengujian, disarankan untuk memiliki kriteria yang menentukan kapan aktivitas pengujian tertentu harus dimulai dan kapan aktivitas selesai. Kriteria masuk (lebih sering disebut definisi siap dalam pengembangan *Agile*) menentukan prasyarat untuk melakukan aktivitas pengujian tertentu. Jika kriteria masuk tidak terpenuhi, kemungkinan besar aktivitas itu akan terbukti lebih sulit, lebih memakan waktu, lebih mahal, dan lebih berisiko. Kriteria keluar (lebih sering disebut definisi selesai dalam pengembangan *Agile*) menentukan kondisi apa yang harus dicapai untuk menyatakan tingkat pengujian atau serangkaian pengujian selesai. Kriteria masuk dan keluar harus ditentukan untuk setiap tingkat pengujian dan jenis pengujian, dan akan berbeda berdasarkan tujuan pengujian.

Kriteria masuk yang umum meliputi:

- Ketersediaan persyaratan yang dapat diuji, cerita pengguna, dan/atau model (misalnya, saat mengikuti strategi pengujian berbasis model)
- Ketersediaan item tes yang telah memenuhi kriteria keluar untuk setiap tingkat tes Sebelumnya
- Ketersediaan lingkungan pengujian
- Ketersediaan peralatan pengujian yang diperlukan
- Ketersediaan data pengujian dan sumber daya lain yang diperlukan
  Kriteria keluar yang umum termasuk:
- Tes yang direncanakan telah dijalankan
- Tingkat cakupan yang ditentukan (misalnya, persyaratan, cerita pengguna, kriteria penerimaan, risiko, kode) telah tercapai
- Jumlah cacat yang belum terselesaikan berada dalam batas yang disepakati
- Jumlah estimasicacat yang tersisa cukup rendah
- Tingkat keandalan, efisiensi kinerja, kegunaan, keamanan, dan karakteristik kualitas lain yang relevan yang dievaluasi sudah cukup

Bahkan tanpa memenuhi kriteria keluar, biasanya juga aktivitas pengujian dibatasi karena anggaran yang dikeluarkan, waktu yang dijadwalkan selesai, dan/atau tekanan untuk membawa produk ke pasar. Mengakhiri pengujian dalam situasi seperti itu dapat diterima jika pemangku kepentingan proyek dan pemilik bisnis telah meninjau dan menerima risiko untuk ditayangkan tanpa pengujian lebih lanjut.

### 5.2.4 Jadwal Pelaksanaan Tes

Setelah berbagai kasus pengujian dan prosedur pengujian diproduksi (dengan beberapa prosedur pengujian berpotensi otomatis) dan dirakit menjadi rangkaian pengujian, rangkaian pengujian dapat diatur dalam jadwal pelaksanaan pengujian yang menentukan urutan pelaksanaannya. Jadwal pelaksanaan tes harus

mempertimbangkan faktor-faktor seperti prioritas, ketergantungan, tes konfirmasi, tes regresi, dan urutan yang paling efisien untuk melaksanakan tes.

Idealnya, kasus uji akan diperintahkan untuk dijalankan berdasarkan tingkat prioritasnya, biasanya dengan mengeksekusi kasus uji dengan prioritas tertinggi terlebih dahulu. Namun, praktik ini mungkin tidak berfungsi jika kasus uji memiliki ketergantungan atau fitur yang diuji memiliki ketergantungan. Jika kasus uji dengan prioritas lebih tinggi bergantung pada kasus uji dengan prioritas lebih rendah, kasus uji dengan prioritas lebih rendah harus dijalankan terlebih dahulu. Demikian pula, jika ada dependensi di seluruh kasus uji, mereka harus diurutkan dengan tepat terlepas dari prioritas relatifnya. Konfirmasi dan uji regresi harus diprioritaskan juga,

berdasarkan pentingnya umpan balik yang cepat terhadap perubahan, tetapi di sini sekali lagi dependensi mungkin berlaku.

Dalam beberapa kasus, berbagai rangkaian pengujian dimungkinkan, dengan tingkat efisiensi yang berbeda terkait dengan rangkaian tersebut. Dalam kasus seperti itu, pertukaran antara efisiensi pelaksanaan tes versus kepatuhan terhadap prioritas harus dibuat.

### 5.2.5 Faktor-Faktor yang Mempengaruhi Upaya Pengujian

Estimasi upaya pengujian melibatkan memprediksi jumlah pekerjaan terkait pengujian yang akan dibutuhkan untuk memenuhi tujuan pengujian untuk proyek, rilis, atau iterasi tertentu. Faktor-faktor yang mempengaruhi upaya pengujian dapat mencakup karakteristik produk, karakteristik proses pengembangan, karakteristik orang-orang, dan hasil pengujian, seperti yang ditunjukkan di bawah ini.

**Karakteristik produk**

- Risiko yang berhubungan dengan produk
- Kualitas dasar pengujian
- Ukuran dari produk
- Kompleksitas domain produk
- Persyaratan untuk karakteristik kualitas (misalnya, keamanan, keandalan)
- Tingkat detail yang diperlukan untuk dokumentasi pengujian
- Persyaratan untuk kepatuhan hukum dan peraturan

**Karakteristik proses pengembangan**

- Stabilitas dan kedewasaan organisasi
- Model pengembangan yang digunakan
- Pendekatan tes
- Peralatan yang digunakan
- Proses pengujian
- Tekanan waktu

**Karakteristik orang**

- Keterampilan dan pengalaman orang-orang yang terlibat, terutama dengan proyek dan produk serupa (misalnya, pengetahuan domain)
- Kekompakan dan kepemimpinan tim

**Hasil Tes**

- Jumlah dan beratnya cacat yang ditemukan
- Jumlah pengerjaan ulang yang dibutuhkan

### 5.2.6 Teknik Estimasi Pengujian

Ada beberapa teknik estimasi yang digunakan untuk menentukan upaya yang diperlukan untuk pengujian yang memadai. Dua teknik yang paling sering digunakan adalah:

- Teknik berbasis metrik: memperkirakan upaya pengujian berdasarkan metrik proyek serupa sebelumnya, atau berdasarkan nilai tipikal
- Teknik berbasis pakar: memperkirakan upaya pengujian berdasarkan pengalaman pemilik tugas pengujian atau oleh ahli

Misalnya, dalam pengembangan Agile, bagan *burndown* adalah contoh pendekatan berbasis metrik saat upaya yang tersisa dicatat dan dilaporkan, dan kemudian digunakan untuk memasukkan kecepatan tim untuk menentukan jumlah pekerjaan yang dapat dilakukan tim dalam iterasi berikutnya; sedangkan perencanaan poker adalah contoh pendekatan berbasis ahli, karena anggota tim memperkirakan upaya untuk memberikan fitur berdasarkan pengalaman mereka (ISTQB-CTFL-AT).

Dalam proyek berurutan, model penghapusan cacat adalah contoh pendekatan berbasis metrik, di mana volume cacat dan waktu untuk menghapusnya ditangkap dan dilaporkan, yang kemudian memberikan dasar untuk memperkirakan proyek masa depan dengan sifat yang serupa; sedangkan teknik estimasi *Wideband Delphi* adalah contoh dari pendekatan berbasis ahli/pakar di mana sekelompok ahli memberikan estimasiberdasarkan pengalaman mereka (ISTQB-CTAL-TM).

## 5.3 Pemantauan dan Kontrol Pengujian

Tujuan dari pemantauan pengujian adalah untuk mengumpulkan informasi dan memberikan umpan balik dan visibilitas tentang aktivitas pengujian. Informasi yang akan dipantau dapat dikumpulkan secara manual atau otomatis dan harus digunakan untuk menilai kemajuan pengujian dan untuk mengukur apakah kriteria uji keluar, atau tugas pengujian yang terkait dengan definisi proyek Agile yang telah selesai, terpenuhi, seperti memenuhi target cakupan risiko produk, persyaratan, atau kriteria penerimaan.

Kontrol pengujian menjelaskan setiap tindakan panduan atau perbaikan yang diambil sebagai hasil dari informasi dan metrik yang dikumpulkan dan (mungkin) dilaporkan. Tindakan dapat mencakup semua aktivitas pengujian dan dapat memengaruhi aktivitas siklus hidup perangkat lunak lainnya.

Contoh tindakan pengendalian pengujian meliputi:

- Memprioritaskan kembali pengujian saat risiko yang teridentifikasi terjadi (misalnya,. perangkat lunak terlambat dikirim)
- Mengubah jadwal pengujian karena ketersediaan atau ketidaktersediaan lingkungan pengujian atau sumber daya lainnya
- Mengevaluasi kembali apakah barang pengujian memenuhi kriteria masuk atau keluar karena pengerjaan ulang

### 5.3.1 Metrik yang Digunakan dalam Pengujian

Metrik dapat dikumpulkan selama dan pada akhir aktivitas pengujian untuk menilai:

- Kemajuan terhadap jadwal dan anggaran yang direncanakan
- Kualitas benda uji saat ini
- Kecukupan dari tes pendekatan
- Keefektifan aktivitas pengujian sehubungan dengan tujuan

Metrik pengujian umum meliputi:

- Persentase pekerjaan yang direncanakan yang dilakukan dalam persiapan kasus uji (atau persentase kasus uji yang direncanakan dilaksanakan)
- Persentase pekerjaan yang direncanakan yang dilakukan dalam persiapan lingkungan pengujian
- Eksekusi kasus uji (misalnya., jumlah kasus uji yang dijalankan/tidak dijalankan, kasus uji yang lulus/gagal, dan/atau kasus uji syarat lulus/gagal)
- Informasi cacat (misalnya, kepadatan cacat, cacat ditemukan dan diperbaiki, tingkat kegagalan, dan uji konfirmasi hasil)
- Uji cakupan persyaratan, cerita pengguna, kriteria penerimaan, risiko, atau kode
- Penyelesaian tugas, alokasi dan penggunaan sumber daya, dan upaya
- Biaya pengujian, termasuk biaya dibandingkan dengan manfaat menemukan cacat berikutnya atau biaya dibandingkan dengan manfaat menjalankan tes berikutnya

### 5.3.2 Tujuan, Isi, dan Audiens untuk Laporan Pengujian

Tujuan dari pelaporan pengujian adalah untuk meringkas dan mengkomunikasikan informasi aktivitas pengujian, baik selama dan pada akhir aktivitas pengujian (misalnya, tingkat pengujian). Laporan pengujian yang disiapkan selama aktivitas pengujian dapat disebut sebagai laporan kemajuan pengujian, sementara laporan pengujian yang disiapkan pada akhir aktivitas pengujian dapat disebut sebagai laporan ringkasan pengujian.

Selama pemantauan dan kontrol pengujian, manajer pengujian secara teratur mengeluarkan laporan kemajuan pengujian untuk pemangku kepentingan. Selain konten yang umum untuk laporan kemajuan pengujian dan laporan ringkasan pengujian, laporan kemajuan pengujian yang umum juga dapat mencakup:

- Status aktivitas pengujian dan kemajuan terhadap rencana pengujian
- Faktor-faktor yang menghambat kemajuan
- Pengujian direncanakan untuk periode pelaporan berikutnya
- Kualitas objek tes

Ketika kriteria keluar tercapai, manajer pengujian mengeluarkan laporan ringkasan pengujian. Laporan ini memberikan ringkasan pengujian yang dilakukan, berdasarkan laporan kemajuan pengujian terbaru dan informasi lain yang relevan.

Laporan ringkasan pengujian yang umum dapat mencakup:

- Ringkasan pengujian yang dilakukan
- Informasi tentang apa yang terjadi selama periode pengujian
- Penyimpangan dari rencana, termasuk penyimpangan dalam jadwal, durasi, atau upaya aktivitas pengujian
- Status pengujian dan kualitas produk sehubungan dengan kriteria keluar atau definisi selesai
- Faktor yang menghalangi atau terus menghalangi kemajuan
- Metrik cacat, kasus uji, cakupan pengujian, kemajuan aktivitas, dan konsumsi sumber daya. (misalnya, seperti yang dijelaskan dalam 5.3.1)
- Risiko residual (lihat bagian 5.5)
- Produk kerja uji yang dihasilkan dapat digunakan kembali

Isi laporan pengujian akan bervariasi tergantung pada proyek, persyaratan organisasi, dan siklus pengembangan perangkat lunak. Misalnya, sebuah proyek yang kompleks dengan banyak pemangku kepentingan atau proyek yang diatur mungkin memerlukan pelaporan yang lebih rinci dan ketat daripada pembaruan perangkat lunak yang cepat. Sebagai contoh lain, dalam pengembangan Agile, pelaporan kemajuan pengujian dapat dimasukkan ke dalam papan tugas, ringkasan cacat, dan bagan burndown, yang dapat didiskusikan selama pertemuan stand-up harian (lihat ISTQBCTFL-AT).

Selain menyesuaikan laporan pengujian berdasarkan konteks proyek, laporan pengujian harus disesuaikan berdasarkan audiensi laporan. Jenis dan jumlah informasi yang harus disertakan untuk audiensi teknis atau tim penguji mungkin berbeda dari apa yang akan dimasukkan dalam laporan ringkasan eksekutif. Dalam kasus pertama, informasi rinci tentang jenis dan tren cacat mungkin penting. Dalam kasus terakhir, laporan tingkat tinggi (misalnya, ringkasan status cacat berdasarkan prioritas, anggaran, jadwal, dan kondisi pengujian yang lulus/gagal/tidak diuji) mungkin lebih tepat.

Standar ISO (ISO/IEC/IEEE 29119-3) mengacu pada dua jenis laporan pengujian, laporan kemajuan pengujian dan laporan penyelesaian pengujian (disebut laporan ringkasan pengujian dalam silabus ini), dan berisi struktur dan contoh untuk masing-masing jenis.

## 5.4 Manajemen Konfigurasi

Tujuan dari manajemen konfigurasi adalah untuk menetapkan dan memelihara integritas komponen atau sistem, perangkat uji, dan hubungannya satu sama lain melalui siklus hidup proyek dan produk.

Untuk mendukung pengujian dengan benar, manajemen konfigurasi mungkin melibatkan memastikan hal-hal berikut:

- Semua item tes diidentifikasi secara unik, dikontrol versi, dilacak perubahannya, dan terkait satu sama lain
- Semua item perangkat uji diidentifikasi secara unik, dikontrol versi, dilacak untuk perubahan, terkait satu sama lain, dan terkait dengan versi item uji sehingga keterlacakan dapat dipertahankan selama proses pengujian
- Semua dokumen dan item perangkat lunak yang diidentifikasi direferensikan dengan jelas dalam dokumentasi pengujian

selama perencanaan pengujian, prosedur manajemen konfigurasi dan infrastruktur (peralatan) harus diidentifikasi dan diterapkan.

## 5.5 Risiko dan Pengujian

### 5.5.1 Definisi Risiko

Risiko melibatkan kemungkinan suatu peristiwa di masa depan yang memiliki konsekuensi negatif. Tingkat risiko ditentukan oleh kemungkinan terjadinya peristiwa dan dampak (kerugian) dari peristiwa tersebut.

### 5.5.2 Risiko Produk dan Proyek

Risiko produk melibatkan kemungkinan bahwa produk kerja (misalnya, spesifikasi, komponen, sistem, atau pengujian) mungkin gagal untuk memenuhi kebutuhan yang sah dari pengguna dan/atau pemangku kepentingan. Ketika risiko produk dikaitkan dengan karakteristik kualitas tertentu dari suatu produk (misalnya, kesesuaian fungsional, keandalan, efisiensi kinerja, kegunaan, keamanan, kompatibilitas, pemeliharaan, dan portabilitas), risiko produk juga disebut risiko kualitas. Contoh risiko produk meliputi:

- Perangkat lunak mungkin tidak menjalankan fungsi yang dimaksudkan sesuai dengan spesifikasi
- Perangkat lunak mungkin tidak menjalankan fungsinya sesuai dengan kebutuhan pengguna, pelanggan, dan/atau pemangku kepentingan
- Arsitektur sistem mungkin tidak cukup mendukung beberapa persyaratan non-fungsional
- Perhitungan tertentu dapat dilakukan secara tidak benar dalam beberapa Keadaan
- Struktur kontrol loop mungkin salah dikodekan
- Waktu respons mungkin tidak memadai untuk sistem pemrosesan transaksi berkinerja tinggi
- Umpan balik pengalaman pengguna (UX) mungkin tidak memenuhi harapan Produk

Risiko proyek melibatkan situasi yang, jika terjadi, dapat berdampak negatif pada kemampuan proyek untuk mencapai tujuannya. Contoh risiko proyek meliputi:

- Masalah proyek:
  - Keterlambatan dapat terjadi dalam pengiriman, penyelesaian tugas, atau kepuasan kriteria keluar atau definisi selesai
  - Estimasiyang tidak akurat, realokasi dana ke proyek dengan prioritas lebih tinggi, atau pemotongan biaya umum di seluruh organisasi dapat mengakibatkan pendanaan yang tidak memadai
  - Perubahan yang terlambat dapat mengakibatkan pengerjaan ulang yang substantial
- Masalah organisasi:
  - Keterampilan, pelatihan, dan staf mungkin tidak cukup
  - Masalah personel dapat menyebabkan konflik dan masalah
  - Pengguna, staf bisnis, atau pakar materi pelajaran mungkin tidak tersedia karena prioritas bisnis yang bertentangan
- Isu-isu politik:
  - Penguji mungkin tidak mengomunikasikan kebutuhan mereka dan/atau hasil tes secara memadai
  - Pengembang dan/atau penguji mungkin gagal menindaklanjuti informasi yang ditemukan dalam pengujian dan tinjauan (misalnya, tidak meningkatkan praktik pengembangan dan pengujian)
  - Mungkin ada sikap yang tidak tepat terhadap, atau harapan, pengujian (misalnya, tidak menghargai nilai menemukan cacat selama pengujian)
- Masalah teknis:
  - Persyaratan mungkin tidak didefinisikan dengan cukup baik
  - Persyaratan mungkin tidak terpenuhi, mengingat kendala yang ada
  - Lingkungan pengujian mungkin tidak siap tepat waktu
  - Konversi data, perencanaan migrasi, dan dukungan peralatannya mungkin terlambat
  - Kelemahan dalam proses pengembangan dapat mempengaruhi konsistensi atau kualitas produk pekerjaan proyek seperti desain, kode, konfigurasi, data uji, dan kasus uji
  - Manajemen cacat yang buruk dan masalah serupa dapat mengakibatkan akumulasi cacat dan utang teknis lainnya
- Masalah pemasok:
  - Pihak ketiga mungkin gagal memberikan produk atau layanan yang diperlukan, atau bangkrut
  - Masalah kontraktual dapat menyebabkan masalah pada proyek

Risiko proyek dapat mempengaruhi baik aktivitas pengembangan maupun aktivitas pengujian. Dalam beberapa kasus, manajer proyek bertanggung jawab untuk menangani semua risiko proyek, tetapi tidak jarang manajer pengujian memiliki tanggung jawab untuk risiko proyek terkait pengujian.

### 5.5.3 Pengujian Berbasis Risiko dan Kualitas Produk

Risiko digunakan untuk memfokuskan upaya yang diperlukan selama pengujian. Ini digunakan untuk memutuskan di mana dan kapan memulai pengujian dan untuk mengidentifikasi area yang membutuhkan perhatian lebih. Pengujian digunakan untuk mengurangi kemungkinan yang merugikan peristiwa yang terjadi, atau untuk mengurangi dampak dari peristiwa yang merugikan. Pengujian digunakan sebagai

aktivitas mitigasi risiko, untuk memberikan informasi tentang risiko yang teridentifikasi, serta memberikan informasi tentang risiko residual (belum terselesaikan).

Pendekatan berbasis risiko untuk pengujian memberikan peluang proaktif untuk mengurangi tingkat risiko produk. Ini melibatkan analisis risiko produk, yang mencakup identifikasi risiko produk dan penilaian kemungkinan dan dampak setiap risiko. Informasi risiko produk yang dihasilkan digunakan untuk memandu perencanaan pengujian, spesifikasi, persiapan dan pelaksanaan kasus uji, serta pemantauan dan pengendalian pengujian. Menganalisis risiko produk lebih awal berkontribusi pada keberhasilan suatu proyek.

Dalam pendekatan berbasis risiko, hasil analisis risiko produk digunakan untuk:

- Tentukan teknik tes yang akan digunakan
- Menentukan tingkat dan jenis pengujian tertentu yang akan dilakukan (misalnya, pengujian keamanan, pengujian aksesibilitas)
- Tentukan sejauh mana pengujian yang akan dilakukan
- Prioritaskan pengujian dalam upaya untuk menemukan cacat kritis sedini mungkin
- Tentukan apakah ada aktivitas selain pengujian yang dapat digunakan untuk mengurangi risiko (misalnya, memberikan pelatihan kepada desainer yang tidak berpengalaman)

Pengujian berbasis risiko mengacu pada pengetahuan dan wawasan kolektif dari pemangku kepentingan proyek untuk melakukan analisis risiko produk. Untuk memastikan bahwa kemungkinan kegagalan produk diminimalkan, manajemen risiko aktivitas memberikan pendekatan disiplin untuk:

- Analisis (dan evaluasi ulang secara teratur) apa yang bisa salah (risiko)
- Tentukan risiko mana yang penting untuk dihadapi
- Menerapkan tindakan untuk mengurangi risiko tersebut
- Membuat rencana kontinjensi untuk menghadapi risiko jika risiko tersebut menjadi kejadian nyata

Selain itu, pengujian dapat mengidentifikasi risiko baru, membantu menentukan risiko apa yang harus dikurangi, dan menurunkan ketidakpastian tentang risiko.

## 5.6 Manajemen Cacat

Karena salah satu tujuan pengujian adalah untuk menemukan cacat, cacat yang ditemukan selama pengujian harus dicatat. Cara cacat dicatat dapat bervariasi, tergantung pada konteks komponen atau sistem yang diuji, tingkat pengujian, dan model siklus hidup pengembangan perangkat lunak. Setiap cacat yang diidentifikasi harus diselidiki dan harus dilacak dari penemuan dan klasifikasi hingga penyelesaiannya (misalnya, koreksi cacat dan pengujian konfirmasi yang berhasil dari solusi, penundaan untuk rilis berikutnya, penerimaan sebagai batasan produk permanen, dll.). Untuk mengelola semua cacat hingga resolusi, organisasi harus menetapkan proses manajemen cacat yang mencakup alur kerja dan aturan untuk klasifikasi. Proses ini harus disetujui oleh semua pihak yang berpartisipasi dalam manajemen cacat, termasuk arsitek, perancang, pengembang, penguji, dan pemilik produk. Di beberapa organisasi, pencatatan cacat dan pelacakan mungkin sangat informal.

Selama proses manajemen cacat, beberapa laporan mungkin berubah menjadi positif palsu, bukan kegagalan aktual karena cacat. Misalnya, tes mungkin gagal saat koneksi jaringan terputus atau waktu habis. Perilaku ini bukan hasil dari cacat pada benda uji, tetapi merupakan anomali yang perlu diselidiki. Penguji harus berusaha meminimalkan jumlah positif palsu yang dilaporkan sebagai cacat.

Cacat dapat dilaporkan selama pengkodean, analisis statis, tinjauan, atau selama pengujian dinamis, atau penggunaan produk perangkat lunak. Cacat dapat dilaporkan untuk masalah dalam kode atau sistem kerja, atau dalam jenis dokumentasi termasuk persyaratan, cerita pengguna dan kriteria penerimaan, dokumen pengembangan, dokumen uji, manual pengguna, atau panduan instalasi. Untuk memiliki cacat yang efektif dan efisien proses manajemen, organisasi dapat menentukan standar untuk atribut, klasifikasi, dan alur kerja cacat.

Laporan cacat tipikal memiliki tujuan sebagai berikut:

- Memberikan informasi kepada pengembang dan pihak lain tentang setiap kejadian buruk yang terjadi, untuk memungkinkan mereka mengidentifikasi efek spesifik, untuk mengisolasi masalah dengan uji reproduksi minimal, dan untuk memperbaiki potensi cacat, sesuai kebutuhan atau untuk menyelesaikan masalah
- Memberikan manajer pengujian sarana untuk melacak kualitas produk kerja dan dampaknya terhadap pengujian (misalnya, jika banyak cacat dilaporkan, penguji akan menghabiskan banyak waktu untuk melaporkannya alih-alih menjalankan tes, dan akan ada lebih banyak pengujian konfirmasi yang diperlukan)
- Memberikan ide untuk pengembangan dan peningkatan proses pengujian

Laporan cacat yang diajukan selama pengujian dinamis biasanya mencakup:

- Pengidentifikasi
- Judul dan ringkasan singkat dari cacat yang dilaporkan
- Tanggal laporan cacat, organisasi penerbit, dan penulis
- Identifikasi item tes (item konfigurasi yang diuji) dan lingkungan
- Fase siklus hidup pengembangan di mana cacat diamati
- Denaskahsi cacat untuk memungkinkan reproduksi dan resolusi, termasuk catatan, dump database, tangkapan layar, atau rekaman (jika ditemukan selama pelaksanaan pengujian)
- Hasil yang diharapkan dan aktual
- Cakupan atau tingkat dampak (keparahan) cacat pada kepentingan pemangku kepentingan
- Urgensi/prioritas untuk diperbaiki
- Status laporan cacat (misalnya, terbuka, ditangguhkan, duplikat, menunggu untuk diperbaiki, menunggu pengujian konfirmasi, dibuka kembali, ditutup)

- Kesimpulan, rekomendasi dan persetujuan
- Masalah global, seperti area lain yang mungkin terpengaruh oleh perubahan akibat cacat
- Riwayat perubahan, seperti urutan tindakan yang diambil oleh anggota tim proyek sehubungan dengan cacat untuk mengisolasi, memperbaiki, dan mengonfirmasinya sebagai telah diperbaiki
- Referensi, termasuk kasus uji yang mengungkapkan masalah

Beberapa detail ini mungkin disertakan dan/atau dikelola secara otomatis saat menggunakan peralatan manajemen cacat, misalnya, penetapan otomatis pengidentifikasi, penetapan dan pembaruan status laporan cacat selama alur kerja, dll.

Cacat yang ditemukan selama pengujian statis, khususnya tinjauan, akan biasanya didokumentasikan dengan cara yang berbeda, misalnya, dalam tinjauan catatan rapat.

Contoh isi laporan cacat dapat ditemukan dalam standar ISO (ISO/IEC/IEEE 29119-3) (yang mengacu pada laporan cacat sebagai laporan atas kejadian).

| 6 Dukungan Peralatan untuk Pengujian | 40 menit |
|---|---|

**Kata kunci**

pengujian berbasis data, pengujian berdasarkan kata kunci, otomatisasi pengujian, peralatan pelaksanaan pengujian, peralatan manajemen pengujian

**Tujuan Pembelajaran untuk Peralatan pengujian**

## 6.1 Pertimbangan peralatan pengujian

FL-6.1.1 (K2) Mengklasifikasikan peralatan pengujian menurut tujuannya dan aktivitas pengujian yang didukungnya

FL-6.1.2 (K1) Mengidentifikasi manfaat dan risiko otomatisasi pengujian

FL-6.1.3 (K1) Ingat pertimbangan khusus untuk pelaksanaan pengujian dan peralatan manajemen pengujian

## 6.2 Penggunaan peralatan yang efektif

FL-6.2.1 (K1) Identifikasi prinsip-prinsip utama untuk memilih peralatan

FL-6.2.2 (K1) Ingat kembali tujuan penggunaan proyek percontohan untuk memperkenalkan peralatan

FL-6.2.3 (K1) Mengidentifikasi faktor-faktor keberhasilan untuk evaluasi, implementasi, penerapan, dan dukungan peralatan pengujian yang berkelanjutan dalam suatu organisasi

## 6.1 Pertimbangan Peralatan pengujian

Peralatan pengujian dapat digunakan untuk mendukung satu atau lebih aktivitas pengujian. Peralatan tersebut meliputi:

- Peralatan yang langsung digunakan dalam pengujian, seperti peralatan eksekusi pengujian dan peralatan persiapan data pengujian
- Peralatan yang membantu mengelola persyaratan, kasus pengujian, prosedur pengujian, naskah pengujian otomatis, hasil pengujian, data pengujian, dan cacat, serta untuk pelaporan dan pemantauan pelaksanaan pengujian
- Perangkat yang digunakan untuk analisis dan evaluasi
- Setiap peralatan yang membantu dalam pengujian (lembar kerja juga merupakan peralatan pengujian dalam arti ini)

### 6.1.1 Klasifikasi Peralatan pengujian

Peralatan pengujian dapat memiliki satu atau beberapa tujuan berikut tergantung pada konteksnya:

- Meningkatkan efisiensi aktivitas pengujian dengan mengotomatiskan tugas yang berulang atau tugas yang memerlukan sumber daya yang signifikan bila dilakukan secara manual (misalnya, pelaksanaan pengujian, pengujian regresi)
- Meningkatkan efisiensi aktivitas pengujian dengan mendukung aktivitas pengujian manual selama pengujian proses (lihat bagian 1.4)
- Meningkatkan kualitas aktivitas pengujian dengan memungkinkan pengujian yang lebih konsisten dan tingkat. yang lebih tinggi reproduktifitas cacat
- Mengotomatiskan aktivitas yang tidak dapat dijalankan secara manual (misalnya. pengujian kinerja skala besar)
- Meningkatkan keandalan pengujian (misalnya, dengan mengotomatiskan perbandingan data yang besar atau mensimulasikan perilaku)

Peralatan dapat diklasifikasikan berdasarkan beberapa kriteria seperti tujuan, harga, model lisensi (misalnya, komersial atau ***open source***), dan teknologi yang digunakan. Peralatan-peralatan diklasifikasikan dalam silabus ini sesuai dengan aktivitas pengujian yang didukungnya.

Beberapa peralatan jelas hanya mendukung atau terutama satu aktivitas; yang lain mungkin mendukung lebih dari satu aktivitas, tetapi diklasifikasikan di bawah aktivitas yang paling erat hubungannya dengan aktivitas tersebut. Peralatan dari satu penyedia, terutama yang telah dirancang untuk bekerja sama, dapat disediakan sebagai rangkaian terintegrasi.

Beberapa jenis peralatan pengujian dapat mengganggu, yang berarti bahwa peralatan tersebut dapat memengaruhi hasil pengujian yang sebenarnya. Sebagai contoh, waktu respons yang sebenarnya untuk aplikasi mungkin berbeda karena instruksi tambahan yang dijalankan oleh peralatan pengujian kinerja, atau jumlah cakupan kode yang dicapai mungkin terdistorsi karena penggunaan peralatan cakupan. Konsekuensi dari penggunaan peralatan intrusif disebut efek probe.

Beberapa peralatan menawarkan dukungan yang biasanya lebih sesuai untuk pengembang (misalnya, peralatan yang digunakan selama pengujian komponen dan integrasi). Peralatan tersebut ditandai dengan "(D)" di bagian bawah.

**Dukungan peralatan untuk manajemen pengujian dan perangkat uji**

Peralatan manajemen mungkin berlaku untuk aktivitas pengujian apa pun di seluruh siklus hidup pengembangan perangkat lunak. Contoh peralatan yang mendukung pengelolaan pengujian dan perangkat uji meliputi:

- Peralatan manajemen pengujian dan peralatan manajemen siklus hidup aplikasi (ALM)
- Peralatan manajemen persyaratan (misalnya, ketertelusuran untuk menguji objek)
- Peralatan pengelolaan cacat
- Peralatan manajemen konfigurasi
- Peralatan integrasi berkelanjutan (D)

**Dukungan peralatan untuk pengujian statis**

Peralatan pengujian statis terkait dengan aktivitas dan manfaat yang dijelaskan dalam bab 3. Contoh peralatan tersebut meliputi:

- Peralatan analisis statis (D)

**Dukungan peralatan untuk desain dan implementasi pengujian**

Peralatan desain pengujian membantu dalam pembuatan produk kerja yang dapat dipelihara dalam desain dan implementasi pengujian, termasuk kasus pengujian, prosedur pengujian, dan data pengujian. Contoh peralatan tersebut meliputi:

- Peralatan pengujian Berbasis Model
- Peralatan persiapan data uji

Dalam beberapa kasus, peralatan yang mendukung desain dan implementasi pengujian juga dapat mendukung eksekusi dan pencatatan pengujian, atau memberikan outputnya langsung ke peralatan lain yang mendukung eksekusi dan pencatatan pengujian.

**Dukungan peralatan untuk eksekusi pengujian dan pencatatan**

Ada banyak peralatan untuk mendukung dan meningkatkan pelaksanaan pengujian dan aktivitas pencatatan. Contoh peralatan ini antara lain:

- Peralatan eksekusi pengujian (misalnya, untuk menjalankan pengujian regresi)
- Peralatan cakupan (misalnya, cakupan persyaratan, cakupan kode (D))
- Uji pengendali (D)

**Dukungan peralatan untuk pengukuran kinerja dan analisis dinamis**

Peralatan ukur kinerja dan analisis dinamik sangat penting dalam mendukung kinerja dan aktivitas pengujian beban, karena aktivitas ini tidak dapat dilakukan secara manual secara efektif. Contoh peralatan tersebut antara lain:

- Peralatan pengujian kinerja
- Peralatan analisis dinamis (D)

**Dukungan peralatan untuk keperluan pengujian khusus**

Selain peralatan-peralatan yang mendukung proses pengujian secara umum, ada banyak peralatan lain yang mendukung pengujian yang lebih khusus untuk karakteristik non-fungsional.

### 6.1.2 Manfaat dan Resiko Otomasi Pengujian

Mendapatkan peralatan saja tidak menjamin kesuksesan. Setiap peralatan baru yang diperkenalkan ke dalam organisasi akan membutuhkan upaya untuk mencapai manfaat yang nyata dan bertahan lama. Ada manfaat dan peluang potensial dengan penggunaan peralatan dalam pengujian, tetapi ada juga risiko. Hal ini terutama berlaku untuk peralatan eksekusi pengujian (yang sering disebut sebagai otomatisasi pengujian).

Manfaat potensial dari penggunaan peralatan untuk mendukung eksekusi pengujian meliputi:

- Pengurangan pekerjaan manual yang berulang (misalnya., menjalankan tes regresi, tugas pengaturan/meruntuhkan lingkungan, memasukkan kembali data pengujian yang sama, dan memeriksa standar pengkodean), sehingga menghemat waktu
- Konsistensi dan pengulangan yang lebih baik (misalnya, data pengujian dibuat dengan cara yang koheren, pengujian dijalankan oleh peralatan dalam urutan yang sama dengan frekuensi yang sama, dan pengujian secara konsisten diturunkan dari persyaratan)
- Penilaian yang lebih objektif (misalnya, pengukuran statis, cakupan)
- Akses yang lebih mudah ke informasi tentang pengujian (misalnya., statistik dan grafik tentang kemajuan pengujian, cacat tarif dan kinerja)

Risiko potensial menggunakan peralatan untuk mendukung pengujian meliputi:

- Harapan untuk peralatan ini mungkin tidak realistis (termasuk fungsionalitas dan kemudahan penggunaan)
- Waktu, biaya, dan upaya untuk pengenalan awal suatu peralatan mungkin kurang dari estimasi(termasuk pelatihan dan keahlian eksternal)
- Waktu dan upaya yang diperlukan untuk mencapai manfaat yang signifikan dan berkelanjutan dari peralatan ini mungkin kurang diperhitungkan (termasuk kebutuhan untuk perubahan dalam proses pengujian dan perbaikan terus-menerus dalam cara peralatan digunakan)
- Upaya yang diperlukan untuk mempertahankan produk kerja pengujian yang dihasilkan oleh peralatan ini mungkin dianggap remeh
- Peralatan ini mungkin terlalu diandalkan (dilihat sebagai pengganti desain atau eksekusi pengujian, atau penggunaan pengujian otomatis di mana pengujian manual akan lebih baik)
- Kontrol versi produk kerja uji mungkin diabaikan
- Masalah hubungan dan interoperabilitas antara peralatan-peralatan penting dapat diabaikan, seperti peralatan manajemen persyaratan, peralatan manajemen konfigurasi, peralatan manajemen cacat, dan peralatan dari banyak vendor
- Vendor peralatan mungkin gulung tikar, menghentikan peralatan, atau menjual peralatan ke vendor lain
- Vendor mungkin memberikan tanggapan yang buruk untuk dukungan, peningkatan, dan perbaikan cacat
- Proyek sumber terbuka mungkin ditangguhkan
- Sebuah platform atau teknologi baru mungkin tidak didukung oleh peralatan ini
- Mungkin tidak ada kepemilikan yang jelas dari peralatan tersebut (misalnya, untuk mentoring, pembaruan, dll.)

### 6.1.3 Pertimbangan Khusus untuk Pelaksanaan pengujian dan Peralatan Manajemen Tes

Agar pelaksanaannya dapat berjalan dengan lancar dan sukses, ada beberapa hal yang harus dipertimbangkan ketika memilih dan mengintegrasikan pelaksanaan pengujian dan peralatan manajemen pengujian ke dalam suatu organisasi.

### Peralatan eksekusi uji

Peralatan eksekusi pengujian mengeksekusi objek pengujian menggunakan naskah pengujian otomatis. Jenis peralatan ini seringkali membutuhkan upaya yang signifikan untuk mencapai manfaat yang signifikan.

- **Pendekatan uji tangkap**: Menangkap pengujian dengan merekam tindakan penguji manual tampaknya menarik, tetapi pendekatan ini tidak berskala ke

  sejumlah besar naskah pengujian. Naskah yang diambil adalah representasi linier dengan data dan tindakan spesifik sebagai bagian dari setiap naskah. Jenis naskah ini mungkin tidak stabil Ketika terjadi peristiwa yang tidak terduga, dan memerlukan pemeliharaan berkelanjutan karena antarmuka pengguna sistem berkembang seiring waktu.

- **Pendekatan pengujian berbasis data:** Pendekatan pengujian ini memisahkan input pengujian dan hasil yang diharapkan, biasanya ke dalam lembar kerja, dan menggunakan naskah pengujian yang lebih umum yang dapat membaca data input dan menjalankan naskah pengujian yang sama dengan data yang berbeda.

- **Pendekatan pengujian berbasis kata kunci:** Pendekatan pengujian ini, naskah generik memproses kata kunci yang menjelaskan tindakan yang akan diambil (juga disebut kata tindakan), yang kemudian memanggil naskah kata kunci untuk memproses data pengujian terkait.

Pendekatan di atas membutuhkan seseorang untuk memiliki keahlian dalam bahasa scripting (penguji, pengembang atau spesialis dalam otomatisasi pengujian). Saat menggunakan pendekatan pengujian berbasis data atau kata kunci, penguji yang tidak terbiasa dengan bahasa naskah juga dapat berkontribusi dengan membuat data pengujian dan/atau kata kunci untuk naskah yang telah ditentukan ini. Terlepas dari teknik naskah yang digunakan, hasil yang diharapkan untuk setiap pengujian perlu dibandingkan dengan hasil sebenarnya dari pengujian, baik secara dinamis (saat pengujian berjalan) atau disimpan untuk perbandingan nanti (pasca-eksekusi).

Rincian lebih lanjut dan contoh pendekatan pengujian yang digerakkan oleh data dan kata kunci diberikan dalam ISTQBCTAL-TAE, Fewster 1999 dan Buwalda 2001.

Peralatan pengujian Berbasis Model (MBT) memungkinkan spesifikasi fungsional ditangkap dalam bentuk model, seperti diagram aktivitas. Tugas ini umumnya dilakukan oleh perancang sistem. Peralatan MBT menginterpretasikan model untuk membuat spesifikasi kasus uji yang kemudian dapat disimpan dalam peralatan manajemen pengujian dan/atau dijalankan oleh peralatan eksekusi uji (lihat ISTQB-CTFL-MBT).

### Peralatan manajemen tes

Peralatan manajemen pengujian sering kali perlu berinteraksi dengan peralatan atau lembar kerja lain karena berbagai alasan, termasuk:

- Menghasilkan informasi yang berguna dalam format yang sesuai dengan kebutuhan organisasi

- Untuk menjaga ketertelusuran yang konsisten ke persyaratan dalam peralatan manajemen persyaratan

- Untuk menautkan dengan informasi versi objek uji di peralatan manajemen Konfigurasi

Hal ini sangat penting untuk dipertimbangkan ketika menggunakan peralatan terintegrasi (misalnya, Manajemen Siklus Hidup Aplikasi), yang mencakup modul

manajemen pengujian, serta modul lain (misalnya, jadwal proyek dan informasi anggaran) yang digunakan oleh kelompok yang berbeda dalam suatu organisasi.

## 6.2 Penggunaan Peralatan Secara Efektif

### 6.2.1 Prinsip Utama untuk Pemilihan Peralatan

Pertimbangan utama dalam memilih peralatan untuk organisasi meliputi:

- Penilaian kematangan organisasi itu sendiri, kekuatan dan kelemahannya
- Identifikasi peluang untuk proses pengujian yang lebih baik yang didukung oleh peralatan
- Pemahaman tentang teknologi yang digunakan oleh objek uji, untuk memilih peralatan yang kompatibel dengan teknologi itu
- Memahami peralatan bantu build dan integrasi berkelanjutan yang sudah digunakan dalam organisasi, untuk memastikan kompatibilitas dan integrasi peralatan
- Evaluasi peralatan terhadap persyaratan yang jelas dan kriteria yang objektif
- Pertimbangan apakah peralatan tersebut tersedia untuk periode uji coba gratis (dan berapa lama)
- Evaluasi vendor (termasuk pelatihan, dukungan, dan aspek komersial) atau dukungan untuk peralatan nonkomersial (misalnya., sumber terbuka)
- Identifikasi kebutuhan internal untuk pembinaan dan pendampingan dalam penggunaan peralatan ini
- Evaluasi kebutuhan pelatihan, dengan mempertimbangkan keterampilan pengujian (dan otomatisasi pengujian) dari mereka yang akan bekerja secara langsung dengan peralatan tersebut
- Pertimbangan pro dan kontra dari berbagai model perizinan (misalnya, komersial atau sumber terbuka)
- Estimasi rasio biaya-manfaat berdasarkan kasus bisnis konkret (jika diperlukan)

Sebagai langkah terakhir, evaluasi bukti dari konsep harus dilakukan untuk menetapkan apakah peralatan bekerja secara efektif dengan perangkat lunak yang diuji dan di dalam infrastruktur saat ini atau, jika perlu, untuk mengidentifikasi perubahan yang diperlukan pada infrastruktur tersebut untuk menggunakan peralatan secara efektif.

### 6.2.2 Proyek Percontohan untuk Memperkenalkan Perangkat ke dalam Organisasi

Setelah menyelesaikan pemilihan peralatan dan bukti konsep yang berhasil, memperkenalkan peralatan yang dipilih ke dalam organisasi biasanya dimulai dengan proyek percontohan, yang memiliki tujuan berikut:

- Mendapatkan pengetahuan mendalam tentang peralatan ini, memahami kekuatan dan kelemahannya
- Mengevaluasi kesesuaian peralatan dengan proses dan praktik yang ada, dan menentukan apa yang perlu diubah
- Memutuskan cara standar untuk menggunakan, mengelola, menyimpan, dan memelihara peralatan dan produk kerja pengujian (misalnya, memutuskan konvensi penamaan untuk file dan pengujian, memilih standar pengkodean, membuat perpustakaan, dan menentukan modularitas rangkaian pengujian)
- Menilai apakah manfaat akan dicapai dengan biaya yang wajar
- Memahami metrik yang Anda inginkan untuk dikumpulkan dan dilaporkan oleh peralatan, dan mengonfigurasi peralatan untuk memastikan metrik ini dapat ditangkap dan dilaporkan

### 6.2.3 Faktor Keberhasilan Peralatan

Faktor keberhasilan untuk evaluasi, implementasi, penerapan, dan dukungan peralatan yang berkelanjutan dalam suatu organisasi meliputi:

- Meluncurkan peralatan ini ke seluruh organisasi secara bertahap
- Menyesuaikan dan meningkatkan proses agar sesuai dengan penggunaan peralatan
- Memberikan pelatihan, pembinaan, dan pendampingan bagi pengguna peralatan
- Mendefinisikan pedoman untuk penggunaan peralatan (misalnya, standar internal untuk otomatisasi)
- Menerapkan cara untuk mengumpulkan informasi penggunaan dari penggunaan sebenarnya dari peralatan tersebut
- Penggunaan dan manfaat peralatan pemantauan
- Memberikan dukungan kepada pengguna dari peralatan yang diberikan
- Mengumpulkan pelajaran yang diperoleh dari semua pengguna

Penting juga untuk memastikan bahwa peralatan tersebut terintegrasi secara teknis dan organisasi ke dalam siklus hidup pengembangan perangkat lunak, yang mungkin melibatkan organisasi terpisah yang bertanggung jawab atas operasi dan/atau pemasok pihak ketiga. Lihat Graham 2012 untuk mendapatkan pengalaman dan saran tentang penggunaan peralatan eksekusi pengujian.

# 7 Referensi

## *Standar*

ISO/IEC/IEEE 29119-1 (2013) Software and systems engineering - Software testing - Part 1: Concepts and definitions

ISO/IEC/IEEE 29119-2 (2013) Software and systems engineering - Software testing - Part 2: Test processes

ISO/IEC/IEEE 29119-3 (2013) Software and systems engineering - Software testing - Part 3: Test documentation

ISO/IEC/IEEE 29119-4 (2015) Software and systems engineering - Software testing - Part 4: Test techniques

ISO/IEC 25010, (2011) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models

ISO/IEC 20246: (2017) Software and systems engineering — Work product reviews UML 2.5, Unified Modeling Language Reference Manual, http://www.omg.org/spec/UML/2.5.1/, 2017

## *ISTQB® dokumen*

ISTQB® Glossary ISTQB® Tingkat Dasar Overview 2018

ISTQB-CTFL-MBT Tingkat Dasar Model-Based Tester Extension Syllabus ISTQB-CTFL-AT

Tingkat Dasar Agile Tester Extension Syllabus

ISTQB-CTAL-TA Advanced Tingkat Test Analyst Syllabus

ISTQB-CTAL-TTA Advanced Tingkat Technical Test Analyst Syllabus

ISTQB-CTAL-TM Advanced Tingkat Test Manager Syllabus

ISTQB-CTAL-SEC Advanced Tingkat Security Tester Syllabus

ISTQB-CTAL-TAE Advanced Tingkat Test Automation Engineer Syllabus

ISTQB-CTEL-TM Expert Tingkat Test Management Syllabus

ISTQB-CTEL-ITP Expert Tingkat Improving the Test Process Syllabus

## *Buku dan Artikel*

Beizer, B. (1990) Software Testing Techniques (2e), Van Nostrand Reinhold: Boston MA Black, R. (2017) Agile Testing Foundations, BCS Learning & Development Ltd: Swindon UK Black, R. (2009) Managing the Testing Process (3e), John Wiley & Sons: New York NY Buwalda, H. et al. (2001) Integrated Test Design and Automation, Addison Wesley: Reading MA Copeland, L. (2004) A Practitioner's Guide to Software Test Design, Artech House: Norwood MA Craig, R. and Jaskiel, S. (2002) Systematic Software Testing, Artech House: Norwood MA Crispin, L. and Gregory, J. (2008) Agile Testing, Pearson Education: Boston MA Fewster, M. and Graham, D. (1999) Software Test Automation, Addison Wesley: Harlow UK Gilb, T. and Graham, D. (1993) Software Inspection, Addison Wesley: Reading MA Graham, D. and Fewster, M. (2012) Experiences of Test Automation, Pearson Education: Boston MA Gregory, J. and Crispin, L. (2015) More Agile Testing, Pearson Education: Boston MA Jorgensen, P. (2014) Software Testing, A Craftsman's Approach (4e), CRC Press: Boca Raton FL

Kaner, C., Bach, J. and Pettichord, B. (2002) Lessons Learned in Software Testing, John Wiley & Sons: New York NY

Kaner, C., Padmanabhan, S. and Hoffman, D. (2013) The Domain Testing Workbook, Context-Driven Press: New York NY

Kramer, A., Legeard, B. (2016) Model-Based Testing Essentials: Guide to the ISTQB® Certified ModelBased Tester: Tingkat Dasar, John Wiley & Sons: New York NY

Myers, G. (2011) The Art of Software Testing, (3e), John Wiley & Sons: New York NY

Sauer, C. (2000) "The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research," IEEE Transactions on Software Engineering, Volume 26, Issue 1, pp 1-

Shull, F., Rus, I., Basili, V. July 2000. "How Perspective-Based Reading can Improve Requirement Inspections." IEEE Computer, Volume 33, Issue 7, pp 73-79

van Veenendaal, E. (ed.) (2004) The Testing Practitioner (Chapters 8 - 10), UTN Publishers: The Netherlands

Wiegers, K. (2002) Peer Reviews in Software, Pearson Education: Boston MA Weinberg, G. (2008) Perfect Software and Other Illusions about Testing, Dorset House: New York NY

## *Sumber Daya Lainnya (tidak dirujuk secara langsung dalam Silabus ini)*

Black, R., van Veenendaal, E. and Graham, D. (2019) Foundations of Software Testing: ISTQB® Certification (4e), Cengage Learning: London UK

Hetzel, W. (1993) Complete Guide to Software Testing (2e), QED Information Sciences: Wellesley MA

# 8 Lampiran A – Latar Belakang Silabus

## *Sejarah Dokumen ini*

Dokumen ini adalah ISTQB® Silabus Penguji Bersertifikat Tingkat Dasar, kualifikasi internasional tingkat pertama yang disetujui oleh ISTQB® (www.istqb.org).

Dokumen ini disiapkan selama bulan Juni hingga Agustus 2019 oleh Kelompok Kerja yang terdiri dari anggota yang ditunjuk oleh Dewan Kualifikasi Pengujian Perangkat Lunak Internasional (ISTQB®). Pembaruan ditambahkan setelah meninjau masukan dari pengurus anggota yang telah menggunakan Silabus Foundation 2018.

Silabus Foundation 2018 sebelumnya disiapkan antara 2014 dan 2018 oleh Kelompok Kerja yang terdiri dari anggota yang ditunjuk oleh ***International Software Testing Qualifications Board*** (ISTQB®). Versi 2018 awalnya ditinjau oleh perwakilan dari semua dewan anggota ISTQB®, dan kemudian oleh perwakilan yang diambil dari komunitas pengujian perangkat lunak internasional.

## *Tujuan dari Kualifikasi Sertifikat Dasar*

- Untuk mendapatkan pengakuan atas pengujian sebagai spesialisasi rekayasa perangkat lunak yang esensial dan profesional
- Untuk menyediakan kerangka kerja standar untuk pengembangan karir penguji
- Untuk memungkinkan penguji yang berkualifikasi secara profesional agar diakui oleh pemberi kerja, pelanggan, dan rekan kerja, dan untuk meningkatkan profil penguji
- Untuk mempromosikan praktik pengujian yang konsisten dan baik dalam semua disiplin ilmu rekayasa perangkat lunak
- Untuk mengidentifikasi topik-topik pengujian yang relevan dan bernilai bagi industri
- Untuk memungkinkan pemasok perangkat lunak untuk mempekerjakan penguji bersertifikat dan dengan demikian mendapatkan keuntungan komersial atas pesaing mereka dengan mengiklankan kebijakan perekrutan penguji mereka
- Untuk memberikan kesempatan bagi penguji dan mereka yang berminat dalam pengujian untuk memperoleh kualifikasi yang diakui secara internasional dalam mata pelajaran tersebut

## *Tujuan dari Kualifikasi Internasional*

- Untuk dapat membandingkan pengetahuan pengujian di berbagai negara
- Untuk memungkinkan penguji bergerak melintasi batas negara dengan lebih mudah
- Agar proyek multinasional/internasional memiliki pemahaman yang sama tentang masalah pengujian
- Untuk meningkatkan jumlah penguji yang memenuhi syarat di seluruh dunia
- Untuk memiliki dampak/nilai yang lebih besar sebagai prakarsa berbasis internasional daripada dari pendekatan khusus negara mana pun
- Untuk mengembangkan pemahaman dan pengetahuan umum internasional tentang pengujian melalui silabus dan terminocatatani, dan untuk meningkatkan

tingkat pengetahuan tentang pengujian untuk semua peserta

- Untuk mempromosikan pengujian sebagai sebuah profesi di lebih banyak Negara
- Untuk memungkinkan penguji mendapatkan kualifikasi yang diakui dalam bahasa asli mereka
- Untuk memungkinkan berbagi pengetahuan dan sumber daya di seluruh negara
- Untuk memberikan pengakuan internasional terhadap penguji dan kualifikasi ini karena partisipasi dari banyak negara

## *Persyaratan Masuk untuk Kualifikasi ini*

Kriteria masuk untuk mengikuti ujian Tingkat Yayasan Penguji Bersertifikat ISTQB® adalah bahwa kandidat memiliki minat dalam pengujian perangkat lunak. Namun, sangat disarankan agar kandidat juga:

- Setidaknya memiliki latar belakang minimal dalam pengembangan perangkat lunak atau
  pengujian perangkat lunak, seperti: pengalaman enam bulan sebagai penguji penerimaan sistem atau pengguna atau sebagai pengembang perangkat lunak

- Ikuti kursus yang telah diakreditasi oleh salah satu dewan anggota yang diakui ISTQB dengan standar ISTQB®.

## *Latar Belakang dan Sejarah Sertifikat Foundation dalam Pengujian Perangkat Lunak*

Sertifikasi independen penguji perangkat lunak dimulai di Inggris dengan Badan Pemeriksaan Sistem Informasi (ISEB) dari British Computer Society, ketika Dewan Pengujian Perangkat Lunak didirikan pada tahun 1998 (www.bcs.org.uk/iseb). Pada tahun 2002, ASQF di Jerman mulai mendukung skema kualifikasi penguji Jerman (www.asqf.de). Silabus ini berdasarkan ISEB dan ASQF silabus; itu mencakup konten yang diatur ulang, diperbarui, dan tambahan, dan penekanannya diarahkan pada topik yang akan memberikan bantuan paling praktis bagi penguji.

Sertifikat Dasar yang ada dalam Pengujian Perangkat Lunak (misalnya, dari ISEB, ASQF atau dewan anggota yang diakui ISTQB) yang diberikan sebelum Sertifikat Internasional ini dirilis, akan dianggap setara dengan Sertifikat Internasional. Sertifikat Dasar tidak kedaluwarsa dan tidak perlu diperpanjang. Tanggal pemberiannya tertera pada Sertifikat.

Di setiap negara yang berpartisipasi, aspek lokal dikendalikan oleh Dewan Pengujian Perangkat Lunak yang diakui ISTQB nasional atau regional. Tugas dewan anggota ditentukan oleh ISTQB®, tetapi diterapkan di masing-masing negara. Tugas dewan anggota diharapkan mencakup akreditasi penyelenggara pelatihan dan pengaturan ujian

# 9 Lampiran B – Tujuan Pembelajaran/Tingkat Pengetahuan Kognitif

Tujuan pembelajaran berikut didefinisikan sebagai penerapan silabus ini. Setiap topik dalam silabus akan diperiksa sesuai dengan tujuan pembelajarannya.

## *Tingkat 1: Ingat (K1)*

Kandidat akan mengenali, mengingat, dan mengingat istilah atau konsep.
**Kata kunci:** Mengidentifikasi, mengingat, mengambil kembali, mengingat, mengenali, mengetahui

**Contohnya:**

Dapat mengenali definisi "kegagalan" sebagai:

- "Tidak terkirimnya layanan ke pengguna akhir atau pemangku kepentingan lainnya" atau
- "Penyimpangan komponen atau sistem dari penyampaian, layanan, atau hasil yang diharapkan"

## *Tingkat 2: Memahami (K2)*

Kandidat dapat memilih alasan atau penjelasan untuk pernyataan yang terkait dengan topik, dan dapat meringkas, membandingkan, mengklasifikasikan, mengkategorikan dan memberikan contoh untuk konsep pengujian.
**Kata kunci:** Meringkas, menggeneralisasi, abstrak, mengklasifikasikan, membandingkan, memetakan, kontras, mencontohkan, menafsirkan, menerjemahkan, mewakili, menyimpulkan, menyimpulkan, mengkategorikan, membangun model

**Contohnya:**

Dapat menjelaskan alasan mengapa analisis dan desain pengujian harus dilakukan sedini mungkin:

- Untuk menemukan cacat ketika lebih murah untuk dihilangkan
- Untuk menemukan cacat yang paling penting terlebih dahulu
  Dapat menjelaskan persamaan dan perbedaan integrasi dan pengujian sistem:
- Kesamaan: objek pengujian untuk pengujian integrasi dan pengujian sistem mencakup lebih dari satu komponen, dan pengujian integrasi dan pengujian sistem dapat mencakup jenis pengujian non-fungsional
- Perbedaan: pengujian integrasi berkonsentrasi pada antarmuka dan interaksi, dan pengujian sistem berkonsentrasi pada aspek keseluruhan sistem, seperti pemrosesan ujung ke ujung

## *Tingkat 3: Terapkan (K3)*

Kandidat dapat memilih penerapan konsep atau teknik yang benar dan menerapkannya pada konteks tertentu.
**Kata kunci:** Menerapkan, melaksanakan, menggunakan, mengikuti prosedur, menerapkan prosedur

**Contohnya:**

- Dapat mengidentifikasi nilai batas untuk partisi yang valid dan tidak valid
- Dapat memilih kasus uji dari diagram transisi status tertentu untuk mencakup semua transisi

**Referensi** (Untuk tingkat kognitif dari tujuan pembelajaran)

*Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon: Boston MA*

# 10 Lampiran C – Catatan Rilis

ISTQB® Foundation Silabus 2018 V3.1 adalah pembaruan kecil dari rilis 2018. Catatan Rilis 2018 V3.1 terpisah telah dibuat dengan gambaran umum per bab.
Selain itu, versi Foundation Silabus 2018 V3.1 dengan perubahan alur telah dirilis,

ISTQB® Foundation Silabus 2018 adalah pembaruan dan penulisan ulang utama dari rilis 2011. Karena alasan ini, tidak ada catatan rilis terperinci per bab dan bagian. Namun demikian, ringkasan perubahan utama disediakan di sini. Selain itu, dalam dokumen Catatan Rilis terpisah, ISTQB® menyediakan ketertelusuran antara tujuan pembelajaran di Silabus Tingkat Dasar versi 2011 dan tujuan pembelajaran di Silabus Tingkat Dasar versi 2018, menunjukkan mana yang telah ditambahkan, diperbarui, atau dihapus.

Pada awal tahun 2017 lebih dari 550.000 orang di lebih dari 100 negara telah mengikuti ujian dasar, dan lebih dari 500.000 merupakan penguji bersertifikat di seluruh dunia. Dengan harapan bahwa mereka semua telah membaca Silabus Dasar untuk dapat lulus ujian, hal ini membuat Silabus Dasar menjadi dokumen pengujian perangkat lunak yang paling banyak dibaca!

Pembaruan besar ini dibuat sehubungan dengan warisan ini dan untuk meningkatkan nilai yang diberikan ISTQB® kepada 500.000 orang berikutnya di komunitas pengujian global.

Dalam versi ini, semua tujuan pembelajaran telah diedit untuk membuatnya atomik, dan untuk membuat ketertelusuran yang jelas dari setiap tujuan pembelajaran ke bagian konten (dan pertanyaan ujian) yang terkait dengan tujuan pembelajaran tersebut, dan untuk memiliki ketertelusuran yang jelas dari bagian konten (dan pertanyaan ujian) kembali ke tujuan pembelajaran terkait. Selain itu, alokasi waktu bab telah dibuat lebih realistis dibandingkan dengan silabus versi 2011, dengan menggunakan heuristik yang telah terbukti dan rumus yang digunakan dengan silabus ISTQB® lainnya, yang didasarkan pada analisis tujuan pembelajaran yang akan dicakup dalam setiap bab.

Meskipun ini adalah silabus Dasar, yang mengungkapkan praktik dan teknik terbaik yang telah bertahan dalam ujian waktu, kami telah membuat perubahan untuk memodernisasi penyajian materi, terutama dalam hal metode pengembangan perangkat lunak.
(misalnya, Scrum dan penyebaran berkelanjutan) dan teknologi (misalnya, Internet of Things).
Kami telah memperbarui standar yang direferensikan agar lebih baru sebagai berikut:

1. ISO/IEC/IEEE 29119 menggantikan IEEE Standar 829.
2. ISO/IEC 25010 menggantikan ISO 9126.
3. ISO/IEC 20246 menggantikan IEEE 1028.

Selain itu, karena portofolio ISTQB® telah berkembang secara dramatis selama dekade terakhir, kami telah menambahkan referensi silang yang luas ke materi terkait di silabus ISTQB® lainnya, jika relevan, serta meninjau dengan cermat untuk keselarasan dengan semua silabus dan dengan ISTQB® Glosarium. Tujuannya adalah untuk membuat versi ini lebih mudah dibaca, dipahami, dipelajari, dan diterjemahkan, dengan fokus pada peningkatan kegunaan praktis dan keseimbangan antara pengetahuan dan keterampilan.

Untuk analisis terperinci tentang perubahan yang dibuat dalam rilis Silabus Dasar 2018, lihat Catatan Rilis Tingkat Yayasan Penguji Bersertifikat ISTQB® 2018.

# 11 Daftar Istilah