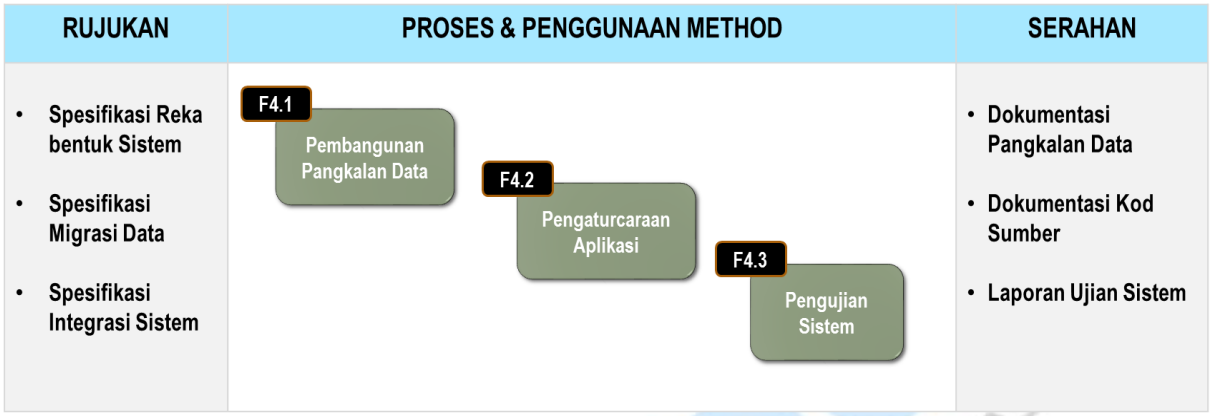
BAB:05

FASA PEMBANGUNAN

Bab ini menerangkan aktiviti-aktiviti pembangunan sistem aplikasi serta amalan terbaik dalam pengaturcaraaan, pembangunan pangkalan data dan pengujian yang perlu dilaksanakan ke atas sistem aplikasi.

5 PEMBANGUNAN

5.1 Gambaran Keseluruhan



Rajah 83 : Gambaran Keseluruhan Fasa 4 - Pembangunan

5.2 Pengenalan

Fasa pembangunan merupakan fasa yang sangat penting dalam Kitar Hayat Pembangunan Sistem (SDLC), pada fasa inilah bermulanya pembangunan sistem yang sebenar, fasa sebelum ini merupakan asas kepada pembangunan sistem. Manakala fasa seterusnya adalah untuk memastikan sistem memenuhi kehendak pengguna. Pada fasa ini kebanyakan fungsi-fungsi atau modul-modul sistem yang akan dibangunkan telah dimuktamatkan.

Matlamat utama fasa ini ialah untuk menterjemahkan atau merealisasikan SDS yang dihasilkan dalam fasa reka bentuk kepada kod aturcara dalam bahasa pengaturcaraan yang telah dipilih, memasang dan seterusnya melaksanakan pengujian sistem. Pengujian dilaksanakan agar sistem yang dibangunkan bebas dari sebarang ralat, dapat berfungsi sepenuhnya dan berjaya memenuhi keperluan sebenar pembangunan. Ini dapat memastikan sistem memenuhi kriteria-kriteria kualiti yang telah ditetapkan bagi sesebuah perisian serta meningkatkan tahap keyakinan pengguna ke atas perisian yang dibangunkan.

3 aktiviti utama dalam Fasa Pembangunan iaitu:

a) Pembangunan Pangkalan Data

b) Pengaturcaraan Aplikasi

c) Ujian Sistem

Dokumen Rujukan kepada Fasa Pembangunan adalah seperti berikut:

a) D04 Spesifikasi Reka bentuk Sistem

b) D06 Spesifikasi Migrasi Data

c) D08 Spesifikasi Integrasi Sistem

Dokumen Serahan kepada Fasa Pembangunan adalah seperti berikut:

a) D09 Dokumentasi kod Pangkalan Data

b) D10 Dokumentasi Kod Sumber

c) D11 Laporan Ujian Sistem

5.3 Penglibatan Pemegang taruh

Pemegang taruh utama yang patut terlibat dalam fasa pembangunan adalah Pengaturcara Program, Penguji Perisian, Pemilik prosidur/proses semasa dan Pengguna. Pengaturcara program perlulah berkeupayaan untuk menterjemahkan keperluan reka bentuk sistem kepada kod program dan logik pembangunan untuk menghasilkan sistem yang berkualiti dan memenuhi kehendak pelanggan bagi memastikan sistem yang dibangunkan bertepatan dengan keperluan. Manakala Penguji Sistem pula perlulah berkeupayaan membangunkan pelan pengujian dan skrip ujian berdasarkan SRS dan SDS yang diberikan.

Cadangan penglibatan kategori pemegang taruh adalah seperti berikut:

a) Pengurus Projek

Memantau kemajuan pembangunan supaya mengikut perancangan, mengenalpasti risiko semasa pembangunan, berkomunikasi antara pembangunan dengan SME.

b) Pengaturcara Program

Menulis kod aturcara untuk menghasilkan sistem aplikasi

c) Penguji Sistem

Menguji sistem bagi memastikan sistem aplikasi yang dibangunkan berkualiti dan sedia untuk dilancarkan.

d) Pemilik Prosidur/Proses Semasa dan Pengguna

Melaksanakan ujian penerimaan pengguna.

5.4 Faktor Kejayaan

Untuk memastikan aktiviti berjaya dilaksanakan, berikut adalah faktor kejayaan utama yang perlu dipertimbangkan sebelum dan semasa aktiviti dilaksanakan:

a) Spesifikasi Reka bentuk Sistem (SDS) yang didokumenkan adalah lengkap dan memenuhi kehendak pengguna.

b) Pasukan pengaturcara program berupaya menterjemahkan SDS kepada kod aturcara dan logik pemprograman, mempunyai kemahiran dan kepakaran dalam bahasa pengaturcaraan yang dipilih dan SQL (*skill coding dan SQL query*).

c) Bilangan pengaturcara yang mencukupi dan bersesuaian dengan masa pembangunan.

d) Kelengkapan tools dan persekitaran pembangunan yang sempurna.

5.5 Pembangunan Pangkalan Data [F4.1]

Keterangan

Pembangunan Pangkalan Data merupakan proses mewujudkan pangkalan data fizikal berdasarkan reka bentuk pangkalan data logikal dan arkitektur pangkalan data. Model data fizikal menggambarkan bagaimana penstoran data dan capaian data dilakukan. Model ini juga turut menggambarkan jadual, medan yang terdapat dalam jadual serta spesifikasi bagi storan secara fizikal yang juga boleh merangkumi pengagihan data dan mekanisma capaian. Ciri-ciri data model fizikal adalah seperti berikut:-

a) Spesifikasi semua jadual dan medan;
b) Kekunci Asing digunakan bagi mengenalpasti hubungan antara jadual;
c) Data model fizikal adalah berbeza mengikut RDBMS.

Pembangunan Pangkalan Data dilaksanakan oleh Pentadbir Pangkalan Data atau lebih dikenali sebagai *Database Administrator* (DBA). Terdapat beberapa bahasa yang boleh digunakan untuk membangunkan pangkalan data, walau bagaimanapun buku panduan ini hanya mengkhususkan kepada penggunaan *Structured Query Language* (SQL). SQL merupakan bahasa khusus yang digunakan untuk pembangunan pangkalan data dan mengurus data-data yang berada dalam Sistem Pengurusan Pangkalan Data Hubungan (*Relational Database Management System*-RDBMS). SQL terdiri daripada:

a) Bahasa Kawalan Data (*Data Control Language*-DCL) seperti *GRANT, REVOKE*;
b) Bahasa Definisi Data (*Data Definition Language*-DDL) seperti *CREATE, DROP, ALTER, RENAME, TRUNCATE;* dan
c) Bahasa Manipulasi Data (*Data Manipulation Language*-DML) seperti *SELECT, INSERT, UPDATE, DELETE*.

Objektif

- Membangunkan pangkalan data fizikal untuk tujuan pembangunan dan pengujian sistem

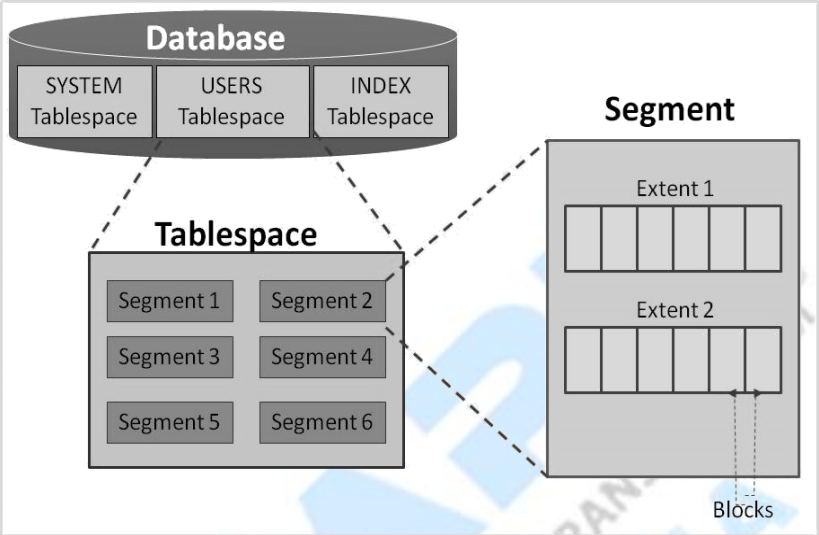
Langkah-Langkah

Langkah 1 : Pemasangan (*Install*) Perisian RDBMS

Bagi membangunkan pangkalan data fizikal, perisian RDBMS yang dipilih perlu dipasang terlebih dahulu. Sekiranya MySQL yang dipilih, perisian tersebut boleh dimuat turun daripada laman web MySQL dan rujuk dokumentasi pemasangan langkah demi langkah daripada laman web yang sama.

Langkah 2 : Peruntukan Ruang Jadual (*Tablespace*)

a) Struktur storan pangkalan data terdiri daripada struktur storan fizikal dan struktur storan logikal. Struktur fizikal terdiri fail-fail seperti *datafiles, redo log files* dan *control files.* Manakala struktur logikal pula terdiri daripada beberapa *tablespace* iaitu ruang sebenar bagi menyimpan beberapa *datafile*.



Rajah 84 : Struktur Storan Secara Logikal Dalam Pangkalan Data

Berdasarkan rajah di atas, t*ablespace* adalah storan logikal yang mengandungi beberapa *segment. Segment* adalah objek pangkalan data yang terdiri daripada jadual-jadual dan *index*. Satu *segment* mengandungi beberapa set *extent* yang diperuntukkan bagi objek pangkalan data secara spesifik seperti jadual. Sebagai contoh, jadual pengguna disimpan dalam data *segment* yang tersendiri manakala *index* bagi jadual pengguna disimpan dalam *index segment* itu sendiri. Namun begitu, semua *extent* yang diperuntukkan dalam satu *segment* adalah disimpan dalam *tablespace* yang sama. Sebagai contoh, satu *segment* disimpan dalam pengguna01.dbf manakala *segment* yang satu lagi adalah dalam pengguna02.dbf. Setiap *extent* terdiri daripada beberapa data *block* yang diperuntukkan untuk menyimpan data yang spesifik. Satu data *block* adalah ruang cakera yang spesifik kepada jumlah bait (*byte)*. Sebagai contoh, satu data *block* adalah ruang cakera fizikal berjumlah 2KB. Peruntukan bagi 24KB dalam satu *extent,* sebanyak 12 data *block* diperlukan. Data *block* adalah unit terkecil dalam storan pangkalan data yang diperuntukkan untuk menyimpan *datafile* secara fizikal.

b) Ruang jadual adalah lokasi storan bagi data sebenar yang terdapat dalam objek pangkalan data. Ruang jadual hanya memperuntukkan lokasi storan bagi pangkalan data. Dengan menggunakan ruang jadual, DBA boleh mengawal bagaimana reka bentuk penstoran dilakukan semasa proses pembangunan pangkalan data. Fungsi penggunaan ruang jadual adalah untuk mengoptimumkan prestasi pangkalan data iaitu seperti pengasingan lokasi atau jenis cakera bagi penyimpanan jenis pengaksesan ke atas data. Iaitu seperti data yang kerap diindeks atau diakses perlu disimpan dalam cakera yang lebih stabil seperti *solid-state drive* (SDD) manakala data yang mengandungi data atau fail arkib disimpan dalam cakera yang biasa seperti *standard hard drive* (HDD).

c) Arahan yang digunakan untuk mencipta ruang jadual bagi Oracle, DB2, Informix, PostgreSQL, MySQL adalah seperti berikut.

```
CREATE TABLESPACE <nama ruang jadual>;
```

Manakala SQL pula arahan adalah seperti berikut.

```
FILEGROUP <nama ruang jadual>;
```

Langkah 3 : Ciptakan Pangkalan Data (*Create A Database*)

a) Pengguna yang mempunyai capaian *root* dibenarkan untuk mencipta pangkalan data. Arahan yang digunakan untuk mencipta pangkalan data ialah:

```
CREATE DATABASE <nama pangkalan data>;
```

Contoh arahan untuk mencipta pangkalan data pdata1 adalah seperti berikut:

```
mysql> CREATE DATABASE pdata1
```

b) Bagi melihat senarai pangkalan data yang telah dicipta, arahan yang digunakan adalah seperti berikut.

```
SHOW DATABASES;
```

Berikut adalah paparan bagi arahan tersebut.

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pdata1             |
+--------------------+
4 rows in set (0.01 sec)
```

Paparan di atas menunjukkan bahawa terdapat empat (4) pangkalan data yang terdapat dalam hos pelayan (*server host*).

Langkah 4 : Wujudkan Jadual (*Create Table*)

a) Jadual diwujudkan setelah pangkalan data siap dibangunkan. Jadual adalah terdiri daripada baris dan lajur yang mengandungi rekod maklumat dalam pangkalan data. Dalam reka bentuk pangkalan data logikal (sila rujuk **Reka Bentuk Pangkalan Data Logikal [F3.3]**) Entiti adalah merujuk kepada jadual (*table*) bagi pangkalan data fizikal. Manakala atribut pula adalah medan (*field*). Jadual berikut adalah pemadanan secara teknologi antara reka bentuk pangkalan data logikal dan fizikal.

Jadual 63 : Pemadanan Istilah antara reka bentuk logikal dan fizikal pangkalan data

Istilah dalam reka bentuk logikal	Istilah dalam reka bentuk fizikal
Entiti	Jadual (*table*)
Atribut	Medan (*column* atau *field*)
Primary UID	*Primary Key*
Secondary UID	*Unique Key*
Relationship	*Foreign Key*

b) Arahan bagi mencipta jadual adalah seperti berikut.

```
CREATE TABLE <nama jadual> (<nama medan> <jenis medan> NOT NULL
PRIMARY KEY <AUTO_INCREMENT sekiranya jenis medan adalah INT>,
<nama medan> <jenis medan>;
```

i) Untuk mengelakkan data tidak diisi ciri-ciri *NOT NULL* diberikan kepada medan tersebut.

ii) Ciri medan *AUTO_INCREMENT* adalah bagi menambahkan satu nilai seterusnya dalam medan dengan mengambil kira bahawa jenis medan adalah *INT*

iii) Medan yang ditakrif sebagai *PRIMARY KEY* adalah menunjukkan bahawa medan tersebut adalah kunci utama dalam jadual. *PRIMARY KEY* boleh ditakrifkan kepada satu atau lebih medan.

Contoh arahan bagi mencipta jadual aset adalah seperti berikut.

```
CREATE TABLE aset (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
nama VARCHAR(20),
catatan VARCHAR(50);
```

Langkah 5 : Wujudkan *VIEW* (*Create VIEW*)

a) *VIEW* adalah merujuk kepada jadual maya yang dihasilkan melalui arahan *SQL*. Jadual adalah terdiri daripada baris dan lajur yang mengandungi rekod maklumat berdasarkan arahan *SQL* yang telah dilaksanakan.

b) Dalam satu pangkalan data, *view* dan jadual berkongsi ruang jadual. Namun begitu, *view* dan jadual tidak boleh mempunyai nama yang sama.

c) Beberapa fungsi *SQL* boleh dimasukkan ke dalam arahan seperti *WHERE* dan arahan *JOIN* daripada beberapa jadual lain kepada jadual maya yang mana fungsi arahan tersebut dipaparkan sebagai satu jadual. Arahan bagi mencipta *view* adalah seperti berikut.

```
CREATE VIEW <nama_view> AS
SELECT <medan1>, <medan2>, ...
```

```
FROM <nama_jadual>
WHERE condition;
```

Contoh arahan bagi mencipta view bagi tempahan bilik oleh pengguna adalah seperti berikut.

```
CREATE VIEW penggunaTempahBilik AS
SELECT pengguna.nama, bilik_mesy.namabilik, bilik_mesy.lokasi
FROM pengguna, bilik_mesy
WHERE pengguna.nokp == bilik_mesy.nokp;
```

Langkah 6 : Wujudkan *INDEX* (*Create INDEX*)

a) Kebiasaannya data disimpan tidak mengikut urutan. Data baru yang dimasukkan tidak disimpan mengikut susunan berdasarkan data yang dimasukkan terdahulu. Oleh yang demikian, tempoh untuk menemui data berdasarkan arahan adalah kurang pantas berbanding kemasukkan data. Dengan itu, index perlu dilakukan bagi membolehkan data ditemui dengan lebih cepat.

b) Arahan *INDEX* digunakan untuk medan-medan tertentu dalam sesuatu jadual bagi mempercepatkan carian data. Lokasi sesuatu data lebih pantas ditemui berbanding carian satu-persatu baris yang terdapat dalam jadual jika tidak menggunakan *INDEX*. Arahan bagi mencipta *index* adalah seperti berikut.

```
create index <nama index> on <nama jadual> (<nama medan>);
```

Contoh arahan bagi mencipta indeks no kad pengenalan pengguna adalah seperti berikut.

```
create index nokp_idx on pengguna(nokp);
```

Langkah 7 : Memuat Masuk (*Load*) Data Ke Dalam Pangkalan Data

Sekiranya terdapat data daripada pangkalan data lama, data tersebut boleh dimuat masuk ke dalam pangkalan data yang baru dibina. Terdapat dua cara untuk memuat masuk data yang sedia ada ke dalam pangkalan data iaitu:

a) Menggunakan Penyataan INSERT

Penyataan ini adalah untuk memasukkan rekod ke dalam jadual. Sintaks bagi penyataan INSERT adalah seperti berikut.

```
INSERT INTO <nama jadual> (<nama medan>)
VALUES (data1),
       (data2);
```

Contoh penyataan untuk memasukkan rekod ke dalam jadual adalah seperti berikut.

```
INSERT INTO pengguna (nama, emel, nombor_telefon, alamat,
BAHAGIAN_id)
VALUES ('Sanem Can Divit', 'sanem@erkenci.com.my', '03-88723038',
'No.1, Jalan Albatross, 62300 Putrajaya', 'BPI');
```

b) Menggunakan Penyataan *LOAD DATA*

Penyataan *LOAD DATA* membolehkan data banyak yang terdapat dalam fail teks dimasukkan ke dalam pangkalan dengan data menggunakan satu arahan sahaja. Sintaks bagi penyataan *LOAD DATA* adalah seperti berikut.

```
LOAD DATA INFILE <nama fail teks.txt> INTO TABLE <nama
pangkalandata.nama jadual>;
```

Contoh penyataan untuk memasukkan rekod ke dalam jadual adalah seperti berikut.

```
LOAD DATA INFILE pengguna.txt INTO TABLE dbtempahan.pengguna;
```

Langkah 8 : Wujudkan Pengguna Dan Kawalan Capaian

a) Pengguna diwujudkan untuk mengakses sesuatu pangkalan data. Dengan itu, pengguna perlu mempunyai capaian *root* untuk melaksanakan aktiviti-aktiviti pembangunan pangkalan data dan juga mencipta pengguna. Berikut adalah arahan yang digunakan untuk mencipta pengguna.

```
CREATE USER '<nama pengguna> '@'<nama hos>' IDENTIFIED BY '<kata laluan>';
```

Contoh arahan untuk mencipta pengguna nama1 adalah seperti berikut:

```
mysql> CREATE USER 'nama1'@'localhost' IDENTIFIED BY
'katalaluan';
```

b) Pentadbir pangkalan data (*Database Administrator* – DBA) boleh memberi kebenaran pengguna untuk mencapai beberapa akses melalui arahan *GRANT* seperti berikut:

```
GRANT ALL PRIVILEGES ON database.table TO 'user'@'localhost';
```

Arahan di atas memberikan semua kawalan akses '*GRANT ALL PRIVILEGES'* bagi semua pangkalan data dan jadual '*database.table'* kepada pengguna '*user*''.

c) Berikut adalah senarai arahan kebenaran yang digunakan mengikut kesesuaian capaian.

i) *ALL PRIVILEGES* – membenarkan semua aktiviti

ii) *CREATE* – membenarkan pengguna mencipta pangkalan data dan jadual

iii) *DROP* – membenarkan pengguna menghapus pangkalan data dan jadual

iv) *DELETE* – membenarkan pengguna menghapuskan baris rekod dalam jadual

v) *INSERT* – membenarkan pengguna menambah baris rekod dalam jadual

vi) *SELECT* – membenarkan pengguna membaca rekod dalam pangkalan data

vii) *UPDATE* – membenarkan pengguna mengemaskini rekod dalam jadual

d) Pentadbir pangkalan data (*Database Administrator* – DBA) boleh memberi menarik semula kebenaran pengguna melalui arahan REVOKE seperti berikut:

```
REVOKE ALL PRIVILEGES ON database.table TO 'user'@'localhost';
```

Arahan di atas menarik balik semua kebenaran kawalan akses '*REVOKE ALL PRIVILEGES*' bagi semua pangkalan data dan jadual '*database.table'* kepada pengguna *'user"*.

Langkah 9 : Dokumenkan Pangkalan Data

Dokumenkan maklumat pangkalan data fizikal yang dibangunkan ke dalam **D09 Dokumen Pangkalan Data**. Dokumentasi mengikut susunan berikut:

a) Ringkasan maklumat pangkalan data fizikal.

b) Skrip yang mengandungi arahan-arahan *SQL.*

Rujukan

1. Oracle Database SQL Language Reference, 11g Release 2 (11.2).
2. MySQL 5.7 Reference Manual.

5.6 Pengaturcaraan Aplikasi [F4.2]

Keterangan

Pengaturcaraan aplikasi adalah proses untuk menulis kod aturcara bagi menghasilkan satu sistem aplikasi. Kod ini menentukan tindakan yang perlu dilaksanakan oleh sistem. Ia ditulis oleh pengaturcara program menggunakan bahasa pengaturcaraan tertentu. Terdapat pelbagai bahasa pengaturcaraan dan *tools* yang boleh digunakan.

Fasa ini sangat penting kerana ia mempengaruhi fasa seterusnya iaitu fasa pengujian dan penyelenggaraan. Satu kod yang ditulis dengan baik mampu mengurangkan kerja-kerja pada kedua-dua fasa tersebut. Sehubungan itu, tumpuan harus diberikan semasa fasa pembangunan ini supaya dapat menghasilkan kod aturcara yang berkualiti dan memenuhi keperluan pengguna.

Terdapat beberapa teknik dalam pengaturcaraan:

a) Teknik Pengaturcaraan Tidak Berstruktur (*Unstructred Programming*)

Kod ditulis tanpa berstruktur, semua arahan ditulis dalam fungsi main().

b) Pengaturcaraan Berstruktur (*Structure Programming*)

Pengaturcaraan berstruktur merupakan pengaturcaraan bertatacara. Kod yang besar dipecahkan kepada kaedah-kaedah pendek (juga dikenali sebagai fungsi atau tatacara) yang lebih kecil agar mudah difahami.

Pengaturcaraan berstruktur biasanya dikaitkan dengan reka bentuk yang menggunakan pendekatan atas-bawah. Dengan pendekatan ini, pengaturcara memetakan struktur yang besar dalam aturcara kepada bentuk operasi kecil, seterusnya melaksanakannya dan menguji operasi-operasi kecil tersebut, dan akhirnya menggabungkan kepada keseluruhan aturcara.

c) Pengaturcaraan Berorientasikan Objek (O*bject-Oriented Programming*)

Pengaturcaraan berorientasikan objek merupakan aturcara komputer yang terdiri daripada sekumpulan unit-unit atau objek. Setiap objek berupaya untuk menerima dan menghantar mesej (pesanan) kepada objek lain. Dengan cara ini, mesej dapat ditangani oleh sebahagian daripada kod. Konsep Asas pengaturcaraan berorientasikan objek adalah seperti berikut:

i) Kelas *(class)* — sebuah kelas mentakrifkan ciri-ciri abstrak bagi sesuatu benda. Ini termasuklah sifat-sifat yang ada padanya dan peranannya.

ii) Objek *(object)* — *instance* bagi suatu kelas.

iii) Kaedah *(method)* — kebolehan bagi sebuah objek.

iv) Pewarisan *(inheritance)* — lazimnya sebuah kelas boleh memiliki "subkelas" yang mengkhususkan kelas tersebut. Semua subkelas ini akan "mewarisi" segala sifat yang ada pada kelasnya.

v) Pengkapsulan *(encapsulation)* — mengasingkan pelaksanaan (implementasi) daripada antaramuka.

vi) Pengabstrakan *(abstraction)* — mengurangkan struktur data dan operasi kepada jenis data yang mudah dan hanya mengandungi *properties* yang penting untuk tujuan tertentu.

vii) Polimorfisme *(polymorphism)* — menggunakan nama yang sama untuk memulakan operasi yang berlainan pada objek yang menggunakan jenis data berbeza.

d) Pengaturcaraan Web (*Web Programming*)

Pengaturcaraan web merujuk kepada penulisan, *markup* dan pengekodan yang terlibat dalam pembangunan web, ia termasuk kandungan web, pelanggan web, skrip pelayan dan keselamatan rangkaian. Bahasa pengaturcaraan yang biasa digunakan untuk pengaturcaraan web adalah XML, HTML, JavaScript, Perl, PHP dan lain-lain.

e) *Stored Procedure*

Satu set arahan komputer yang disimpan dalam pangkalan data untuk membuat capaian data dari pangkalan data. *Store procedure* boleh menyimpan logik pengaturcaraan yang pada asalnya dilaksanakan oleh kod aturcara. *Stored procedure* dapat menjimatkan masa dan *memory* di mana proses yang komplek dan memerlukan pelaksanaan beberapa kenyataan SQL dilaksanakan oleh *store procedure*.

Objektif

- Menulis kod dalam struktur yang tersusun dan ringkas supaya mudah untuk dibaca dan difahami, diubahsuai / ditambahbaik dan diselengara.
- Menukarkan dokumen SDS kepada kod aturcara dan membuat ujian unit ke atas kod yang dibangunkan. Seterusnya menghasilkan produk iaitu Sistem Aplikasi yang memenuhi kehendak pengguna.

Langkah-langkah

Langkah 1: Sediakan Keperluan Pra Pembangunan

a) Menyediakan keperluan teknikal seperti pendekatan, teknik, *tools*, bahasa pengaturcaraan, pangkalan data dan lain-lain keperluan teknikal yang berkaitan sebelum aktiviti pembangunan dilaksanakan seperti dalam **Penyediaan Pelan Pembangunan Sistem [F1.1]** - **Langkah 4: Proses Teknikal.**

b) Pasukan pembangunan perlu memahami dengan mendalam Spesifikasi keperluan sistem dan Spesifikasi Reka bentuk Sistem.

Langkah 2 : Pertimbang Dan Laksanakan Amalan Terbaik Dalam Pengaturcaraan

a) Amalan baik dalam pengaturcaraan (*Best Programming Practise*) adalah satu set peraturan tidak rasmi yang dihasilkan daripada pengalaman dan pembelajaran oleh komuniti pembangun sistem dari semasa ke semasa. Setiap bahasa pengaturcaraan mempunyai set peraturan yang tersendiri. Teknik dan amalan pengaturcaan yang baik dapat meningkatkan kualiti dan prestasi perisian yang dihasilkan. Berikut adalah cadangan-cadangan umum praktis pengaturcaraan yang baik:

i) Penggunaan Konvensyen Ulasan (*Commenting Convention*)

Konvensyen ulasan adalah meletakkan ulasan / komen dalam bahasa yang difahami oleh pembaca dalam kod aturcara. Ia menerangkan secara ringkas tentang apa yang kod laksanakan. Komen-komen ini sangat penting supaya pengaturcara dapat menulis perkara-perkara rumit/penting yang telah dilaksanakan dalam kod tersebut. Melalui ulasan / komen, pengaturcara boleh mengetahui kegunaan kod tanpa perlu membaca keseluruhan kod. Berikut adalah contoh maklumat yang direkodkan sebelum sesuatu kod ditulis (format/*syntax* komen ini bergantung kepada Bahasa pengaturcaraan yang digunakan):

- Bahagian atas setiap fail kod sumber (*File Header Comment)*
  - Pengaturcara asal,
  - Tarikh
  - Tujuan
  - Algorithma yang digunakan (merujuk kepada SDS mana)
  - Senarai pengubahsuaian kepada kod aturcara
- Fungsi / Method
  - Purpose of method
  - Argument descriptions
  - Result descriptions
  - Exceptions thrown

Contoh catatan dalam fungsi:

```
/**
 * This method has no use, and it just illustrative. <br>
 * Although it does suggest adding the two parameters together ! <br>
 *
 * @param first The first number to be added
 * @param second The second number to be added
 * @return The sum of the two parameters
 * @throws BadException if something goes very wrong !
 */
public int sumNumbers(int first, int second) throws BadException{
   . . .
```

- *In line*

  Kod yang rumit dan kurang jelas, diletakkan komen pada bahagian atas sebelum kod atau sebaris dengan kod yang ditulis, untuk memberi penerangan mudah berkaitan kod yang ditulis.

- *Classes and Interfaces*
  - Nama Kelas
  - Keterangan berkaitan kelas dan tujuannya
  - Versi Semakan
  - Nama pengaturcara asal
  - Version

  Contoh penggunaan *classes* dan *interfaces*:

```
/**
 * MyClass <br>
 *
 * This class is merely for illustrative purposes. <br>
 *
 * Revision History:<br>
 * 1.1 – Added javadoc headers <br>
 * 1.0 - Original release<br>
 *
 * @author T.D.Bishop
 * @version 1.1, 19/04/2000
 */
public class MyClass {
   . . .
```

- Pembolehubah *(Variables)*

  Ulasan untuk pembolehubah sepatutnya ringkas sahaja, menerangkan secara ringkas apa kegunaannya.

ii) Penggunaan Konvensyen Pemformatan (*Formatting Convention*)

Pemformatan menjadikan kod yang dihasilkan tersusun, seragam, mudah dibaca dan dicetak. Penggunaan format haruslah konsisten sepanjang kod ditulis. Berikut adalah antara format-format yang di syorkan:

- *Indentation dan Layout*
    - Tetapkan saiz yang standard untuk inden. Secara global saiz inden ditetapkan kepada 4 ruang (*4 spaces*) untuk setiap peringkat.
    - Baris kod tidak terlalu panjang, elakkan penggunakan *horizontal scroll*. Pecahkan kod-kod yang panjang (*Break up long lines*) kepada yang lebih pendek dan pastikan ia dipecahkan pada titik yang bersesuaian supaya mudah dibaca dan dicetak – maksimum 100 aksara untuk setiap baris.

- *Bracketing* – selaraskan susunan penggunaan '{' dan '}' supaya selari, atau menggunakan *slanting style*, di mana '{' adalah dihujung kod manakala '}' selari dengan permulaan kod.

  Contoh penggunaan bracketing:

```
class MyExample {
    public void method(){
        . . .
    }
}
```

- Penulisan HTML – tetapkan format untuk *tags* dan *attributes*, seperti menggunakan huruf besar untuk semua *tags* dan huruf kecil untuk *attributes*.
- Penulisan penyataan SQL – gunakan huruf besar untuk kata kunci (SELECT, DELETE, UPDATE, WHERE, ORDER BY) dan huruf kecil elemen pangkalan data seperti nama *tables, columns* dan *views*.
- Susun setiap klausa SQL yang utama pada baris yang berasingan supaya kenyataan ini adalah lebih mudah untuk dibaca dan dipinda.

Penggunaan format haruslah konsisten dalam setiap kod yang ditulis. Penggunaan gaya pengaturcaraan yang bercampur akan menyukarkan proses penyelenggaraan atau penambahbaikan kod.

iii) Penggunaan Konvensyen Penamaan (*Naming Convention*)

Konvensyen penamaan adalah satu set peraturan untuk menentukan pemilihan nama bagi pembolehubah, fungsi, kelas dan lain-lain entiti dalam kod sumber termasuk nama fail dan folder yang terlibat.

- Gunakan nama yang memberi makna, mudah difahami dan sesuai dengan tujuannya. Contoh fungsi Cetak(), melaksanakan fungsi untuk mencetak.

- Penggunaan nama yang baik mampu menggambarkan kandungan/ tujuan entiti tersebut (*Self Describing*) dan mudah dicari.
- Elakkan penggunaan nama-nama yang hanya berbeza pada aksara, contoh cetak dan Cetak tetapi berbeza tujuan.
- Elakkan penggunakan nama yang boleh mengelirukan, contohnya x. Tidak memberi sebarang makna.

Jadual 64 : Contoh Konvensyen Penamaan

Jenis	Konvensyen Penamaan	Contoh
Classes	• UpperCamelCase: bermula dengan huruf besar dan pada permulaan setiap perkataan baru	DaftarAset(); PermohonanAset(); CleverClassName();

Methods	• lowerCamelCase: bermula dengan huruf kecil dan huruf besar pada permulaan setiap perkataan baru	main();
		kiraJumlah();
		aUsefulMethod();

Atrribute / Variables	• lowerCamelCase: bermula dengan huruf kecil dan huruf besar pada permulaan setiap perkataan baru • nama-nama pembolehubah tidak harus bermula dengan garis bawah (_) atau tanda dolar ($) walaupan dibenarkan • nama pembolehubah sepatutnya pendek tetapi memberi makna • penggunaan 1 aksara sebagai pembolehubah perlu dielakkan kecuali pada pembolehubah sementara, untuk kegunaan loop / for yang digunakan sebagai pembolehubah bilangan sahaja. Kebiasaan pembolehubah sementara menggunakan aksara i,j,k dan m untuk int dan c,d dan e untuk char.	Pembolehubah sementara: int i; char c; myVariable; aClassAttribute; float jumlahMarkah;
Constants / Magic Number	• UpperCase • Dipisahkan dengan '_' setiap perkataan	int MAX_PESERTA = 10;

iv) Konvensyen Pengaturcaraan (*Programming Convention*)

Konvensyen pengaturcaraan menjadikan kod mudah dibaca dan membenarkan pasukan memahami kod yang baru dengan lebih cepat.

Kod Sumber

- Kod yang ditulis perlu mudah dan jelas - elakkan penulisan kod yang panjang, contohnya melebihi 20 baris. Amalkan penggunaan pengaturcaraan berstruktur / bertatacara.
- Elakkan penggunaan nilai *hard-coded* / *magic numbers* – kod sumber seharusnya tidak menggunakan *hard-coded* untuk merujuk kepada mana-mana parameter seperti *paths, file, host*, alamat IP, URLs, *ports* dan lain-lain. Penggunaannya dalam kod menyebabkan ia sukar untuk dikenal pasti jika perlu diubah. Sebaiknya wujudkan pembolehubah dan ia dikonfigurasi dengan baik.

```
private final int ST = 2;
private final int E = 3;
private final int S = 1;
```

- Mewujudkan *Program Versioning*
- Mewujudkan fungsi / method yang boleh dikongsi
- Direktori simpanan - Simpan fail dalam *folder* secara teratur untuk mrmudahkan carian
- Security - validation, code hard (SQL injection)
- Menyediakan mesej ralat yang menggambarkan ralat sebenar, agar mudah untuk mengesan punca ralat.
- Elakkan penggunaan penyataan bersarang yang melebihi 3 peringkat (deeply nested control statements).

Pengaturcaraan Pangkalan Data

- Elakkan penggunaan SELECT *, Biasakan menulis dengan jelas kolum-kolum yang dikehendaki.
- Gunakan *stored procedure* untuk menggantikan penyataan SQL dalam kod sumber bagi meningkatkan prestasi capaian data.
- Melakukan *data validation* pada client semasa data entry. Ini dapat mengelakkan capaian ke pangkalan data berulang kali dengan data yang tidak sah.

Langkah 2 : Bangunkan Sistem

a) Menyediakan persekitaran pembangunan (*development environment*).

i) Membina dan memasang perisian, perkakasan (*server*), rangkaian, *tools* dan lain-lain peralatan yang berkaitan.

ii) Menguji persekitaran pembangunan supaya memenuhi keperluan pembangunan.

b) Mewujudkan dokumentasi *coding standard* (Rujuk Dokumentasi Kod Sumber).

c) Memulakan penulisan kod aturcara (Rujuk Amalan Baik dalam Pengaturcaraan).

d) Melaksanakan ujian unit bagi setiap komponen sistem dan mendokumenkan keputusan dalam Laporan Ujian (Keterangan lanjut dalam perenggan **Ujian Sistem [F4.3]**).

Langkah 3 : Menyediakan Dokumentasi Kod Sumber

a) Dokumentasi kod sumber adalah dokumen yang mengandungi senarai kod aturcara yang meliputi struktur direktori dan hierarki fail serta garis panduan yang mengesyorkan perkara berkaitan gaya pengaturcaraan (*style*), konvensyen penamaan, *indentation*, ulasan / komen, pengistiharan pembolehubah, penyataan SQL dan lain-lain yang perlu dipatuhi oleh semua pengaturcara. **Rujuk D10 Dokumen Kod Sumber.**

b) Tujuan pengwujudan dokumen ini adalah untuk mewujudkan habit / kebiasaan yang baik dalam gaya penulisan kod dan mewujudkan keseragaman kepada semua pengaturcara.

c) Kod aturcara yang telah dibangunkan adalah sangat penting untuk didokumentasikan. Ia termasuk fail README yang menyimpan maklumat umum seperti tujuan sistem, URL kepada kod sumber utama (*main source code*) dan lain-lain maklumat asas berkaitan sistem. Fail ini menjadi rujukan pertama kepada pembaca kod.

d) Pengaturcara sangat digalakkan untuk mengikuti garis panduan ini supaya:

   i) meningkatkan kebolehbacaan dan kefahaman ke atas kod sumber mereka.

   ii) memudahkan semakan, penambahbaikan dan penyelenggaraan ke atas perisian.

   iii) proses pemindahan teknologi (*Transfer Of Technology*) kepada ahli pasukan lain / baru menjadi mudah dan lebih cepat.

   iv) memudahkan pasukan pengaturcara membuat carian kepada fungsi-fungsi atau kelas dalam kod yang ditulis, seterusnya menggalakkan penggunaan semua kod sedia ada (*reusability*) dan

   v) sebagai rujukan pada masa depan.

e) Salah satu cabaran dalam mendokumentasikan kod sumber adalah memastikan bahawa ulasan / komen dikemaskini selari dengan kod aturcara yang ditulis. Kekangan masa menyebabkan pengaturcara hanya fokus kepada kod yang tulis dan sangat sedikit masa diberikan kepada penulisan dokumen kod sumber.

f) Menjana dokumen secara automatik (*Documentation Generator*). Terdapat *tool* yang membolehkan dokumen ini dijana secara automatik dari kod sumber yang ditulis. Ia dikenali sebagai *documentation generator.*

i) *Documentation generator* dijana dari kod sumber seperti ulasan (*comment*) yang ditulis oleh pengaturcara, ini menjadikan dokumen mudah dikemaskini dari masa ke masa selaras dengan kod yang ditulis.

ii) Setiap Bahasa pengaturcaraan mempunyai *documentation generator* sendiri. Contoh: Doxygen, NDoc, javadoc, EiffelStudio, Sandcastle, ROBODoc, POD, TwinText, atau Universal Report dan lain-lain.

Rujukan

1. General Software Development Standard and Guidelines Version 3.5 - Science Infusion Software Engineering Process Group (SISEPG).
2. Java Languange Coding Standard - Sun Microsystem.

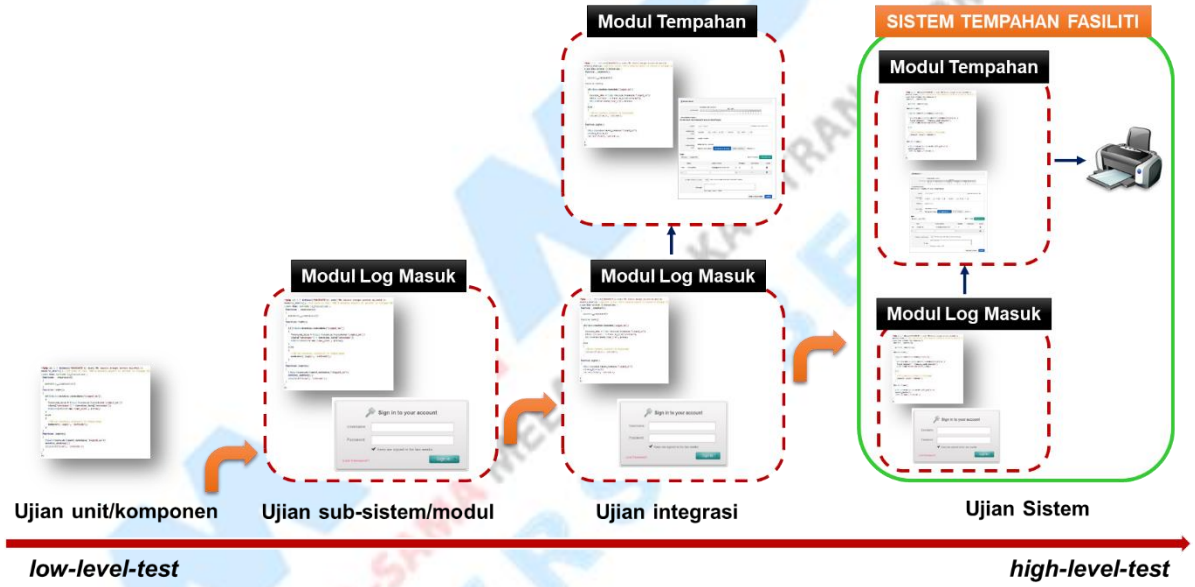
5.7 Pengujian Sistem [F4.3]

Keterangan

Pengujian Sistem merupakan aktiviti verifikasi yang dilakukan terhadap komponen atau sistem (*test object*) untuk memastikan ia dibangunkan berdasarkan kepada **spesifikasi keperluan dan reka bentuk sistem**. Pengujian sistem merangkumi pelbagai peringkat ujian sebelum sistem diuji secara komprehensif di dalam fasa pengujian penerimaan. Semasa pengujian ini dilaksanakan, ralat yang dikesan akan diperbetulkan dan unit/komponen/modul yang berkaitan akan diuji semula sehingga ralat berjaya diperbaiki.

Jenis-jenis pengujian yang dijalankan adalah **pengujian keperluan fungsian**, **pengujian keperluan bukan fungsian (kualiti)** serta **verifikasi terhadap ralat yang telah dibaiki**. Pengujian ini akan **dilaksanakan oleh Pasukan Pembangun Sistem**.

Peringkat-peringkat ujian yang dilaksanakan mengikut turutan adalah seperti rajah di bawah.



Rajah 85 : Peringkat Pengujian Sistem

Terdapat dua jenis pendekatan pengujian yang digunakan semasa melaksanakan pengujian ke atas sistem aplikasi iaitu *big bang testing* dan *incremental testing*. *Big bang testing* merupakan pengujian yang dilaksanakan terhadap sistem aplikasi yang telah lengkap dibangunkan, manakala *incremental testing* merupakan satu bentuk ujian modul di mana modul yang akan diuji digabungkan dengan modul yang sudah diuji.

Pendekatan bagi aktiviti pengujian ini dikenali sebagai *incremental testing* iaitu **pengujian dilakukan secara berperingkat** bermula daripada pengujian unit/komponen terkecil sistem aplikasi seperti fungsi, kelas, prosedur dan antara muka (**Ujian unit/komponen**); diikuti dengan ujian modul (**Ujian sub-sistem/modul**); seterusnya menguji dua atau lebih modul/sistem/elemen perkakasan yang disepadukan (**Ujian integrasi**); dan akhirnya semua modul yang terlibat diuji secara menyeluruh (**Ujian sistem**).

Pendekatan ini menerangkan bahawa **ujian tahap rendah (*low-level-test*)** perlu dilaksanakan terlebih dahulu bertujuan untuk mengesahkan bahawa ujian segmen kod sumber telah dilaksanakan dengan betul sebelum ke peringkat ujian berikutnya iaitu, **ujian tahap tinggi (*high-level test*)** yang bertujuan mengesahkan fungsi-fungsi utama sistem aplikasi.

Bagi setiap peringkat pengujian, terdapat empat (4) elemen yang perlu ditetapkan sebelum pelaksanaan ujian iaitu:

a) *Entry Criteria* boleh merujuk kepada dokumen, status/ aktiviti serta tahap pencapaian atau pengukuran yang menjadi pra-syarat untuk melaksanakan sesuatu peringkat pengujian.

b) Aktor merujuk kepada individu/kumpulan yang terlibat dengan sesuatu ujian.

c) Aktiviti ialah aktiviti yang perlu dijalankan semasa ujian.

d) *Exit Criteria* pula merujuk kepada dokumen, status/aktiviti serta tahap pencapaian atau pengukuran yang menjadi syarat untuk menamatkan sesuatu peringkat pengujian

Objektif

- Menilai kualiti keseluruhan sistem selepas pembangunan bagi memastikan sistem aplikasi yang dibangunkan sedia untuk diuji di peringkat pengujian penerimaan pengguna.

Langkah-langkah

Langkah 1 : Laksana Ujian Unit/Komponen

a) Tetapkan elemen *Entry Criteria*, aktor, aktiviti dan *Exit Criteria* bagi Ujian Unit/ Komponen seperti jadual di bawah:

Jadual 65 : *Entry Criteria* dan *Exit Criteria* bagi Ujian Unit/Komponen

Entry Criteria	i) Dokumen SRS dan SDS telah disahkan ii) Kod sumber bagi unit/komponen telah selesai dibangunkan/ telah diperbaiki bagi tujuan *re-test* iii) Unit/komponen untuk diuji serta *stubs*/*drivers* yang diperlukan telah disediakan dalam persekitaran pembangunan
Aktor	Pasukan Pembangun Sistem
Aktiviti	i) Pasukan Pembangun Sistem melaksanakan ujian unit/komponen ii) Pembetulan kod sumber dilakukan bagi ujian yang gagal. iii) Pengujian semula dilaksanakan bagi ujian yang gagal

Exit Criteria	i) Senarai semak hasil ujian menunjukkan unit/komponen berjaya melaksanakan fungsian atau bukan fungsian seperti yang telah ditetapkan ii) Ralat telah berjaya dibaiki

b) Kenalpasti unit/komponen adalah bahagian terkecil di dalam sistem aplikasi yang boleh diuji (*testable*) seperti fungsi, kelas, prosedur dan antara muka.

c) Sediakan *stubs dan drivers* yang diperlukan di dalam ujian unit/ komponen bagi menggantikan unit/ komponen/ modul yang perlu semasa berinteraksi dengan unit/komponen yang diuji. *Stubs* adalah simulasi (*dummy module*) bagi menggantikan unit/komponen selepas (*subordinate/ lower level*) unit/komponen yang diuji. *Drivers* pula adalah simulasi bagi menggantikan unit/komponen sebelum (*upper level*) unit/komponen yang diuji. Rajah di bawah menunjukkan gambaran penggunaan *stubs* dan *drivers* di dalam ujian unit/komponen.

Rajah 86 : Ujian Komponen - Stubs dan Drivers

d) Laksana ujian unit/komponen bagi memastikan kod program yang dibangunkan memenuhi fungsi unit/komponen dan mengenal pasti ralat di dalam unit/komponen berkenaan. Ujian ini juga hendaklah merangkumi ujian fungsian dan ujian bukan fungsian.

e) Rujuk asas ujian (*test basis*) yang berkaitan iaitu spesifikasi keperluan unit/komponen, dokumen reka bentuk terperinci seperti SRS atau SDS dan kod sumber program semasa melaksanakan ujian bagi memastikan ianya memenuhi keperluan dan reka bentuk unit/ komponen yang telah ditetapkan.

f) Guna kombinasi teknik pengujian yang bersesuaian seperti teknik *white-box testing* dan juga *black box testing*. *White-box testing* merupakan teknik pengujian yang terperinci yang dilakukan terhadap logik dalam dan struktur kod. Teknik ini memerlukan penguji mempunyai pengetahuan penuh tentang kod sumber. Manakala *black box te*sting pula merupakan teknik pengujian berdasarkan kepada spesifikasi sistem aplikasi. Pengujian ini dilaksanakan dengan memasukan *input* dan *output* akan diperiksa sama ada memenuhi fungsi yang telah ditetapkan atau tidak.

g) Uji pengendalian ralat (*error handling*) sekiranya input yang tidak sah diberikan.

h) Baiki ralat yang ditemui dan uji semula unit/komponen berkenaan. Pengujian akan dilaksanakan sehingga unit/ komponen memenuhi *Exit Criteria* yang telah ditetapkan untuk ke peringkat ujian seterusnya iaitu Ujian Sub-Sistem/Modul.

Langkah 2 : Laksana Ujian Sub-Sistem/Modul

a) Tetapkan elemen *Entry Criteria*, aktor, aktiviti dan *Exit Criteria* bagi Ujian Sub-Sistem/Modul seperti jadual di bawah:

Jadual 66 : *Entry Criteria* dan *Exit Criteria* bagi Ujian Sub-Sistem/Modul

Entry Criteria	i) Ujian unit/komponen telah dilaksanakan dengan sempurna ii) *Defects/bugs* yang dilaporkan dalam ujian unit/komponen telah diperbaiki dan diuji semula iii) *Test script* dan *test data* bagi ujian sub-sistem/modul telah disediakan oleh Pasukan Pembangun Sistem
Aktor	Pasukan Pembangun Sistem
Aktiviti	i) Pasukan Pembangun Sistem melaksanakan ujian sub-sistem/modul ii) Pembetulan dilakukan bagi ujian sub-sistem/modul yang gagal iii) Pengujian semula dilaksanakan bagi ujian yang gagal
Exit Criteria	i) *Test script* telah diuji ii) Senarai semak hasil ujian menunjukkan sub-sistem/modul berjaya melaksanakan fungsian atau bukan fungsian seperti yang telah ditetapkan iii) Ralat telah berjaya dibaiki

b) Sediakan *test data*, bangunkan *test script* berdasarkan pertimbangan di bawah bagi memastikan ujian sub-sistem/modul dilaksanakan dengan berkesan:

i) *Module Interface Test*: bertujuan untuk menguji maklumat yang masuk dan keluar daripada modul;

ii) *Local data structures*: Struktur data tempatan diperiksa untuk memastikan data yang disimpan secara sementara dapat mengekalkan integritinya semasa pelaksanaan algoritma.

iii) *Boundary conditions*: bertujuan untuk memastikan modul beroperasi dengan betul di sempadan yang ditetapkan (*to limit or restrict processing*).

iv) *Independent paths:* bertujuan untuk menguji semua *independent paths* yang melalui struktur kawalan bagi memastikan bahawa semua kenyataan dalam modul telah dilaksanakan sekurang-kurangnya sekali.

v) *Error handling paths*: untuk memastikan ralat ditangani dengan betul dan *error handling paths* yang dikenalpasti dapat digunakan selepas melepasi beberapa siri ujian.

c) Laksanakan ujian sub-sistem/modul untuk menguji interaksi di antara unit/komponen dalam modul bagi memastikan ianya dapat berfungsi secara kolektif bagi fungsi, kelas, prosedur dan antara muka yang terlibat di dalam sesebuah modul.

d) Uji setiap modul yang terdapat dalam sistem aplikasi secara berasingan sebelum ianya diuji secara bersepadu di peringkat ujian integrasi.

Langkah 3 : Laksana Ujian Integrasi

a) Tetapkan elemen *Entry Criteria*, aktor, aktiviti dan *Exit Criteria* bagi Ujian Integrasi seperti jadual di bawah:

Jadual 67 : *Entry Criteria* dan *Exit Criteria* bag Ujian Integrasi

Entry Criteria	i) Ujian sub-sistem/modul telah dilaksanakan dengan sempurna ii) *Defects/bugs* yang dilaporkan dalam ujian sub-sistem/modul telah diperbaiki dan diuji semula iii) Sistem/antara muka sistem luar telah sedia diintegrasi dengan sistem yang diuji (*SUT*). iv) *Test script* dan *test data* bagi ujian integrasi telah disediakan oleh Pasukan Pembangun Sistem v) Penguji Integrasi telah diberikan penerangan
Aktor	i) Pasukan Pembangun Sistem ii) Penguji Integrasi
Aktiviti	i) Penguji integrasi melaksanakan ujian integrasi ii) Penguji integrasi merekodkan hasil ujian iii) Pembetulan dilakukan bagi ujian integrasi yang gagal iv) Pengujian semula dilaksanakan bagi ujian yang gagal
Exit Criteria	i) *Test script* telah diuji ii) Senarai semak hasil ujian ujian menunjukkan integrasi berjaya melaksanakan fungsian atau bukan fungsian seperti yang telah ditetapkan iii) Ralat telah berjaya dibaiki

b) Sediakan *test data*, bangunkan *test script* bagi memastikan ujian sub-sistem/modul dilaksanakan dengan berkesan. Objek yang akan dinilai semasa ujian integrasi adalah sub-sistem, pangkalan data, infrastruktur, antara muka serta konfigurasi sistem. Bagi tujuan ini, Pasukan Pembangun Sistem hendaklah mempunyai kefahaman yang jelas terhadap reka bentuk dan keperluan integrasi sebelum ujian ini dilaksanakan.

c) Laksanakan ujian integrasi yang merangkumi ujian fungsian dan ujian bukan fungsian bagi menguji perkara berikut:

i) interaksi di antara sub-sistem/modul dalam sistem aplikasi;

ii) interaksi dalaman di antara sistem operasi, fail sistem dan API perkakasan sistem; DAN

iii) integrasi di antara sistem yang dibangunkan dengan sistem luaran/antara muka luaran (sekiranya ada).

d) Rujuk asas ujian (test basis) yang berkaitan iaitu reka bentuk dan arkitektur sistem serta dokumen **D02 Spesifikasi Keperluan Bisnes**.

e) Laksanakan ujian integrasi dalam beberapa sesi mengikut keperluan integrasi yang wujud dalam sistem aplikasi sehingga memenuhi *Exit Criteria* yang telah ditetapkan untuk ke peringkat ujian seterusnya iaitu Ujian Sistem.

Langkah 4 : Laksana Ujian Sistem

a) Tetapkan elemen *Entry Criteria*, aktor, aktiviti dan *Exit Criteria* bagi Ujian Sistem seperti jadual di bawah:

Jadual 68 : Entry Criteria dan Exit Criteria bag Ujian Sistem

Entry Criteria	i) Ujian integrasi telah dilaksanakan dengan sempurna ii) *Defects/bugs* yang dilaporkan dalam ujian integrasi telah diperbaiki. Tiada lagi *defects/bugs* bagi: • *Priority High* atau *Medium;* dan • *Severity Blocking, Critical*, dan *Major* iii) *SUT telah disediakan di dalam persekitaran pengujian.* iv) *Test script* dan *test data* bagi ujian sistem telah disediakan oleh Pasukan Pembangun Sistem
Aktor	Pasukan Pembangun Sistem
Aktiviti	i) Pasukan Pembangun Sistem melaksanakan ujian sistem ii) Pasukan Pembangun Sistem merekodkan hasil ujian

	iii) Pembetulan dilakukan bagi ujian yang gagal iv) Pengujian semula dilaksanakan bagi ujian yang gagal
Exit Criteria	i) *Test script* telah diuji ii) Hasil ujian menunjukkan sistem berjaya melaksanakan fungsian atau bukan fungsian seperti yang telah ditetapkan. iii) Sistem telah berjaya memenuhi keperluan bisnes dan keperluan fungsian iv) Ralat telah berjaya dibaiki

b) Sediakan *test data*, bangunkan *test script* bagi memastikan ujian sistem dilaksanakan dengan berkesan.

c) Laksanakan ujian sistem untuk menentukan keseluruhan sistem aplikasi telah memenuhi spesifikasi yang ditetapkan sebelum ke fasa pengujian penerimaan. Walaubagaimanapun, ujian ini sebaik-baiknya dilaksanakan di dalam persekitaran yang hampir sama dengan persekitaran sebenar (*staging/pre-production).*

d) Rujuk dokumen spesifikasi keperluan, proses bisnes, *Use Case*, laporan analisis risiko, interaksi sistem dengan sistem operasi dan *system resources* semasa melaksanakan ujian sistem.

e) Laksanakan ujian sistem sehingga memenuhi *Exit Criteria* yang telah ditetapkan.

Langkah 5 : Sediakan Laporan Ujian Sistem

a) Sediakan Laporan Ujian Sistem sebagai pengesahan aktiviti Pengujian Sistem telah dilaksanakan sepenuhnya. Laporan Ujian Sistem menentukan tahap kesediaan sistem dan merupakan *Entry Criteria* kepada Ujian Penerimaan Pengguna. Format Laporan Ujian Sistem adalah seperti D11 Laporan Ujian Sistem.

b) Laporan Ujian Sistem hendaklah **mempunyai sekurang-kurangnya elemen** berikut:

Jadual 69 : Format Laporan Ujian Sistem

Bil.	Item	Keterangan
1	**Pengenalan**	Penerangan berkaitan tujuan pelaporan dan skop pelaporan. Skop pelaporan boleh berdasarkan sistem atau modul (sekiranya sistem adalah berskala besar dan mempunyai kompleksiti yang tinggi).

<table>
<tr>
<td>2</td>
<td>Aktiviti Pengujian</td>
<td>
Serahan
Menerangkan bilangan modul/submodul yang terlibat, status lulus dan gagal, serta keterangan yang berkaitan dengannya.

<table>
<tr><th>Bil</th><th>Modul & Sub Modul</th><th>Status Pengujian (Lulus/Gagal/KIV)</th><th>Keterangan</th></tr>
<tr><td rowspan="3">1</td><td>Modul Personel</td><td>Lulus</td><td>Selesai</td></tr>
<tr><td>Modul Gaji</td><td>KIV</td><td>2 modul masih KIV untuk pengujian</td></tr>
<tr><td>Modul Pelaporan</td><td>Lulus</td><td>Selesai</td></tr>
</table>

Rumusan Serahan
Ringkasan kepada aktiviti pengujian sistem yang telah dilakukan berdasarkan sistem atau modul yang diuji.

Sekiranya rumusan adalah diperingkat sistem, maka bilangan yang digunakan adalah bilangan modul. Bagi peringkat modul, bilangan yang digunakan adalah bilangan submodul.

Contoh:

<table>
<tr><th>Bil</th><th>Modul</th><th>Bil Lulus</th><th>Bil Gagal/KIV</th><th>Keterangan</th></tr>
<tr><td>1</td><td>Modul Personel</td><td>5</td><td>0</td><td>Selesai</td></tr>
<tr><td>2</td><td>Modul Gaji</td><td>5</td><td>2</td><td>2 submodul KIV kerana terdapat isu compliance</td></tr>
<tr><td>3</td><td>Modul Pelaporan</td><td>5</td><td>1</td><td>Submodul Business Intelligence tidak diuji untuk unstructured data</td></tr>
<tr><td>4</td><td>Jumlah</td><td>15</td><td>3</td><td></td></tr>
<tr><td>5</td><td>Peratus Lulus/Gagal/KIV</td><td>83.33</td><td>16.67</td><td></td></tr>
</table>
</td>
</tr>
<tr>
<td>3</td>
<td>Dokumen Sokongan</td>
<td>Menyatakan dokumen sokongan yang dirujuk berkaitan dengan Laporan Ujian Sistem yang boleh digunakan oleh pemilik sistem untuk mengesahkan pelaksanaan Ujian Sistem.

Contoh:
Senarai nama penguji, modul/sub-modul yang terlibat dan tarikh selesai pengujian.</td>
</tr>
</table>

Rujukan

1. Software Engineering: A Practitioner's Approach Eighth Edition, Roger S. Pressman, Ph.D, Bruce R. Maxim, Ph.D, Mc Graw Hill Education.
2. Software Quality Assurance From theory to implementation, G. Daniel.
3. Certified Tester Foundation Level Syllabus Version 2018.
4. Certified Tester Advanced Level Syllabus Test Analyst Version 2012.