Bab 9
Pengujian Tahap Uji

Pengujian efektif pada tahap-tahap awal memberikan sebuah jaminan tinggi bahwa program-program *executable* akan berfungsi sesuai ketetapan. Bagaimanapun juga, hal itu tidak sampai mereka (program-program *executable)* dieksekusi sebagai sebuah sistem agar jaminan lengkap bahwa mereka akan berfungsi sebagaimana mestinya terpenuhi.

Semakin banyak pengujian dibentuk selama tahap kebutuhan, perancangan, dan penulisan program, semakin sedikit pengujian yang mungkin harus dibentuk selama tahap-tahap awal pengujian secara luas.

Bab ini menyuguhkan sebuah proses uji terekomendasi untuk setiap tahap uji yang teridentifikasi.

Hasil Tahap Uji (Test Phase Deliverables)

Tahap uji dapat menjadi proses yang memakan waktu dan beaya. Proses dari pendefinisian criteria penerimaan selama kebutuhan dan pengembangan situasi tes dalam kaitannya dengan perancangan dan pemrograman aplikasi dapat mengurangi biaya tersebut.
Dalam kondisi normal, ketika pengujian terencana dengan baik dan terstruktur, waktu dan usaha yang dibutuhkan untuk menguji sebuah aplikasi akan berkurang.

Pengujian sebuah sistem aplikasi selama tahap uji memiliki beberapa hasil baru. Hasil-hasil yang didapatkan selama tahap uji meliputi:
- Tahap uji rencana-uji.
- Uji data.
- Hasil dari uji sebelumnya.
- Reaksi uji *third-party*.
- Dokumentasi hasil uji formal.

Tentang Tahap Uji

Terdapat banyak metoda uji sebuah sistem aplikasi. Tim uji memperhatikan semua bentuk uji sehingga organisasi memiliki probabilitas sukses yang tinggi ketika memasang sebuah sistem aplikasi baru.
Tahap uji tersebut:

- **Pengujian Manual, Regresi dan Fungsional (Reliabilitas)**
  Pengujian manual menjamin orang yang berinteraksi dengan sistem terotomatisasi dapat melakukan fungsinya secara tepat. Pengujian regresi mempakan pembuktian (verifikasi) bahwa apa yang sedang dipasang tidak akan mempengaruhi tiap bagian aplikasi yang telah terpasang atau aplikasi lain yang dijembatani oleh aplikasi baru. Pengujian fungsional membuktikan bah\va kebutuhan sistem dapat dilakukan dengan

tepat ketika menjalani sebuah kondisi yang bef\'3riasi dan perulangan dari tr3llsaksi-transaksi.

- **Pengujian Pemenuhan (Otorisasi)**
  Pengujian harus membuktikan bahwa aturan otorisasi telah diterapkan sebagaimana mestinya. Pengujian akan mengevaluasi pemenuhan dengan aturan otorisasi. Kondisi-kondisi uji harus mencakup transaksi-transaksi tak resmi atau proses-proses untuk memastikan bahwa mereka (transaksi itu) ditolak sebaik transaksi resmi yang sedang diterima.

- **Pengujian Fungsional (Integritas File)**
  Pengawasan tentang integritas ftIe komputer seharusnya dibuktikan (verifikasi) selama pengujian. Dan lagi, peremajaan file secukupnya ltarus dilakukan sedemikian hingga pengawasan integritas dapat diuji selama beberapa kali itefasi pada eksekus~ sistem aplikasi.

- **Pengujian Fungsional *(audit trail)***
  Fungsi *audit trail* seharusnya diuji. Pengujian seharusnya menjamin bahwa transaksi sumber dapat dilacak untuk pengawasan total, dimnana transaksi yang sedang mendukung pengawasan total dapat diidentifikasi, dan dimana pengolahan transaksi tunggal atau keseluruhan sistem dapat dibangun-ulang menggunakan informasi *audit trail.*

- **Pengujian Perbaikn (Kemalaran Proses/ Kontinuitas Proses)**
  Jika pengolahan ham berlanjut selama.periode tertentu dimana sistem terotomatisasi tidak beroperasi, maka prosedur-prosedur pengganti pengolahan harus diuji selama tahap uji. Penguna sistem aplikasi harus dilibatkan..mlam uji perbaikan yang lengkap sedemikian hingga tidak hanya sistem terotomatisasi saja yang diuji tetapi prosedur-prosedur untuk membentuk aspek manual dari perbaikan juga harus diuji.

- **Pengujian Penekanan (Tingkat Pelayanan)**
  Pengujian penekanan harus mencoba memukan tingkatan-tingkatan pengolahan dimana sistem tidak dapat lagi berfungsi secara efektif. Di dalam sistem *on-line*, testing ini mungkin ditentukan oleh volume transaksi, sementara di dalam pengujian sistem *batch* ukuran *batch* atau besar volume dari transaksi tipe tertentu dapat menguji tabel-tabel intemal, dan kemampuan mengurutkan. .

- **Pengujian Pemenuhan (Keamanan)**
  Kecukupan akan prosedur-prosedur keamanan seharusnya diuji oleh percobaan untuk melanggar prosedur-prosedur tersebut. Sebagai contoh, pengujian harus mencoba untuk mengakses atau mengubah data yang diilakukan oleh individu yang tidak sah.

- **Pengujian Mengikuti Metodologi**
  Pengujian seharusnya dibentuk sesuai dengan kebijakan pengujian, prosedur-prosedur, standar-standar, dan garis pedoman dan organisasi. Metodologi harus menetapkan tipe dari perencanaan. Uji yang dibutuhkan, teknik dan *tools* pengujian

terekomendasi, sebaik dokumentasi yang dibutuhkan selama tahap uji. Metodologi juga harus menetapkan metoda untuk menentukan apakah uji berhasil atau tidak.

- **Pengujian Fungsional (Kebenaran)**
  Pengujian kebenaran fungsional membuktikan bahwa persyaratan- persyaratan yang dispesifikasikan oleh pengguna berfungsi sesuai dengan persyaratan tersebut. Pengujian tahap uji mungkin berharap menekankan pada perhatian terhadap uji lain, atau menekankan pada transaksi yang tidak layak dimasukkan untuk menguji keabsahan data dan fungsi pendeteksian kesalahan.

- **Pengujian Dukungan Manual (Mudah Digunakan)**
  Pengujian terhadap kemudahan penggunaan sistem adalah aspek penting dari tahap uji, karena hal ini sulit dievaluasi sebelumnya. Adalah penting bahwa aspek orang dari sistem dievaluasi dalam lingkungan serealistis mungkin.

- **Pemeriksaan *(Maintainability)***
  Sistem yang lengkap seharusnya diperiksa oleh sebuah kelompok bebas, lebih disukai spesialis perawatan sistem, dengan tujuan untuk mengevaluasi *maintainability* dari sistem aplikasi. Standar-standar pengembangan. sistem seharusnya memikirkan juga *maintainabilitynya.*

- **Pengujian Bencana *(Portability)***
  Hanyalah metoda mengeksekusi aplikasi dalam lingkungan yang berbeda. menjamin aplikasi akan berfungsi dalam sistem operasi kedua. Pengujian bencana merupakan sebuah mekanisme yang mensimulasikan persoalan - persoalan dalam lingkungan asli sedemikjan hingga lingkungan pengolahan lainnya dapat diuji. Pada saat hal itu tidak mungkin mensimulasikan semua lingkungan yang memungkinkan sistem aplikasi berpindah pengetahuan bahwa sistem aplikasi tersebut dapat berpindah diantara dua lingkungan berbeda memberikan probabilitas tinggi bahwa perpindahan lain tidak akan menyebabkan komplikasi yang besar.

- **Pengujian Fungsional dan Regresi *(Coupling)***
  Pengujian fungsional membuktikan bahwa setiap fungsi baru saling berhubungan dengan layak Pengujian regresi membuktikan bahwa segmen-segmen tetap dari sistem aplikasi yang saling berhubungan dengan aplikasi lain masih berfungsi dengan layak.

- **Peogujian Pemenuhan (Unjuk-kerja)**
  Kriteria performansi dibangun selama tahap kebutuhan. Kriteria itu harus diremajakan jika kebutuhan berubah selama tahap akhir siklus kehidupan. Banyak kriteria dapat dievaluasi selama tahap uji, dan kriteria yang dapat diuji seharusnya diuji. Akan tetapi, mungkin masih perlu menunggu sampai sistem ditempatkan kedalam produksi untuk membuktikan bahwa semua kriteria telah diraih.

- **Pengujian Operasi (Mudah Dioperasikan)**
  Selama tahap uji, pengujian seharusnya dipimpin oleh staf operasi biasa. Proyek peogembangan personil tidak seharusnya diijinkan memandu atau membantu selama proses uji berlangsung. Hanyalah melalui operasi biasa, personil memimpin pengujian bahwa kelengkapan instruksi operator dan kemudahan pengoperasian sistem dapat dievaluasi dengan layak.

Tanggung-jawab Tahap Uji

Pada akhir tahap uji, sistem aplikasi akan ditempatkan kedalam produksi dengan tujuan pemenuhan keinginan pengguna. Tahap uji memberikan kesempatan akhir bagi pengguna untuk memastikan bahwa fungsi-fungsi sistem secara tepat memenuhi pertanggunganjawab pengguna. Oleh karena alasan ini, pengguna seharusnya menjadi orang pokok dalam pengujian sistem aplikasi.

Bagian pengolahan data memiliki kesempatan mengevaluasi sistem aplikasi selama tahap program. Selama tahap ini, mereka menentukan apakah sistem berfungsi dengan layak atau tidak sesuai dengan pemahaman pengolahan data dari persyaratan-persyaratan. Tahap uji terbaik dibentuk oleh sebuah kelompok daripada tim proyek. Hal ini tidak berarti bahwa tim proyek tidak dapat dilibatkan atau membantu, tetapi mereka tidak dapat menjadi orang dominan dalam tahap uji.

Jikalau individu yang sama bertanggungjawab untuk pengujian tahap program dan pengujian tahap uji. maka tidaklah perlu memiliki dua tahap berbeda. Jikalau pengolahan data mengandaikan responsibilitas uji selama tahap program, dan pengguna menerimanya selama tahap uji. maka kedua tahap itu menjadi komplementer satu sama lain.

Departemen pengguna seharusnya diberi tanggung jawab menguji sistem untuk menentukan apakah sistem berjalan sesuai kehendaknya. Tentang permasalahan komunikasi, mungkin ada perbedaan - perbedaan antara spesifikasi sistem yang telah dibangun dan kebutuhan yang diharapkan oleh peoguna untuk dipenuhi. Idealnya, pengguna akan mengembangkan kondisi uji dari tahap kebutuhan, dan selama tahap uji harus membongkar cacat yang tersisa dalam sistem aplikasi.

Tools Tahap Uji Terekomendasi

Tahap uji merupakan lini pertahanan terakhir terhadap cacat yang memasuki lingkungan operasi. Jika tiada pengujian telah dilakukan sebelum tahap uji, maka tidaklah masuk akal untuk mengharapkan pengujian pada tahap ini akan mengeluarkan semua cacat. Pengalaman menunjukkan bahwa sukar bagi tahap uji akan efektif lebih dari 80% dalam mengurangi cacat. Tak pelak lagi, semakin sedikit cacat memasuki tahap uji, semakin sedikit cacat akan memasuki lingkungan produksi.

Dua buah *tools* pengujian yang telah direkomendasi untuk pengujian selama tahap uji adalah *too/s* uji data-uji dan pengujian volume.
Data-uji dapat dipakai menciptakan sebanyak mungkin kondisi uji yang perlu dipertimbangkan untuk mengevaluasi aplikasi.

Pengujian volume mengembangkan cukup tipe-tipe spesifik transaksi uji, sedemikian hingga batas sistem aplikasi intemal dievaluasi.

Tools-Uji Data Uji
Konsep dari data uji adalah sesuatu yang menciptakan kondisi pengolahan yang representatif memakai uji transaksi. Bagian kompleks dari data uji adalah menentukan transaksi mana yang digunakan sebagai uji transaksi. Untuk itu, mengoptimalkan pengujian melalui pemilihan transaksi-transaksi uji yang paling penting mempakan aspek kunci dari *tools* ini.

Beberapa *tools-uji* merupakan metoda terstruktur untuk merancang data-uji. Sayangnya, semua *tools* ini, meski sangat efektif, membutuhkan banyak waktu dan usaha yang diwujudkan. Sedikit organisasi mengalokasikan dana yang cukup untuk pengujian tipe ini. Sebagai tambahan, personil pengolahan data tidak dilatih menggunakan *tools* ini.

Desain *Test Deek*
Untuk merancang sebuah *test deek* yang memadai, penguji harus akrab dengan: standar uji EDP; kebijakan relevan lainnya; meliputi ketetapan mereka dalam transaksi yang disimulasikan; prosedur pengolahan data; dan format masukan dan keluaran untuk seluruh tipe transaksi yang diolah.
Sangat membantu pula, pengetahuan dasar tentang tujuan sistem dan prosedur operasi yang diperoleh melalui proses peninjauan dan menganalisa sistem diagram alir, instruksi operasi, dan dokumentasi.

Agar efektif, sebuah *test deek* harus memakai transaksi yang memiliki data masukan yang sah dan tidak sah pada sebuah skala yang lebar - data sah untuk pengujian operasi pengolahan normal, dan data tidak sah untuk pengujian kendali-terprogram.

Hanya sebuah transaksi uji seharusnya diolah pada setiap rekaman induk. Hal ini mengijinkan sebuah evolusi terpisah dari kendali program spesifik dengan jaminan bahwa hasil uji tidak akan terpengamh oleh proses transaksi uji lainnya pada rekaman induk yang sama. .
Tipe umum kondisi yang harus diuji didiskusikan berikut:

- **Uji Kejadian Normal Transaksi**
  Untuk menguji kemampuan sistem komputer untuk mengolah secara akurat data-sah, sebuah *test deck* harus mencakup transaksi yang terjadi socara normal. Contoh, pada sistem penggajian, transaksi yang terjadi secara normal meliputi : perhitungan pembayaran teratur: pembayaran kelebihan waktu; dan beberapa tipe pembayaran premi, sebaik menset rekaman induk untuk pekerja-pekerja upahan baru dan peremajaan rekaman induk yang telah ada untuk pekerja-pekerja lain.

- **Uji Memakai Data Tidak-Sah**
  Pengujian adanya atau keefektifan kendali terprogram memerlukan penggunaan data tidak sah. Contoh uji yang menyebabkan data tidak sah ditolak adalah :
  1. Pemasukan karakter huruf ketika karakter numerik diharapkan dan sebaliknya.
  2. Pemakaian catatan tidak sah atau bilangan identifikasi.

3. Pemakaian data tak lengkap atau data yang tak berhubungan dalam sebuah field data khusus atau mengabaikan seluruhnya.
4. Pemasukan jumlah negatif ketika hanya jumlah positif yang valid, dan sebaliknya.
5. Pemasukan kondisi tak logis dalam field data yang seharusnya berrelasi secara logis.
6. Pemasukan sebuah kode transaksi atau jumlah yang tidak sesuai dengan kode atau jumlah yang dibuat oleh prosedur operasi atau label pengontrol. Misalkan. kode sah untuk status kerja dalam sistem penggajian adalah A, B, dan C. maka kode yang dimasukkan adalah sesuatu selain A, B, dan C.
7. Pemasukan transaksi atau kondisi yang akan merusak batasan yang telah ditentukan oleh aturan atau prosedur operasi standar. Contoh. dalam sistem penggajian. masukan dari X + 2 dollar sebagai pembayaran bruto pekerja ketika X dollar merupakan maksimum pembayaran bruto yang diijinkan oleh aturan untuk tingkatan tertinggi.

- **Uji Melanggar *Established Edit Checks***

  Untuk sistem dokumentasi, pemeriksa harus mampu menentukan rutine edit mana yang tercakup dalam program komputer yang diuji. Ia kemudian harus membangun transaksi-transaksi uji untuk melanggar naskah tersebut dengan maksud untuk melihat apakah transaksi uji tersebut ada secara nyata.

  Pemasukan Data Uji

  Setelah tipe transaksi-transaksi uji ditentukan, data uji harus dimasukkan dalam formulir entry yang benar. Jika tim uji menghendaki pengawasan uji terhadap masukan dan pengolahan komputer, mereka harus memberikan data kedalam sistem pada dokumen-dokumen sumber organisasi untuk dikonversi kedalam bentuk yang dapat dibaca oleh mesin. Jika hanya pengujian pengawasan terhadap pengolahan komputer, data harus dimasukkan dalam bentuk yang dapat dibaca oleh mesin.

  Menganalisa Hasil Pengolahan

  Sebelum pengolahan data uji melalui komputer dilakukan, tim uji harus menentukan sebelumnya hasil yang benar untuk setiap transaksi uji untuk pembandingan dengan hasil yang nyata. Setiap perbedaan antara hasil nyata dengan hasil yang ditentukan sebelumnya menunjukkan kelemahan sistem. Tim uji harus menentukan pengaruh kelemahan pada keakuratan dari file induk dan pada reliabilitas laporan dan produk komputer yang lain

  Meogaplikasikan *Test Deek* Pada Program Yang Meremajakan Rekaman Ioduk

  Terdapat 2 pendekatan dasar untuk menguji program-program yang meremajakan rekaman induk. Pendekatan pe;rtama adalah menyalin rekaman induk yang aktual dan atau rekaman induk yang disimulasikan yang biasa dipakai menset sebuah file induk terpisah untuk keperluan uji. Pendekatan lainnya adalah, rekaman audit khusus, menahan dalam file induk organisasi yang sedang *current.*

  Jika menggunakan pendekatan pertama, tim uji harus memiliki sebagian file induk organisasi yang tersalin untuk membuat sebuah file induk uji. Dari sebuah *print-out*

file, tim tersebut menyeleksi rekaman-rekaman yang layak uji. Kemudian penguji meremajakan file uji dengan data sah dan data tidak-sah melalui program-program organisasi yang mengolah transaksi- transaksi yang sedang menyusun *test deck.*

Rekaman induk dapat disimulasikan paling mudah melalui penyiapan dokumen asal dan mengolahnya dengan program yang dipakai organisasi untuk menambah rekaman baru kepada file induknya. Prosedur *test deck* mensimulasikan rekaman-rekaman adalah sama dengan prosedur menyalin rekaman-rekaman.

Sebuah keuntungan memakai rekaman yang disimulasikan adalah bahwa mereka dapat disesuaikan dengan kondisi praktis dan mereka mengurangi kebutuhan melokalisir dan menyalin rekaman-rekaman organisasi yang cocok.

Pendekatan yang paling praktis adalah memakai sebuah file induk uji yang merupakan kombinasi dari rekaman induk yang disalin dan disimulasikan.

Melalui penggunaan rekaman induk yang disalin dalam sebuah file terpisah, penguji menghindari komplikasi dan bahaya *running* data. uji pada pengolahan yang teratur menentang file induk yang sedang *current.*
Kerugian dari rekaman tersalin dan tersimulasikan adalah bahwa program-program komputer harus dimuat dan peralatan disetel dan dioperasikan hanya untuk tujuan *audit,*sehingga akan melibatkan tambahan biaya.

Proses *Test Deck*
Proses yang direkomendasi untuk penciptaan dan penggunaan data uji merupakan proses dengan 9 langkah berikut :

- **Langkah 1- Mengidentifikasi Sumber Pengujian**
  Penciptaan dan pengeksekusian pengujian memakai data uji dapat diinginkan sebagai proses yang diperluas atau terbatas. Sayangnya, banyak programer mendekati penciptaan data uji dari segi ‘kami mungkin akan mengerjakan pekerjaan terbaik’, dan kemudian memulai mengembangkan transaksi uji. Ketika habis waktu, pengujian baru lengkap. Pendekatan yang direkomendasi menganjurkan agar jumlah sumber daya yang dialokasikan pada data uji *test too/* ditentukan dan kemudian sebuah proses dikembangkan untuk mengoptimasi waktu.

- **Langkah 2- Mengidentifikasi Kondisi Pengujian**
  Testing matriks yang diuraikan pada bab 3 dari buku ini direkomendasikan sebagai dasar penentuan kondisi uji (lihat gambar 20). Jika konsep matriks itu tidak digunakan, maka kondisi uji yang mungkin harus ditentukan selama pemakaian *test too/* ini.

- **Langkah 3- Mengatur Kondisi Pengujian**
  Jika sumber daya terbatas, pemakaian maksimum terhadap sumber daya tersebut akan diperoleh melalui pengujian terhadap kondisi uji terpenting. Tujuan melakukan rangking adalah mengidentifikasi kondisi mana yang bams diuji pertama kali

sehingga data uji *test too/* dapat berkonsentrasi pada kondisi uji dengan prioritas tinggi.

Penetapan rangking bukan berarti bahwa kondisi uji dengan rangking rendah tidak diuji rangking dapat dipakai untuk 2 tujuan: pertama, untuk menentukan kondisi mana yang harus diuji pertama kali; dan kedua, untuk menentukan jumlah sumber daya yang dialokasikan untuk setiap kondisi uji.

Contoh, jika pengujian pemotongan FICA relatif rendah kondisinya, hanya sebuah transaksi uji mungkin dibentuk untuk menguji kondisi tersebut, sementara untuk. kondisi uji rangking yang lebih tinggi beberapa transaksi uji mungkin dibentuk dengan tujuan menguji kondisi itu.

- **Langkah 4- Memilih Kondisi Pengujian**

  Berdasarkan rangking, kondisi-kondisi yang diuji harus diseleksi. Contoh, pengujian FICA merupakan sebuah kondisi yang masuk akal diidentifikasi dan dirangking, namun pembentukan kondisi uji yang spesifik sangatlah umum. Tiga situasi uji yang mungkin diidentifikasi seperti: pekerja-pekerja yang akan menerima gaji melampaui pemotongan FICA maksimum; ~kerja yang menerima gaji periode itu akan melampaui perbedaan antara penghasilan tahunan pada saat itu dan pemotongan maksimum; dan pekerja yang menerima gaji tahunan lebih dari jumlah pembayaran satu periode di atas pemotongan FICA maksimum.

  Setiap situasi uji harus didokumentasi dalam sebuah matriks uji.

- **Langkah 5- Menentukan Hasil Pengolahan Yang Benar**

  Hasil pengolahan yang benar untuk setiap situasi uji harus ditetapkan. Situasi uji harus diidentifikasi oleh bilangan unik. kemudian sebuah buku catatan dibuat untuk hasil yang benar dari setiap kondisi uji. Jika sistem menyediakan pemeriksaan otomatis terhadap setiap situasi uji, maka formulir khusus mungkin diperlukan sebagai informasi yang akan dikonversikan melalui media mesin-baca.

  Waktu yang tepat untuk menentukan hasil pengolahan yang tepat adalah sebelum transaksi uji selesai dibentuk. Langkah ini membantu menentukan kelayakan dan kegunaan transaksi uji. Proses dapat juga menunjukan jika terdapat jalan memperluas keefektifan transaksi uji, dan apakah kondisi yang sama telah diuji oleh transaksi lain.

- **Langkah 6- Menciptakan Transaksi Pengujian**

  Setiap situasi uji perlu dikonversi kedalam format yang layak untuk diuji. Metoda yang paling umum untuk menciptakan transaksi pengujian mencakup:
  - Tombol entry melalui kartu, pita-magnetis , disk, atau terminal.
  - Pembangkit data uji.
  - Penyiapan sebuah formulir masukan yang akan diberikan kepada personil pengguna untuk diisi.

- **Langkah 7- Mendokumentasi Situasi Pengujian**

  Situasi uji dan hasil pengujian, keduanya harus didokumentasikan. Lihat bab 12 untuk dokumentasi tes yang direkomendasi.

- **Langkah 8- Memimpin Pengujian**
  Sistem yang *executable* seharusnya berjalan dengan menggunakan kondisi uji. Bergantung pada luasnya uji, sistem itu dapat berjalan di bawah sebuah kondisi uji, atau berjalan dalam sebuah lingkungan produksi yang terstimulasi.

- **Langkah 9- Memverifikasi dan Mengkoreksi**
  Hasil pengujian harus diverifikasi dan setiap kebutuhan mengkoreksi program dibentuk Persoalan dideteksi sebagai hasil pengujian dapat diberi atribut tidak hanya untuk cacat sistem, tetapi juga menguji cacat data uji.

Membangun Contoh Sebuah *Test Deck*- Aplikasi Penggajian

Pertama kali, semua dokumentasi yang tersedia ditinjau terhadap bagian-bagian manual dan otomatis dari setiap sistem. Untuk memahami operasi manual, mereka (kantor akuntan publik U.S.) menginterview supervisor penggajian dan para juru tulis, mereview aturan dan kebiasan- kebiasaan berkaitan dengan pembayaran dan cuti, dan mengenallebih dekat tentang prosedur operasi penggajian standard. Untuk bagian otomatis dari setiap sistem, mereka meng*interview* perancang sistem, meng*interview* sistem, dokumentasi program, dan prosedur operasi.

Setelah memperoleh banyak pengetahuan kerja lerhadap setiap sistem, mereka memutuskan menguji program-program komputer yang dipakai untuk meremajakan rekaman induk penggajian dan digunakan meraghitung pembayaran dwi-mingguan serta cuti.
Meski mereka memperhatikan benar terhadap keterangan program, mereka memutuskan bahwa program-program lain yang dipakai dalam siklus pengolahan gaji dwi-mingguan secara normal harus juga diuji untuk melihat bagaimana mereka akan menangani data uji.

Kemudian mereka membangun *test deek* dari transaksi-transaksi penggajian dan cuti yang disimulasikan untuk menguji keefektifan pengendali internal, pemenuhan akan aturan dan kebiasaan yang dapat diaplikasikan, dan kecukupan dari prosedur operasi penggajian yang standar. *Test deek* meliputi transaksi-transaksi yang dihasilkan oleh data sah dan data tidak sah. Transaksi-transaksi itu didasarkan pada prosedur khusus dan kebiasaan-kebiasaan, dan dirancang unluk mencek efektifitas dari kontrol intemal pada setiap instalasi pengolahan gaji. Mereka memakai satu transaksi untuk setiap rekaman induk terpilih.

Metoda terbaik untuk mendapatkan rekaman-rekaman induk penggajian yang layak uji adalah menggunakan salinan-salinan rekaman induk yang aktual melengkapi simulasi rekaman-rekaman yang disesuaikan jika kondisi uji tidak ditemukan pada rekaman salinan.

Dengan demikian. mereka mendapatkan sebuah duplikat dari setiap file induk penggajian dari agen-agen dan memiliki sebuah bagian yang dicetak dalam salinan yang dapat dibaca.
Dari cetakan ini, mereka menyeleksi rekaman induk khusus yang cocok dengan setiap transaksi uji. Ketika tiada rekaman salinan muncul pada cetakan yang cocok dengan

sebagian transaksi khusus, mereka menyUSUD sebuah rekaman induk yang disimuIasikaIl dengan cara menyiapkan dokumen-dokumen asli dan mengolahnya bersama program yang dipakai oleh setiap instalasi unluk menambahkan rekaman dari pekerja baru kepada file induknya.

Kemudian mereka menambahkan rekaman-rekaman tersimulasi kepada rekaman-rekaman tersalin untuk membangun file induk uji.

Mereka selanjutnya menyiapkan kertas kerja untuk setiap transaksi uji, bilangan pengendali yang diberikan kepada transaksi, tipe dokumen masukan yang dipakai, sifat dan tujuan uji.
Mereka menentukan sebelumnya hasil akhir yang benar untuk seluruh transaksi uji dan merekam hasil tersebut dalam kertas kerja sebagai perbandingan dengan hasil yang aktual.

Dengan bantuan personil kantor penggajian, mereka kemudian mengkodekan transaksi uji di atas dokumen asal. Data kemudian dimasukkan dan diverifikasi. Mereka kemudian mengolah data uji, yaitu program-program penggajian agen yang aktual disesuaikan dengan hasil yang ditentukan sebelumnya, untuk melihat apakah terdapat perbedaan-perbedaan.

Mereka menemukan sistem yang menerima dan mengolah beberapa transaksi uji tidak-sah yang telah ditolak atau ditandai oleh pengendali komputer terprogram. Altematif pengendalian manual kurang efektif penuh atau malahan tidak efektif, karena mereka dapat menghindari atau kompromi dengan kecurangan, pengabaian, atau kesalahan karena kurang hati-hati. Mereka merekomendasi bahwa pengendali otomatis sistem memperkuat jaminan keakuratan penggajian dan melindungi pemerintah dari pembayaran semu.

Salinan kertas kerja skema kondisi uji digambarkan dalam gambar 40. (lihat buku p.245).

Tool Uji Uji-Volume
Pengujian volume merupakan *tool* tambahan bagi data uji. Tujuannya adalah membuktikan bahwa sistem dapat bekerja dengan layak ketika program intemal atau pembatas sistem telah dilampaui.
Untuk ini mungkin memerlukan sejumlah besar volume transaksi yang dimasukkan selama pengujian.

Tipe-tipe pembatas intemal yang dapat dievaluasi dengan pengujian volume meliputi:
- Akumulasi intemal informasi, seperti tabel.
- Jumlah baris item pada sebuah kejadian, seperti item yang tereakup beserta urutannya.
- Ukuran field akumulasi.
- Pembatasan tanggal, seperti tahun kabisat, perubahan dekade, perpindahan tahun kalender.

- Pembatasan ukuran field, seperti jumlah karakter yang dialokasikan untuk nama orang.
- Jumlah entitas akunting. seperti jumlah lokasi bisnis, kota/negara dimana bisnis dibentuk.

Langkah-langkah yang terekomendasi untuk menetapkan pembatas program / sistem sebagai berikut:

- **Langkah 1 - Mengidentifikasi Data Masukan yang Dipakai oleh Program**
  Sebuah metoda untuk mengidentifikasi pembatasan yang lebih disukai adalah mengevaluasi data. Setiap elemen data ditinjau untuk menentukan apakah ia memiliki sebuah pembatasan system. Ini adalah metoda termudah daripada mencoba mengevaluasi program. Metoda ini juga sangat membantu dalam mengidentifikasi pembatasan sistem yang tidak setuju akan pembatasan program. Ia memiliki keuntungan bahwa data hanya mungkin diperlukan untuk dievaluasi sekali, sebagai oposisi terhadap evaluasi numerik program-program individual.

  Seluruh data yang memasuki sebuah sistem aplikasi harus diidentifikasi. Elemen data itu tidak dipakai oleh aplikasi, melainkan hanya kemampuan mengaksesnya, harus dihapus, menghasilkan daftar data masukan yang dipakai oleh sistem aplikasi.

- **Langkah 2 - Mengidentifikasi Data yang Dibuat oleb Program.**
  Data yang dibangkitkan oleh sistem aplikasi harus diidentifikasi. Hal itu tidak hanya berupa elemen-elemen data yang tidak dimasukkan kedalam sistem, namun juga termasuk rekaman data intemal atau rekaman data keluaran. Mengetahui data masukan dan data keluaran adalah proses relatif sederhana unluk mengidentifikasi elemen-elemen data yang baru diciptakan.

- **Langkah 3 - Penolakan Setiap Elemen Data Sebagai Pembatas Potensia1.**
  Langkah kunci dalam menentukan pembatasan program / sistem adalah pada proses penolakan. Pemakaian individual *tools* uji volume harus merulnyakan pertanyaan-pertanyaan berikut berkaitan dengan setiap elemen data:
  - Dapatkah nilai data pada sebuah field memasuki sistem melampaui ukuran elemen data tersebut (jikalau demikian, sebuah pembatas perlu ditetapkan)
  - Apakah nilai pada sebuah field terakumulasi (jikalau demikian, seb~h pembatas perlu ditetapkan).
  - Apakah data sementara disimpan di dalam komputer (jikalau demikian, sebuah pembatas perlu ditetapkan).
  - Apakah informasi pada sebuah elemen data tersimpan di dalam program sampai sebuah transaksi berikutnya dimasukkan (jikalau demikian, sebuah pembatas perlu ditetapkan).
  - Jika sebuah elemen data menyatakan sebuah entitas akuntansi, misalkan jwnlah rekening dari penjualan finansial, sebagaimana pemakaian sebuah field karakter untuk mengidentifikasi distrik penjualan (jikalau demikian, sebuah pembatas perlu ditetapkan).

- **Langkah 4 - Pembatasan Dokumen.**
  Seluruh pembatasan yang diidentiflkasi pada langkah 3 harus didokumentasikan. Formulir ini sebagai dasar untuk pengujian volume. Setiap pembatasan itu harus dievaluasi untuk menentukan perluasan pengujian pada pembatasan tersebut.

- **Langkah 5 - Membentuk Pengujian Volume.**
  Pengujian yang dibentuk mengikuti kesembilan skema langkah yang sama pada metoda *test deck.* Pembatasan yang didokumentasikan pada langkah 4 menjadi kondisi uji yang perlu diidentifikasikan pada langkah 2 dari metodologi *test deck.* Proses uji selanjutnya mengikuti langkah 3 - 9.

ProsesUji Tahap Uji

Program uji mendaftar kriteria uji, bersama dengan uji yang direkomendasi, ditambah teknik uji dan *tool* yang digunakan dalam melengkapi uji tersebut. Tim uji harus mer.ggunakan keputusan dalam menyeleksi teknik uji yang tepat dan *tool* yang bergantung pada keperluan uji sistem aplikasi.

Pengujian efektif selama tahap uji harus menggunakan sebuah rencana uji yang dibentuk lebih awal dalam daur hidup. Pengujian tahap uji merupakan puncak dari pekerjaan sebelumnya untuk menyiapkan tahap ini. Tanpa persiapan ini, pengujian mungkin tidak ekonomis dan tidak efektif.

Referensi : Perry, W.E; A Structured Approach to Systems Testing; QED Information Science; 1983