

Black box testing



ISQA

Introduction

- Pengujian Black Box berfokus pada kebutuhan fungsional perangkat lunak
 - Tester dapat memperoleh set kondisi input yang sepenuhnya akan melaksanakan semua persyaratan fungsional untuk suatu program.
- Pengujian Black Box bukan merupakan alternatif pengujian whitebox, melainkan merupakan pelengkap yang kemungkinan akan mengungkap kelas kesalahan yang berbeda dari metode whitebox.
- Blackbox → Upaya untuk menemukan:
 - Fungsi tidak benar atau hilang
 - kesalahan antarmuka
 - kesalahan dalam struktur data atau akses database eksternal
 - kesalahan kinerja
 - Kesalahan inisialisasi dan penghentian.
- Tes ini dirancang untuk menjawab pertanyaan berikut
 - Bagaimana validitas fungsional diuji?
 - Kelas input apa yang akan membuat kasus uji yang baik?
 - Apakah sistem sangat sensitif terhadap nilai input tertentu?
 - Bagaimana batas-batas kelas data diisolasi
 - Tingkat data dan volume data apa yang dapat mentolerir sistem?
 - Efek apa yang akan mengkombinasi spesifik data terhadap operasi sistem?

Definisi

- Black Box testing merupakan strategi testing dimana hanya memperhatikan/memfokuskan kepada faktor fungsionalitas dan spesifikasi perangkat lunak. Berbeda dengan white box, black box testing tidak membutuhkan pengetahuan mengenai, alur internal (internal path), struktur atau implementasi dari software under test (SUT).

Kategori Kesalahan yang Diuji oleh Black Box Testing

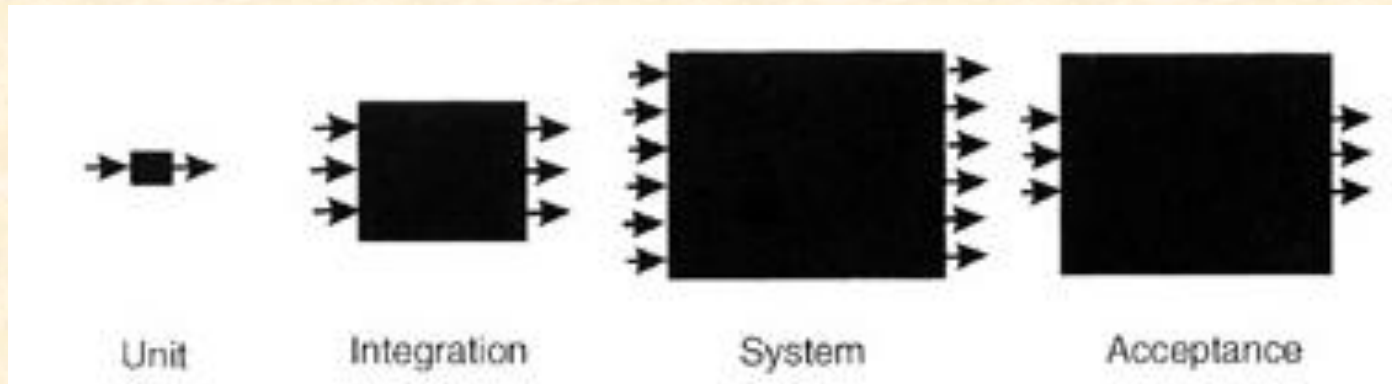
1. Fungsi-fungsi yang salah atau hilang.
2. Kesalahan interface.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan performa.
5. Kesalahan inisialisasi dan terminasi

Proses Black Box Testing

- Menganalisis kebutuhan dan spesifikasi dari perangkat lunak.
- Pemilihan jenis input yang memungkinkan menghasilkan output benar serta jenis input yang memungkinkan output salah pada perangkat lunak yang sedang diuji.
- Menentukan output untuk suatu jenis input.
- Pengujian dilakukan dengan input-input yang telah benar-benar diseleksi.
- Melakukan pengujian.
- Perbandingan output yang dihasilkan dengan output yang diharapkan.
- Menentukan fungsionalitas yang seharusnya ada pada perangkat lunak yang sedang diuji.

Ruang Lingkup Black Box

- *Black Box testing* dapat dilakukan pada setiap level pembangunan sistem. Mulai dari *unit, integration, system, dan acceptance*.



Keunggulan dan kelemahan Black Box

- Meskipun dalam pelaksanaan testing kita dapat menguji keseluruhan fungsionalitas perangkat lunak namun formal *black box testing* yang sebenarnya kita dapat memilih *subset test* yang secara efektif dan efisien dapat menemukan cacat. Dengan cara ini *black box testing* dapat membantu memaksimalkan *testing investment*.
- Ketika *tester* melakukan *black box testing*, *tester* tidak akan pernah yakin apakah perangkat lunak yang diuji telah benar-benar lolos pengujian.

Black Box Testing

- Equivalence Partitioning
- Boundary Value Analysis/Limit Testing
- Comparison Testing
- Sample Testing
- Robustness Testing
- Behavior Testing
- Requirement Testing
- Performance Testing
- Endurance Testing
- Cause-Effect Relationship Testing

Equivalence Class Testing

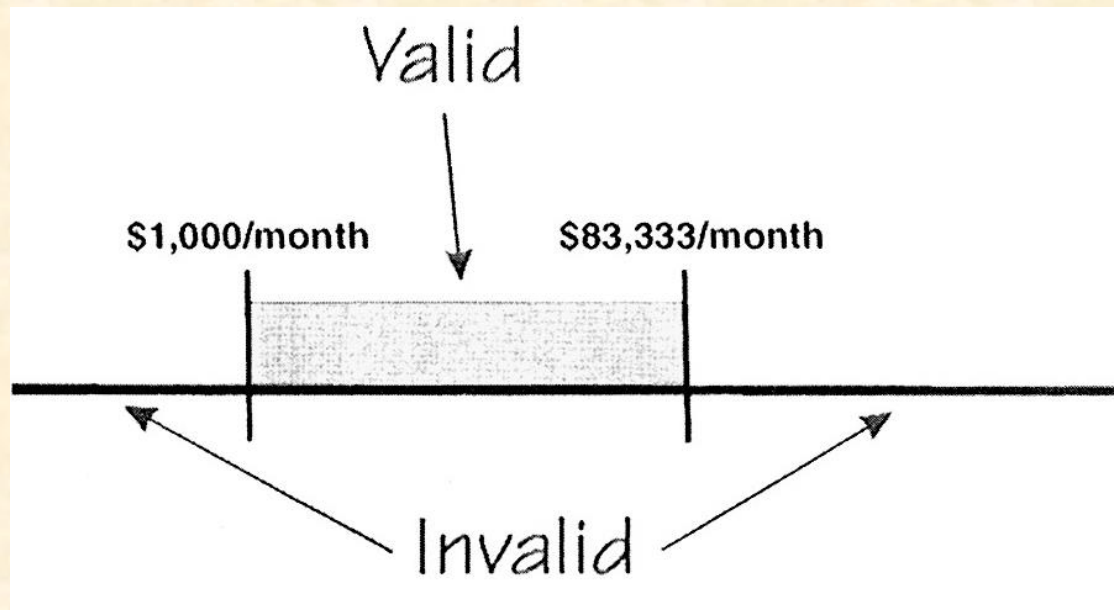
- Merupakan teknik yang digunakan untuk mengurangi jumlah test case yang ada pada saat pengujian. Kebanyakan tester menggunakan teknik yang simpel ini meskipun secara formal tester tersebut tidak mengetahui mengenai metode desain formal dalam pengujian perangkat lunak.
- Kasus uji yang didesain untuk *Equivalence class testing* berdasarkan pada evaluasi dari ekuivalensi jenis/class untuk kondisi input. *Class-class yang* ekuivalen merepresentasikan sekumpulan keadaan valid dan invalid untuk kondisi input. Biasanya kondisi input dapat berupa spesifikasi nilai numerik, kisaran nilai, kumpulan nilai yang berhubungan atau kondisi boolean.

Langkah-langkah *Equivalence Class Testing*

- Identifikasi kelas-kelas yang ekuivalen (*equivalence class*).
- Buat *test case* untuk tiap-tiap *equivalence class*.
- Jika memungkinkan buat test case tambahan yang acak yang memungkinkan ditemukannya cacat pada perangkat lunak.

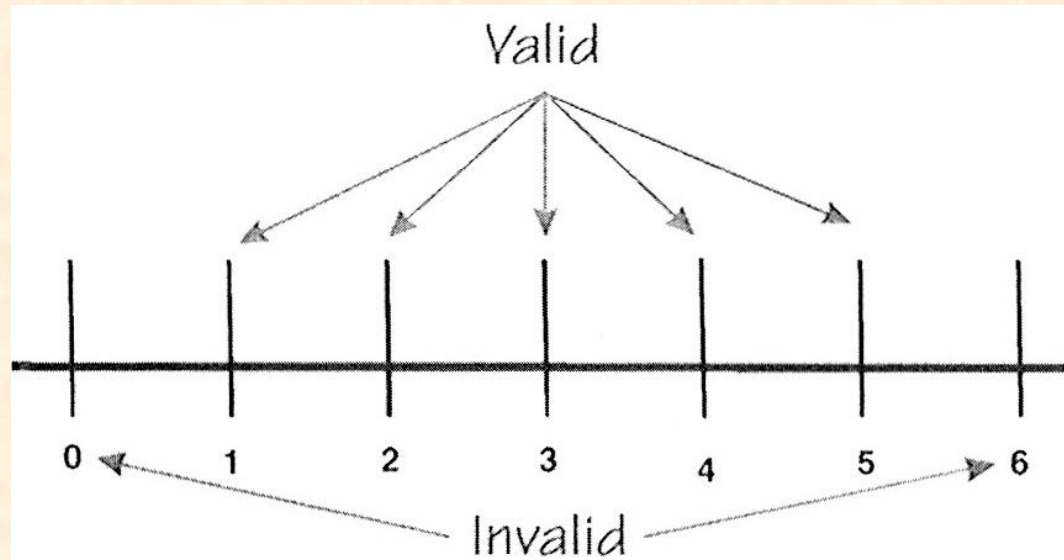
Continuous equivalence classes

Contoh nilai untuk pendapatan/ *salary* yang disyaratkan untuk melakukan pembelian rumah secara kredit:



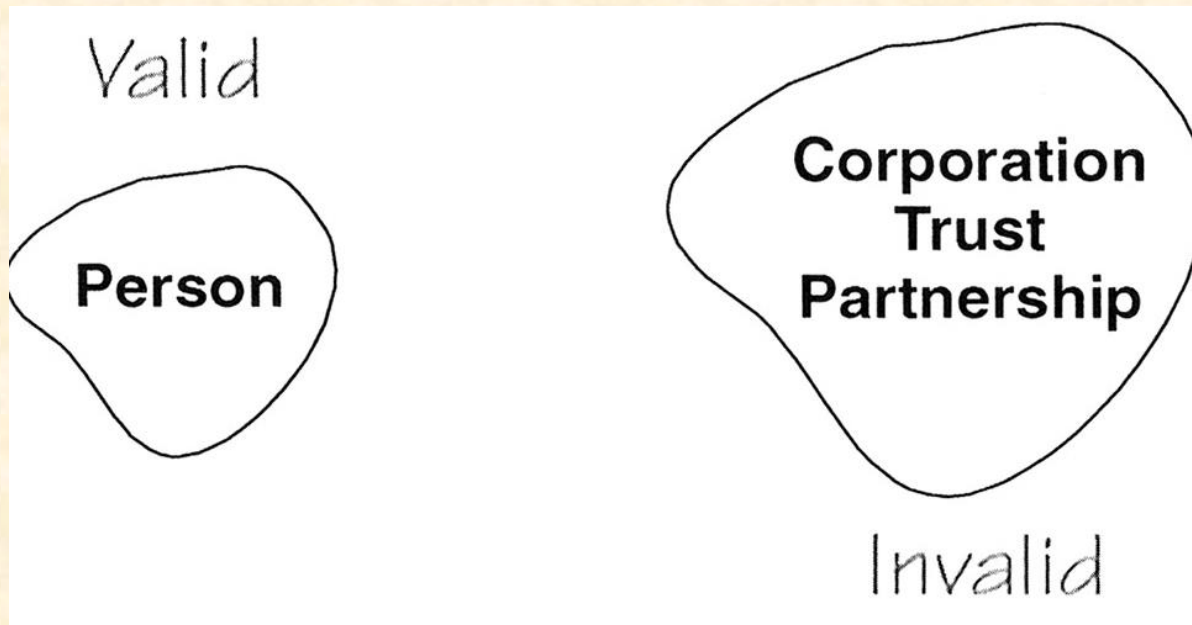
Discrete equivalence classes

Contoh nilai untuk jumlah kemilikan rumah yang
disyaratkan untuk melakukan pembelian rumah secara
kredit:



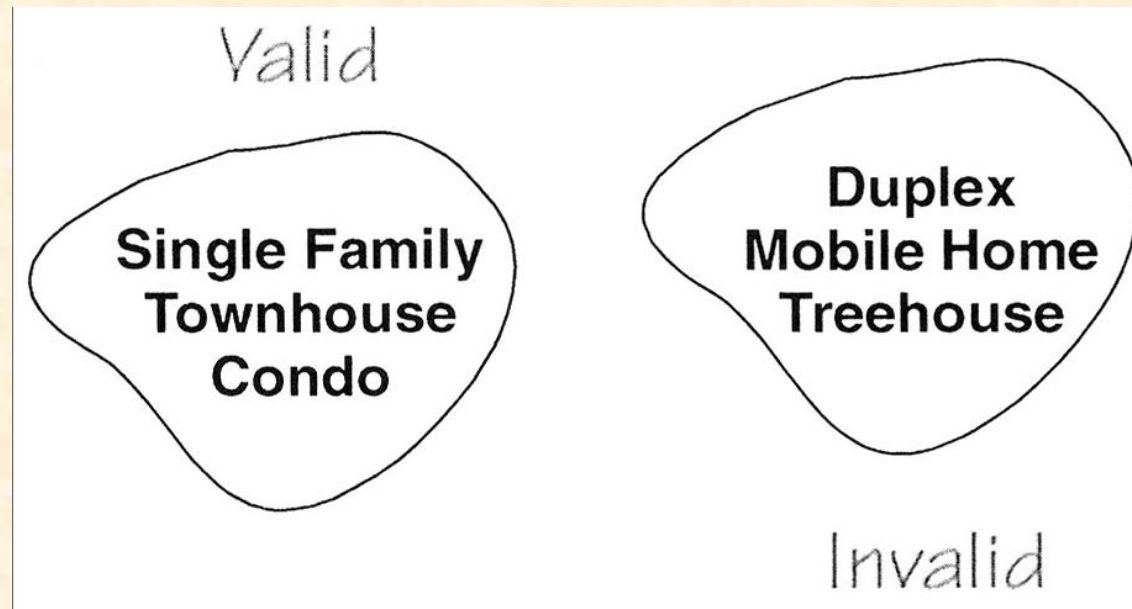
Single selection equivalence classes

Contoh nilai untuk kategori pengajuan yang
disyaratkan untuk melakukan pembelian rumah secara
kredit:



Multiple selection equivalence class

Contoh nilai untuk jenis rumah yang disyaratkan untuk melakukan pembelian rumah secara kredit:



Boundary Value Testing

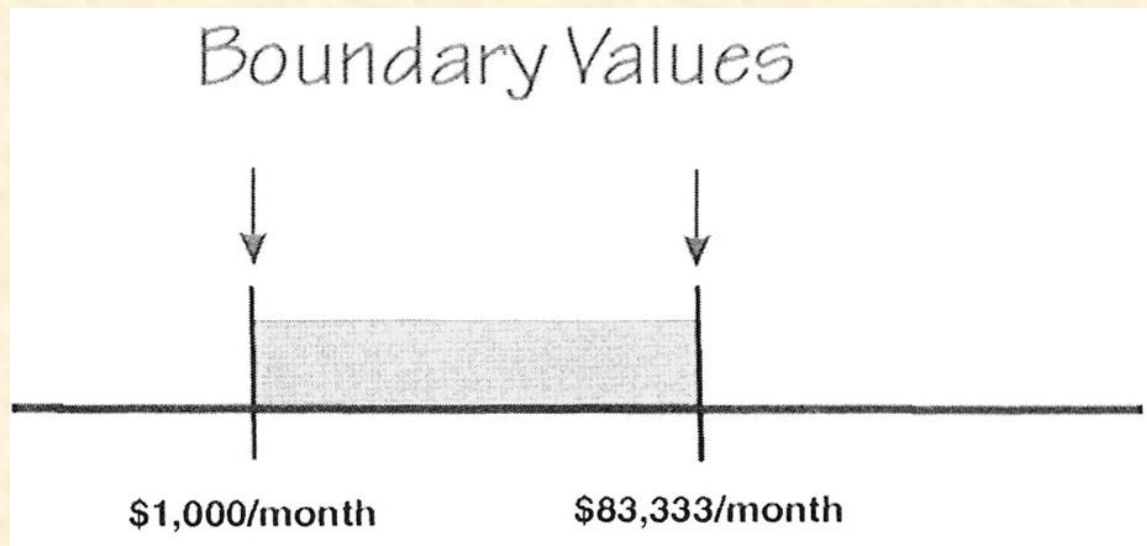
- *Boundary value testing* fokus kepada suatu batasan nilai dimana kemungkinan terdapat cacat yang tersembunyi.
- BVT mengarahkan pada pemilihan kasus uji yang melatih nilai-nilai batas. BVT merupakan desain teknik kasus uji yang melengkapi *Equivalence class testing*. Dari pada memfokuskan hanya pada kondisi input, BVA juga menghasilkan kasus uji dari domain output.

Langkah-langkah Boundary Value Testing

- Identifikasi kelas-kelas yang ekuivalen (*equivalence class*).
- Identifikasi batasan untuk tiap equivalence class.
- Buat *test case* untuk tiap batasan suatu nilai dengan memilih titik pada batasan, satu titik pada nilai bawah batasan dan satu titik pada nilai atas batasan.

Boundary values for a continuous range of inputs

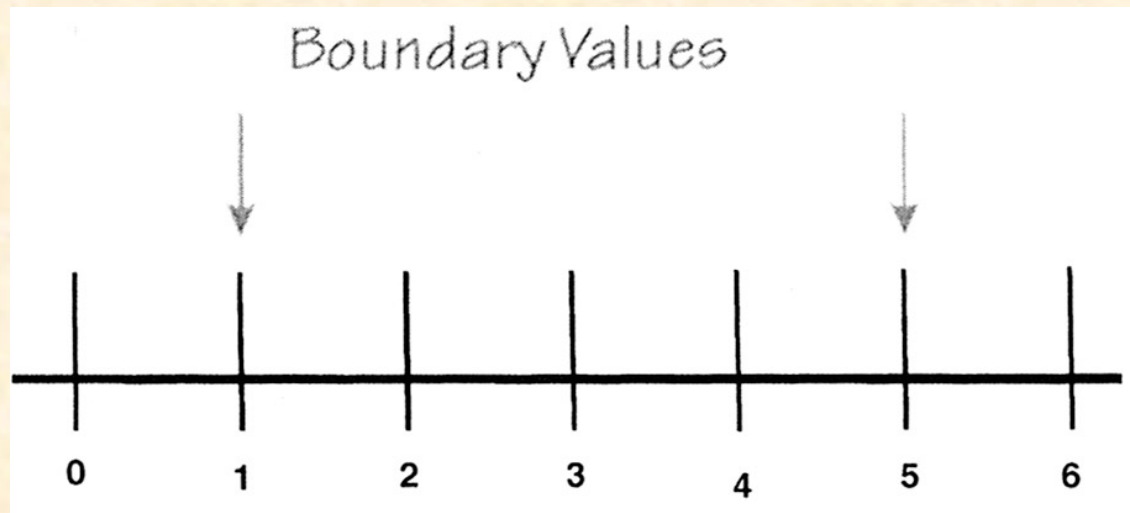
Contoh untuk nilai pendapatan/ salary:



Tes data input untuk batas bawah adalah {\$999, \$1,000, \$1,001} dan untuk batas atas {\$83,332, \$83,333, \$83,334}.

Boundary values for a discrete range of inputs.

Contoh nilai untuk jumlah tempat tinggal (dwellings) yang dimiliki oleh seseorang:



Tes data input untuk batas bawah adalah $\{0, 1, 2\}$ dan untuk batas atas $\{4, 5, 6\}$.

Contoh Kombinasi Pengujian

- Sangat penting untuk menginputkan nilai kombinasi secara bersamaan, misalkan:

Monthly Income	Number of Dwellings	Result	Description
\$1,000	1	Valid	Min income, min dwellings
\$83,333	1	Valid	Max income, min dwellings
\$1,000	5	Valid	Min income, max dwellings
\$83,333	5	Valid	Max income, max dwellings
\$1,000	0	Invalid	Min income, below min dwellings
\$1,000	6	Invalid	Min income, above max dwellings
\$83,333	0	Invalid	Max income, below min dwellings
\$83,333	6	Invalid	Max income, above max dwellings
\$999	1	Invalid	Below min income, min dwellings
\$83,334	1	Invalid	Above max income, min dwellings
\$999	5	Invalid	Below min income, max dwellings
\$83,334	5	Invalid	Above max income, max dwellings

Equivalence Partitioning

- Bagilah domain input ke dalam kelas data yang dapat dihasilkan uji kasus .
- Mencoba untuk mengungkap kelas kesalahan.
- Berdasarkan kelas ekivalensi untuk kondisi input.
- Sebuah kelas ekivalensi merupakan sekeumpulan status valid atau tidak valid
- Kondisi input adalah berupa suatu nilai numerik, kisaran nilai, suatu nilai-nilai yang terkait, atau kondisi boolean.
- Kelas ekivalensi dapat didefinisikan oleh:
 - Jika kondisi input menspesifikasikan kisaran atau nilai tertentu, satu valid dan dua tidak valid yang ditetapkan kelas ekivalensi.
 - Jika kondisi input menspesifikasikan boolean atau anggota dari suatu himpunan, satu valid dan satu tidak valid yang ditetapkan kelas ekivalensi.
- Kasus uji untuk setiap domain item masukan data dibuat dan dijalankan.

Comparison Testing

- In some applications the reliability of SW is critical.
 - Aircraft avionics, nuclear power plant control
- Redundant hardware and software may be used to minimize error.
- For redundant SW, use separate teams to develop independent versions of the software.
- Test each version with same test data to ensure all provide identical output.
- Run all versions in parallel with a real-time comparison of results.
- Even if will only run one version in final system, for some critical applications can develop independent versions and use *comparison testing* or *back-to-back testing*.
- When outputs of versions differ, each is investigated to determine if there is a defect.
- Method does not catch errors in the specification.

Sample Testing

- Sample testing involves selecting a number of values from an equivalence class of input data
- Integrate the values into test cases
- These values may be chosen at constant or variable intervals

Robustness Testing

- Data are chosen outside the range defined by the requirements
- The purpose of this test is to prove that no catastrophic event can be produced by the introduction of an abnormal input data item

Behaviour Testing

- A behavior tests is a test for which the result obtained cannot be evaluated after a single execution of a CSU program, but can be evaluated only after a set of linked calls to subprogram (several calls to a given subprogram, a call to several different subprogram)
 - E/g : stack

Requirement Testing

- The requirement associated with the software (input/output/function/performance) are identified during the software specification and design activities.
- Requirement testing involves producing a test case for each requirement relating to CSU
- To facilitate the implementing of such tests, each requirement may be traced to final test case trough the traceability matrix

Req.	Spec.	Pre-Design	Det-Design	CSU	Test Proc.	Test Rpt

Performance Testing

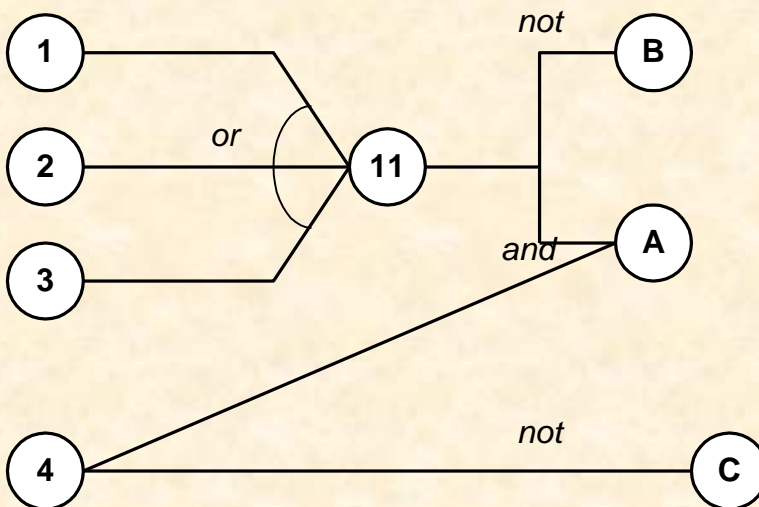
- Performance testing evaluates the csu's ability to operate correctly with regard to the requirement concerning, for example:
 - data flow, memory size, execution time, etc
- To meet under certain workload or configuration conditions, provided, however, that these requirements can be reflected at the level of the CSU
- These requirements are defined during specification or design activities
- Performance testing may also be used to measure and explore the performance limits of a CSU

Cause-effect Relationship Testing

- This technique supplements equivalence testing by providing ways of determining and selecting combinations of input data
- It involves representing the input state (cause) and output states (effect) deduced from functional requirements associated with the CSU in the form of a graph represented by logical relationship (decision table), to avoid of having defined too many test cases
- The Steps are
 - Break down the requirements into subset within which it will possible to work
 - Define cause and effect based on the requirement
 - Analyze the requirements to make a logical relationship
 - Highlight the graph, the impossibilities of combinations of cause/effect due to constraint from the requirements
 - Convert the graph into a decision table
 - Column --> test case
 - Row --> cause/effect
 - Convert the columns of the decision table into test cases

Example

- A CITROEN car will be identified by a letter A, B or C, and have the letter X as the second character. Messages M1 and M2 must be sent in the event of an error in the first or second character respectively. If the identifier is correct, it is inserted in the database.
- The the input state (the causes) and the output states (the effect) can be determined:



Input states:

1. 1st character: A
2. 1st character: B
3. 1st character: C
4. 2nd character: X

Output states:

- A. database insertion
- B. message M1
- C. message M2

	1	2	3	4	5
1	0	1	0	0	
2	0	0	1	0	
3	0	0	0	1	
4		1	1	1	0
A		1	1	1	
B	1				
C					1

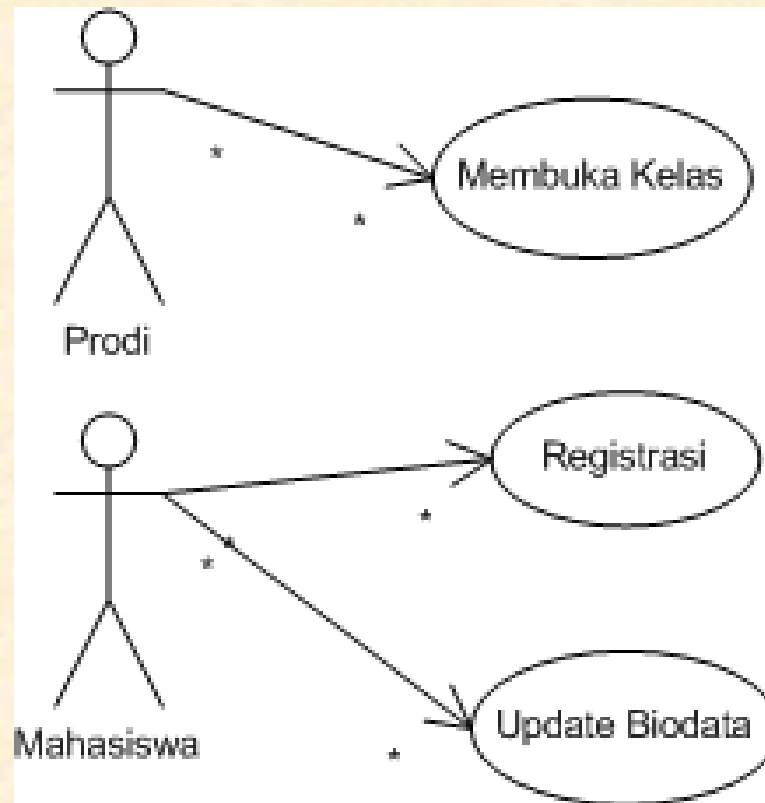
Use Case Testing

- Mendefinisikan transaksi pada proses-proses yang ada pada suatu sistem adalah hal yang sangat penting untuk dilakukan pada proses pendefinisian kebutuhan sistem (*requirements definition*).
- Terdapat banyak pendekatan untuk mendokumentasikan transaksi-transaksi tersebut seperti flowchart, HIPO diagram, dan teks. Sekarang pendekatan yang paling populer dilakukan adalah dengan menggunakan use case diagram.
- Use case biasanya dibuat oleh developer dan untuk developer, namun demikian dalam pengujian perangkat lunak, informasi yang ada pada use case sangat berguna bagi tester.

Fungsi dari use case



- Menggambarkan *functional requirement* dari sebuah sistem dari perspektif pengguna bukan dari perspektif teknik dan mengabaikan paradigma pengembangan yang digunakan.
- Dapat digunakan untuk proses identifikasi kebutuhan pengguna.
- Menyediakan dasar untuk komponen internal sistem, struktur, database, dan keterhubungan.
- Menyediakan dasar dalam membangun *test case* dalam sistem dan *acceptance level*.

Contoh Use Case





Example use case.


Use Case Component	Description
Use Case Number or Identifier	SIM153
Use Case Name	Registrasi
Goal in Context	
Scope	<i>System</i>
Level	<i>Primary task</i>
Primary Actor	Mahasiswa
Preconditions	None
Success End Conditions	Mahasiswa terdaftar dalam kelas untuk matakuliah(kelas diajar oleh dosen)
Failed End Conditions	Daftar matakuliah mahasiswa yang diambil oleh mahasiswa tidak bisa berubah
Trigger	Mahasiswa memilih matakuliah dan mengambil matakuliah tersebut.
Main Success Scenario	StepAction
A: Actor	1 S: menampilkan daftar kelas yang dibuka
S: System	2 A: Memilih kelas dan menambahkan kelas
	3 S: melakukan filter terhadap SKS dan Kuota
	4 Mahasiswa telah selesai melakukan semua proses registrasi
	2a S: Mencegah mahasiswa mengulang registrasi dan menampilkan pesan Matakuliah yang diambil memiliki pre-requisite
	4a S: Mencegah mahasiswa mengambil matakuliah tersebut jika belum mengambil matakuliah pre-requisite dan menampilkan pesan Kelas yang diambil penuh
Extensions	4b S: Mencegah mahasiswa mengambil kelas tersebut.

- 
- 
- Dengan syarat tiap-tiap use case telah benar-benar dipastikan sebelum implementasi. Untuk menguji suatu use case aturan dasar yang harus diperhatikan adalah membuat minimal satu test case untuk main success scenario dan setidaknya satu test case ***extension*** pada use case.
 - Karena use case tidak menspesifikasikan data input, maka tester harus menentukan data input tersebut. Secara umum teknik-teknik yang telah dijelaskan sebelumnya seperti *the equivalence class and boundary value techniques*.

Langkah-Langkah Use Case Testing

- Sangat penting untuk mempertimbangkan resiko dari transaksi dan jenis-jenisnya pada saat pengujian. Transaksi-transaksi yang memiliki resiko rendah diuji dengan intensitas rendah sebaliknya transaksi-transaksi yang resikonya tinggi harus diuji dengan intensitas yang tinggi.
- Untuk membuat test case, mulailah dengan data yang normal dan sering digunakan dalam transaksi. Kemudian pilih transaksi-transaksi yang jarang dilakukan tapi merupakan transaksi yang vital bagi sistem, misalkan transaksi mematikan sistem,dll.

- 
- 
- Pastikan setiap extension pada use case telah diuji. Ujilah dengan kondisi-kondisi dan sesuatu yang ekstrim serta langgarlah ketentuan-ketentuan yang diberikan oleh sistem. Jika suatu transaksi memiliki suatu perulangan/loop, teliti dan cobalah perulangan tersebut berulang-ulang dan jika terdapat alur yang rumit dalam suatu transaksi ujilah transaksi tersebut dengan cara yang ekstrim misalkan uji dengan alur yang tidak seharusnya dilakukan.



Komponen utama dari pengujian suatu transaksi adalah data testing. Boris Beizer dalam bukunya menyarankan bahwa 30 s/d 40 persen dari pengujian suatu transaksi adalah generating, capturing, or extracting test data. Dan jangan dilupakan bahwa kegiatan tersebut juga membutuhkan resource waktu dan personel dalam suatu project.

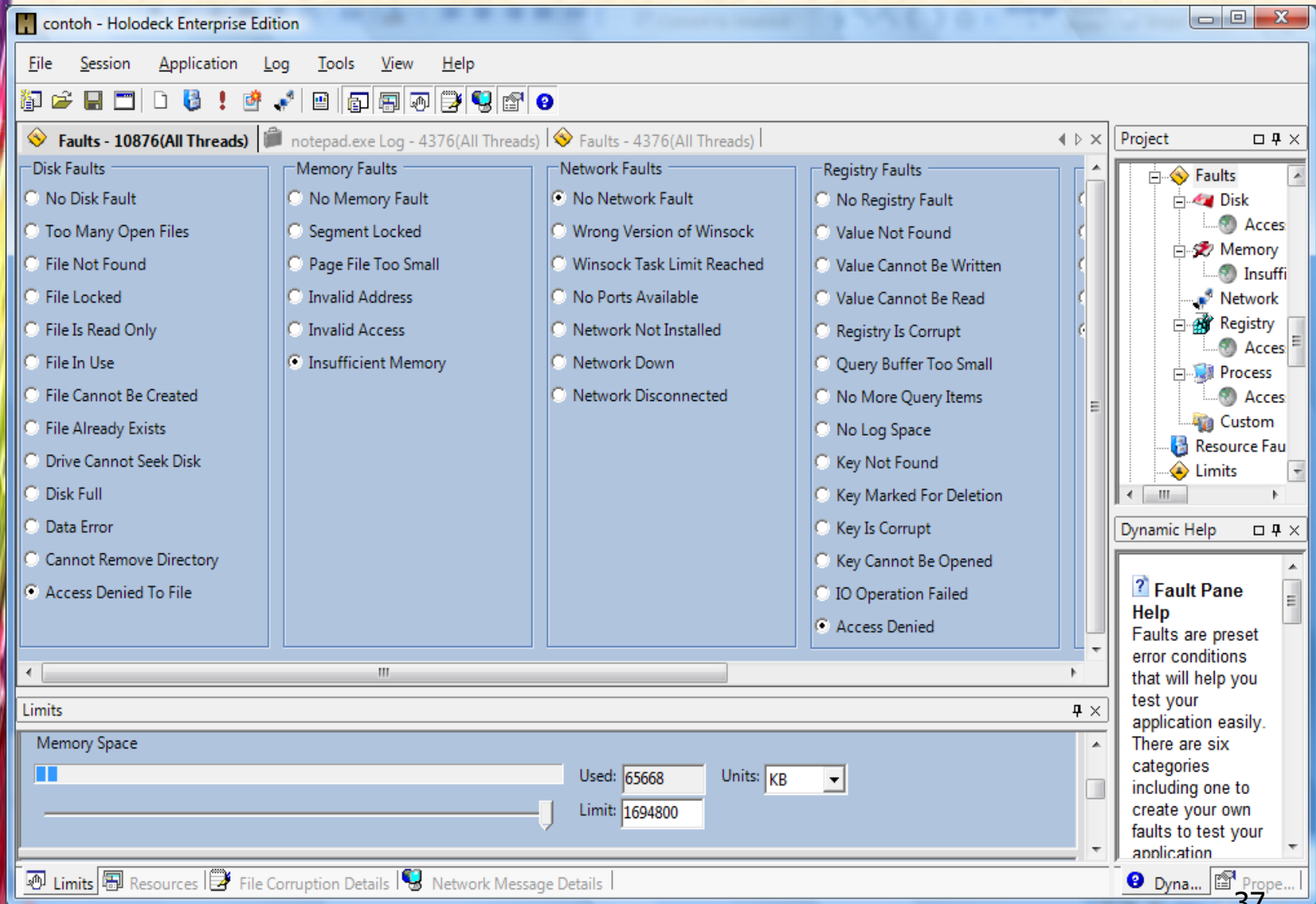
Menguji *exceptional condition*

- Pengujian mengenai alur dari suatu transaksi relatif mudah dilakukan karena sebagian besar hanya memperhitungkan valid atau tidak validnya data yang dimasukkan.
- Hal yang lebih sulit dilakukan adalah menguji *exceptional condition* seperti *low memory, disk full, connection lost, driver not loaded*, dll. Tester membutuhkan waktu yang sangat banyak untuk melakukan simulasi tersebut jika dilakukan secara manual. Untuk itu telah tersedia sebuah tool yang bernama **holodeck** yang dibuat oleh James Whittaker dan timnya dari *Florida Institute of Technology*. Holodeck melakukan monitor terhadap interaksi antara aplikasi dan sistem operasi. Holodeck melakukan pencatatan terhadap aktifitas sistem dan memungkinkan tester untuk melakukan simulasi seperti *low memory, disk full, connection lost, driver not loaded*, dll.

Holodeck

- Holodeck is a unique test tool that uses fault injection to simulate real-world application and system errors for Windows applications and services - allowing testers to work in a controlled, repeatable environment to analyze and debug error-handling code.
- Holodeck is the first commercial fault-simulation tool and was developed by leading researchers in the application quality field. It is used by organizations like Microsoft, Adobe, EMC and McAfee.

Holodeck



Contoh kasus-1

- Program mencatat pembayaran sejumlah barang dari tabel barang. Jika data valid akan menghitung nilai bayar jika data tdk valid akan muncul pesan kesalahan. Data yang diinput adalah kode barang dan jml barang. Isi tabel barang : kd_barang,nama brg, satuan, harga
- Buatlah kelas ekivalensi untuk data input
- buat tabel test case utk menguji program tsb.

Tabel pengujian

Asumsi : kd-brg = 5 digit , karakter pertama A :ATK, B : lainnya

NO	Data Uji	Input	Hasil yg diharapkan	Output testing	Hasil uji/kesimpulan
1	Kd-brg lookup tabel, Jml > 0	Kd-brg=A0010, Jml=2	Hitung nilai bayar	Hitung nilai bayar	OK/sukses
2	Kd-brg kosong, jml > 0	Kd-brg= ` `, Jml=1	Pesan kesalahan	0	Invalid/tdk sesuai
3	Kd-brg lookup tabel, Jml < 0	kd-brg=B0001, Jml= -2	Pesan kesalahan	Pesan kesalahan	Ok
4	Kd-brg lookup tabel, Jml =0	kd-brg=A0001, Jml= 0	Pesan kesalahan	Pesan kesalahan	OK
5	Kd-brg = baru, Jml >0	Kd-brg=99999, Jml=3	Pesan kesalahan	Pesan kesalahan	OK RA /39

Tabel pengujian

No	Data uji	Input	Hasil yg diharapkan	Output	Kesimpulan
1	Kd_brg=kosong, jumlah>0	Kd-Brg= ` ` , Jumlah = 2	Pesan kesalahan	Pesan kesalahan	OK/sukses
2	Kd_brg=look up tabel, jumlah >0	Kd-Brg= 11111, Jumlah = 2	perhitungan	Harga =5000, Nilai = 10000	OK
3	Kd_brg= baru, jumlah >0	Kd-brg = 900099, Jumlah=3	Pesan kesalahan	perhitungan	invalid

dst. Sampai dimungkinkan ketemu cacat program

Kasus-2

- Program mencatat nilai uts mahasiswa dari suatu kuliah. Jika data valid akan menyimpan nilai UTS ke tabel nilai, jika data tdk valid akan muncul pesan kesalahan. Data yang diinput adalah mata kuliah (kd-kuliah), NPM dan nilai UTS (integer). Isi tabel mahasiswa : NPM, nama mhs. Isi tabel nilai: npm, kd-kuliah, nUTS
- Buatlah test case utk menguji program tsb.

Data uji

■ Data valid

1. Isi Kd-kuliah dan/atau NPM sesuai dgn spesifikasi
2. Kd-kuliah lookup tabel
3. NPM lookup tabel
4. $0 \leq \text{NUTS} \leq 100$

■ Data invalid

1. Kd-kuliah : baru
2. NPM : baru
3. Kd-kuliah dan / atau NPM dikosongkan
4. $\text{NUTS} < 0$
5. $\text{NUTS} > 100$
6. NUTS : huruf
7. NUTS bil real

Latihan (buat tabel pengujiannya)

1. Program untuk menginput 2 bilangan bulat kemudian mencari nilai maksimum dr kedua bil tsb.
2. Program untuk menampilkan keterangan terbilang dari bilangan /uang (bulat positif) yang diinput. Nilai terbesar adalah 5 milyar
 - Contoh :
 - Jumlah uang : 1650
 - Terbilang : seribu enam ratus lima puluh

Data uji

■ Data valid

1. $Bil1 \geq 0$, $bil2 \geq 0$ integer
2. Salah satu < 0
3. Keduanya < 0
4. $Bil1 > bil2$
5. $Bil1 = bil\ 2$
6. $Bil1 < bil2$

■ Data invalid

1. Bil real
2. Bukan bilangan (karakter)
3. Bil dikosongkan

GUI test → black box

2 3 4

LOGIN 1

User name : 5

Password : 6 7

8 9 10

Clear Ubah OK

Tabel pengujian terdiri dari :

No	Input	Hasil yang diharapkan	Output tes	Hasil Uji
1	Klik no.2	Kotak dialog menjadi minimum		

Testing Internet Applications



Internet applications

- Internet applications are essentially client-server applications in which the client is a Web browser and the server is a Web or application server.
- Some companies have applications built for
 - business-to-consumer uses such as banking services or retail stores,
 - business-to-business applications such as supply chain management.

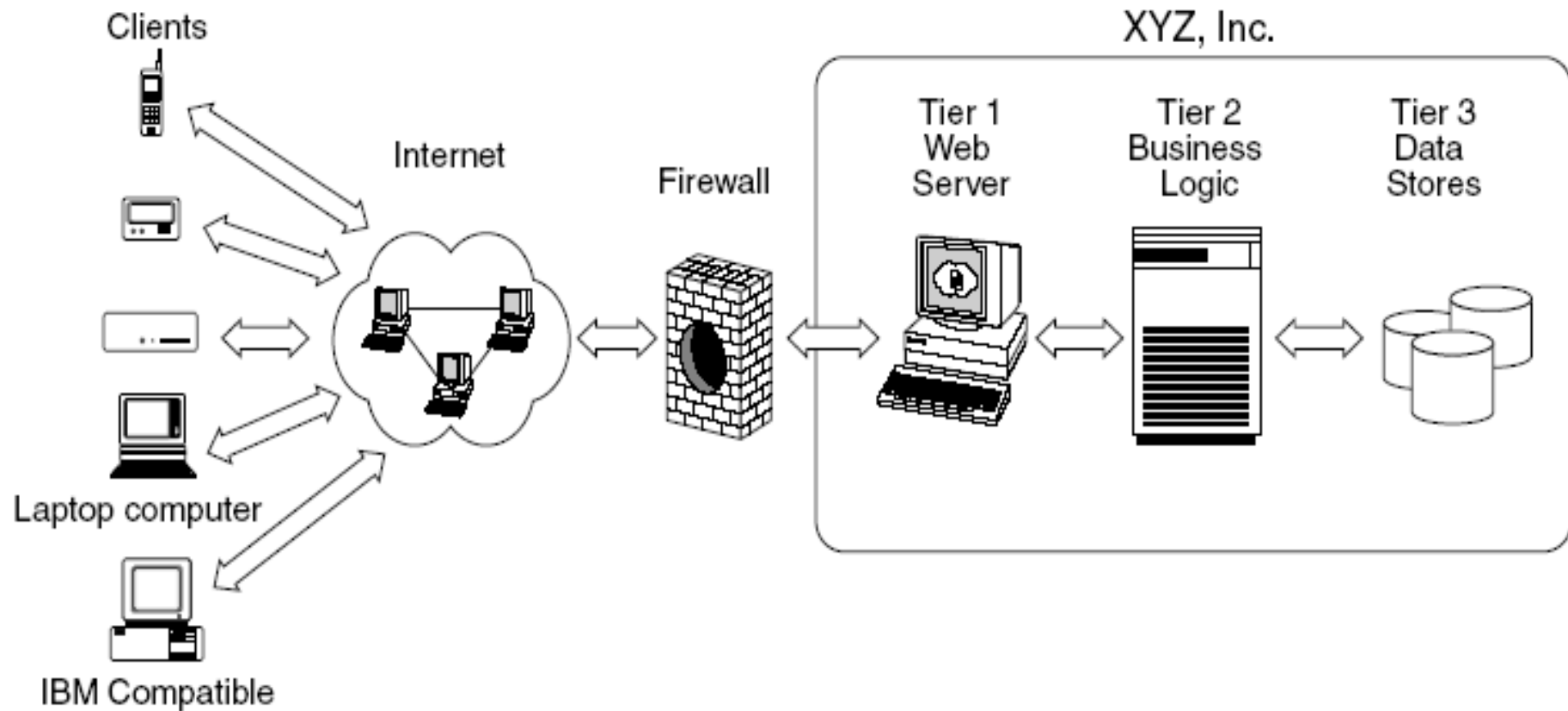
Tujuan *testing Internet-based applications*

- to uncover errors in the application before deploying it to the Internet.
- given the complexity of these applications and the interdependency of the components, you likely will succeed in finding plenty of errors.

Internet-based Architecture

- three-tier client-server (C/S) architecture:
 - each tier is treated as a black box with well-defined interfaces
 - to change the internals of each tier without worrying about breaking another tier.

Typical architecture of an e-commerce site



Internet three-tier architecture

1. Web server/ *Presentation tier* or *layer* : use static HyperText Markup Language (HTML) pages or Common Gateway Interface (CGI) scripts to create dynamic HTML
2. *Business layer*
 - Transaction processing
 - User authentication
 - Data validation
 - Application logging
3. *Data layer* :
 - This tier consists of a database infrastructure to communicate with the second tier

Contoh Internet based Testing

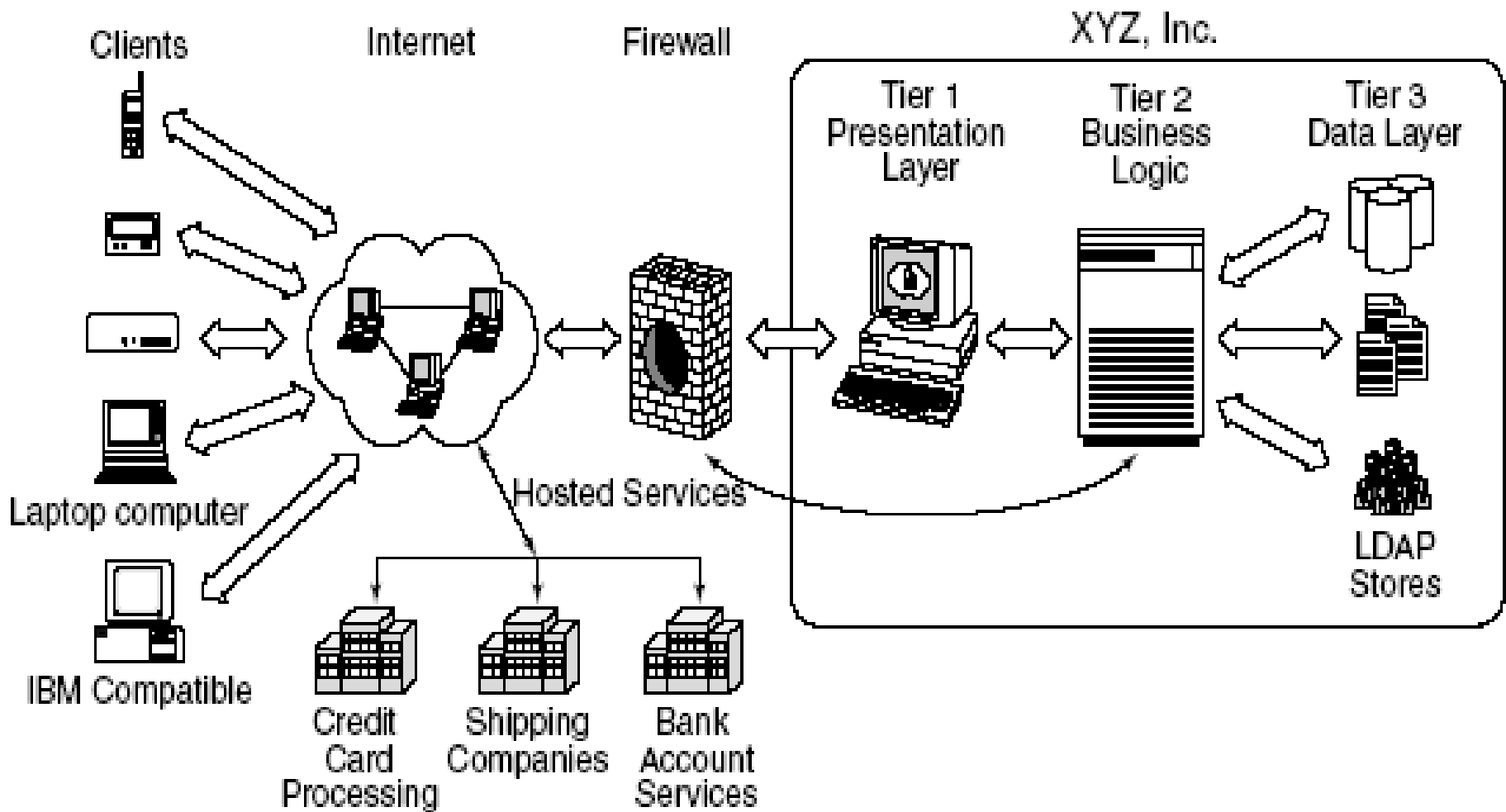
Examples of Presentation, Business, and Data Tier Testing

<i>Presentation Tier</i>	<i>Business Tier</i>	<i>Data Tier</i>
<ul style="list-style-type: none">• Ensure fonts are the same across browsers.• Check to make sure all links point to valid files or Websites.• Check graphics to ensure they are the correct resolution and size.• Spell-check each page.• Allow a copy editor to check grammar and style.• Check cursor positioning when page loads to ensure it is in the correct text box.• Check to ensure default button is selected when the page loads.	<ul style="list-style-type: none">• Check for proper calculation of sales tax and shipping charges.• Ensure documented performance rates are met for response times and throughput rates.• Verify that transactions complete properly.• Ensure failed transactions roll back correctly.• Ensure data are collected correctly.	<ul style="list-style-type: none">• Ensure database operations meet performance goals.• Verify data are stored correctly and accurately.• Verify that you can recover using current backups.• Test failover or redundancy operations.

Strategi Testing

1. *Presentation layer.* The layer of an Internet application that provides the GUI (graphical user interface).
2. *Business Logic layer.* The layer that models your business processes such as user authentication and transactions.
3. *Data Access layer.* The layer that houses data used by the application or that is collected from the end user.

Detailed view of Internet application architecture.



presentation layer testing

- 1. Content testing.* Overall aesthetics, fonts, color, spelling, content accuracy, default values.
- 2. Website architecture.* Broken links or graphics.
- 3. User environment.* Web browser versions and operating system configuration.

Items to Test in Each Tier

<i>Test Area</i>	<i>Comments</i>
Usability/human factors	<ul style="list-style-type: none">• Review overall look and feel.• Fonts, colors, and graphics play a major role in the application aesthetics.
Performance	<ul style="list-style-type: none">• Check for fast-loading pages.• Check for quick transactions.• Poor performance often creates a bad impression.
Business rules	<ul style="list-style-type: none">• Check for accurate representation of business process.• Consider business environment for target user groups.
Transaction accuracy	<ul style="list-style-type: none">• Ensure transactions complete accurately.• Ensure cancelled transactions roll back correctly.
Data validity and integrity	<ul style="list-style-type: none">• Check for valid formats of phone number, e-mail addresses, and currency amounts.• Ensure proper character sets.
System reliability	<ul style="list-style-type: none">• Test the failover capabilities of your Web, application, and database servers.• Maximize MTBF and minimize MTTR.
Network architecture	<ul style="list-style-type: none">• Test connectivity redundancy.• Test application behavior during network outages.

Business Layer Testing

- *Performance.* Test to see whether the application meets documented performance specifications (generally specified in response times and throughput rates).
- *Data validity.* Test to detect errors in data collected from customers.
- *Transactions.* Test to uncover errors in transaction processing, which may include items such as credit card processing, e-mailing verifications, and calculating sales tax.

Data Layer Testing

- *Response time.* Quantifying completion times for Data
- Manipulation Language (DML) (Structured Query Language [SQL] INSERTs, UPDATES, and DELETES), queries (SELECTs), and transactions.
- *Data integrity.* Verifying that the data are stored correctly and accurately.
- *Fault tolerance and recoverability.* Maximize the MTBF-mean time between failures and minimize the MTTR-mean time to recovery

Natal semakin bermakna dalam hangatnyanya kebersamaan

Segenap Direksi dan Karyawan PT Bank Central Asia Tbk
mengucapkan Selamat Hari Natal dan Tahun Baru 2007



1

Selamat Datang di situs



5

Kurs BCR

8-Jan-2007 / 15:37 WIB

KURS	JUAL	BELI
USD	9060.00	9010.00
SGD	5902.80	5851.80
HKD	1163.35	1155.05
CHF	7352.05	7294.05

kurs lainnya

6

Internet Banking

Login

INDIVIDUAL

demo

Login

BISNIS

demo

2

CORPORATE

3

SMALL MEDIUM
ENTERPRISE (SME)

4

INDIVIDUAL

Prioritas

Gold

Silver

Contoh-2



INDIVIDUAL

[HOME]

USER ID dan PIN Internet Banking dapat diperoleh pada saat Anda melakukan Registrasi Internet melalui ATM BCA. Untuk informasi lebih lanjut hubungi Halo BCA (021) 52999888.

HOW TO GET STARTED:
To start using BCA Internet Banking, You must first register through any BCA ATM. For further information, please contact Halo BCA (021) 52999888.

[[PRIVACY POLICY](#)]

Silakan memasukkan USER ID Anda

Please enter Your USER ID

Silakan memasukkan PIN Internet Banking Anda

Please enter Your Internet Banking PIN

LOGIN

Catatan : - Anda harus menggunakan 'KeyBCA' setiap kali Anda melakukan transaksi finansial.
- Situs ini hanya dapat ditampilkan dengan menggunakan Internet Explorer ver. 6.0 keatas.

Notes : - You have to use 'KeyBCA' to do financial transaction.
- This site can be viewed only with Internet Explorer ver. 6.0 and above.

Klik BCA is secured with
SSL 128 bit encryption.



Copyright © 2000  **BCA** All Rights Reserved