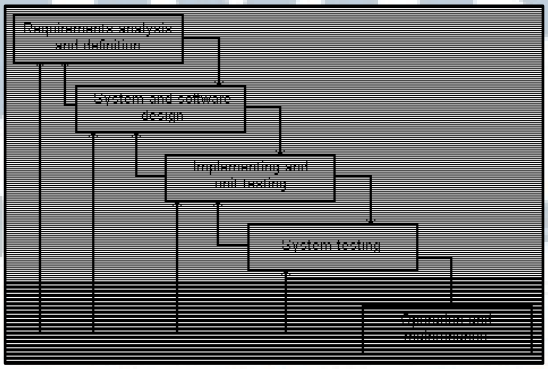
BAB II

TELAAH LITERATUR

2.1. Waterfall Model Process Software Engineering

Nama model ini sebenarnya adalah "*Linear Sequential Model*". Model ini sering disebut dengan "*classic life cycle*" atau model *waterfall*. Model ini adalah model yang muncul pertama kali yaitu sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai didalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan urut mulai dari level kebutuhan sistem lalu menuju ke tahap analisis, desain, *coding*, *testing / verification*, dan *maintenance*. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Sebagai contoh tahap desain harus menunggu selesainya tahap sebelumnya yaitu tahap *requirement*. Secara umum tahapan pada model *waterfall* dapat dilihat pada gambar berikut (Pressman, 1997).



Gambar 2.1 Waterfall Model Process(Pressman, 1997)

Gambar 2.1 adalah tahapan umum dari model proses ini. Akan tetapi Roger S. Pressman memecah model ini menjadi 6 tahapan meskipun secara garis besar sama dengan tahapan-tahapan model *waterfall* pada umumnya. Berikut adalah penjelasan dari tahap-tahap yang dilakukan di dalam model ini (Pressman, 1997).

a. *System / Information Engineering and Modeling*. Permodelan ini diawali dengan mencari kebutuhan dari keseluruhan sistem yang akan diaplikasikan ke dalam bentuk *software*. Hal ini sangat penting, mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware, database*, dsb. Tahap ini sering disebut dengan *project definition*.

b. *Software Requirements Analysis*. Proses pencarian kebutuhan diintensifkan dan difokuskan pada *software*. Untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang domain informasi dari *software*, misalnya fungsi yang dibutuhkan, *user interface*, dsb. Dari 2 aktivitas tersebut (pencarian kebutuhan sistem dan *software*) harus didokumentasikan dan ditunjukkan kepada pelanggan.

c. *Design*. Proses ini digunakan untuk mengubah kebutuhan-kebutuhan diatas menjadi representasi ke dalam bentuk "*blueprint*" *software* sebelum *coding*dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti 2 aktivitas sebelumnya, maka proses ini juga harus didokumentasikan sebagai konfigurasi dari *software*.

d. *Coding*. Untuk dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu ke dalam bahasa pemrograman melalui proses *coding*. Tahap ini merupakan implementasi dari tahap *design* yang secara teknis nantinya dikerjakan oleh programmer.

e. *Testing / Verification*. Sesuatu yang dibuat haruslah diujicobakan. Demikian juga dengan *software*. Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari *error*, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

f. *Maintenance*. Pemeliharaan suatu *software* diperlukan, termasuk di dalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada *error* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut. Pengembangan diperlukan ketika adanya perubahan dari eksternal perusahaan seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

2.2. Perancangan Sistem

2.2.1. Unified Modeling Language (UML)

Unified Modeling Language adalah sebuah bahasa yang diterima dan digunakan oleh *software developer* dan *software analyst* sebagai suatu bahasa yang cocok untuk merepresentasikan grafik dari suatu relasi antar entitas-entitas

software(Gornik, 2003). Dengan menggunakan UML, tim pengembangan *software* akan mempunyai banyak keuntungan, yaitu memudahkan komunikasi dengan sesama anggota tim tentang *software* seperti apa yang dibuat, memudahkan integrasi ke dalam area pengerjaan *software* karena bahasa ini berbasiskan *meta-models* dimana *meta-models* bisa mendefinisikan proses-proses untuk mengkonstruksikan konsep-konsep yang ada. UML juga menggunakan formatur input dan output yang sudah mempunyai bentuk standar yaitu XML *Metadata Interchange* (XMI), menggunakan aplikasi dan pemodelan data yang universal, merepresentasikan dari tahap analisis ke implementasi lalu ke *deployment* yang terpadu, dan mendeskripsikan keutuhan tentang spesifikasi *software* (Hofmeister, Nord, & Soni, 1999).

2.2.1.1. Use Case Model

Use case model adalah teknik pemodelan untuk mendapatkan *functional requirement* dari sebuah sistem, menggambarkan interaksi antara pengguna dan sistem, menjelaskan secara naratif bagaimana sistem akan digunakan, menggunakan skenario untuk menjelaskan setiap aktivitas yang mungkin terjadi. Ada beberapa bagian di dalam *use case model* (Dharwiyanti & Wahono, 2003).

a. *Use Case*, untuk mengetahui *action* atau prosedur apa yang ada di dalam sistem.
b. *Actor*, siapa saja yang terlbibat dalam *action* tersebut.
c. *Relationship*, bagaimana *actions* saling berelasi satu sama lain di dalam sistem.

2.2.1.2. Class Diagram

Diagram yang paling umum dijumpai pada pemodelan berbasis UML. Di dalam *class diagram* terdapat class dan *interface* beserta atribut dan operasinya, relasi yang terjadi antar objek, *constraint* terhadap objek-objek yang saling berhubungan dan inheritance untuk organisasi *class* yang lebih baik. *Class diagram* juga terdapat *static view* dari elemen pembangun sistem. Pada intinya *class diagram* mampu membantu proses pembuatan sistem dengan memanfaatkan konsep forward ataupun reverse engineering (Rational Software Corporation, 1997).

Class diagram mempunyai 2 komponen penting.

a. *Structural*, yaitu ciri pembeda objek.

b. *Behavioral*, yaitu tingkah laku atau kegiatan yang mampu dilakukan objek.

2.2.1.3. Sequence Diagram

Menjelaskan interaksi obyek-obyek yang saling berkolaborasi (berhubungan), mirip dengan *activity diagram* yaitu menggambarkan alur kejadian sebuah aktivitas tetapi lebih detil dalam menggambarkan aliran data termasuk data atau *behaviour* yang dikirimkan atau diterima namun kurang mampu menjelaskan detil dari sebuah algoritma (Hofmeister, Nord, & Soni, 1999).

Dalam *sequence diagram* terdapat beberapa bagian.

a. *Participant*, yaitu objek yang terkait dengan sebuah urutan proses.

b. *Lifeline*, menggambarkan daur hidup sebuah objek.

c. *Activation*, suatu titik waktu dimana sebuah objek mulai berpartisipasi dalam sebuah sequence.

d. *Time*, elemen paling penting dalam sequence diagram yang konteksnya adalah urutan, bukan durasi.

e. *Return*, suatu hasil kembalian sebuah operasi. Operasi mengembalikan hasil tetapi boleh tidak ditulis jika tidak ada perbedaan dengan Getter-nya.

2.2.1.4. Activity Diagram

Teknik untuk menjelaskan *business process*, *procedural logic*, dan *work flow*. Bisa dipakai untuk menjelaskan teks *use case* dalam notasi grafis dengan menggunakan notasi yang mirip *flow chart*, meskipun terdapat sedikit perbedaan notasi (Hofmeister, Nord, & Soni, 1999).

a. *Nodes*, menandakan initial dan final node, final node boleh lebih dari 1.

b. *Activity*, aktivitas sistem dapat berupa aktivitas fisik juga bagi *user*.

c. *Flow/edge*, arah sebuah proses.

d. *Fork*, awal sebuah proses paralel.

e. *Join* akhir proses paralel.

f. *Condition*, kondisi yang dituliskan dalam bentuk teks

g. *Decision*, implementasi if dan then.

h. *Merge*, penyatuan beberapa flow.

i. *Partition*, siapa tau apa yang menjalankan aktivitas.

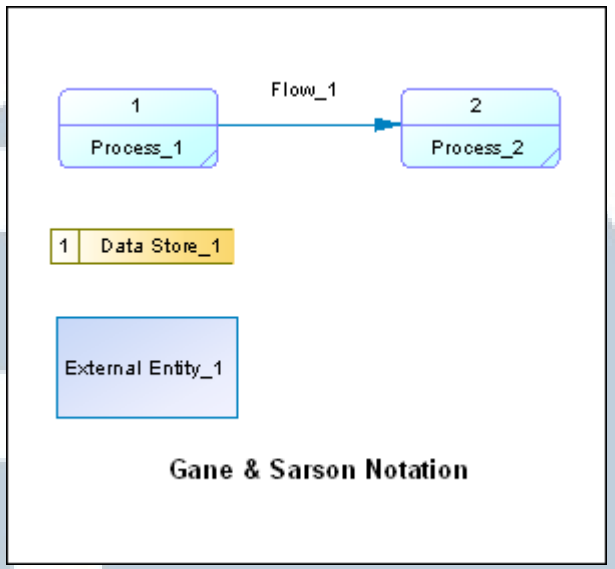
2.2.1.5. Deployment Diagram

Menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Digunakan untuk memodelkan sistem tambahan (*embedded system*) seperti *device*, *node* dan *hardware*, sistem *client/server*, sistem terdistribusi, dan rekayasa ulang aplikasi (Dharwiyanti & Wahono, 2003).

2.2.2. Data Flow Diagram

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama *bubble chart*, *bubble diagram*, model proses, diagram alur kerja, atau model fungsi (Jogiyanto, 1990).

DFD ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem. DFD ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program (Burch, 1992).



Gambar 2.2 Notasi DFD versi Gane dan Sarson

2.3. Push Technology

Push technology merupakan sarana yang relatif baru untuk mengotomatisasi pengiriman berita dan informasi ke perangkat telpon genggam melalui Internet. Metode *push* berbeda dari *e-mail* dalam pendekatannya (data *real-time deliveryversus* sebuah metodologi "simpan dan teruskan" digunakan untuk *e-mail* yang menyebabkan penundaan penyampaian informasi langsung ke penerima). Pendekatan *push* mengharuskan pengembang mengimplementasikan dua aplikasi yaitu aplikasi bagi penyedia informasi/*publisher* sebagai *serverapplication* dan aplikasi bagi penerima informasi sebagai *client application*(Leung & Liangjie, 2003).

Push technology berkembang dari ide yang sederhana. Informasi yang dibutuhkan *user* dapat langsung dikirim kepada daripada *user* dari suatu sistem informasi harus melakukan *request* ke dalam sistem. Keuntungannya adalah otomasi

pengiriman langsung ke dalam perangkat penerima yang dimiliki *user*. Pada metode tradisional, *user* dihadapkan pada masalah kapan dan dimana mereka saat mereka ingin mengetahui suatu data atau informasi. Sebelum adanya *Push technology*, *user* harus melakukan peninjauan dan pengecekan aktif ke suatu situs atau bentuk sistem informasi lain dan *push technology* dapat mengatasi masalah ini (Umbach, 1997).

2.3.1. BlackBerry Push Service

BlackBerry® *push service* adalah bagian penting dari pengalaman Blackberry *smartphone* yang *real time* dan selalu siap digunakan. Layanan ini menawarkan cara yang sangat efisien dan handal untuk mengirim informasi kepada pengguna. Aplikasi juga dapat memproses informasi di latar belakang dan memberi notifikasi kepada pengguna(Research In Motion, 2011).

Blackberry *push service* digunakan untuk menambahkan karakteristik yang disukai pelanggan *smartphone* ke aplikasi , termasuk masa pakai baterai yang lebih lama, aplikasi yang berjalan di latar belakang, dan kemampuan memperoleh informasi dengan cepat(Research In Motion, 2011).

Blackberry *push service* tersedia untuk semua pengembang dan penyedia konten *web*. Dengan teknologi *push*, dapat membuat aplikasi yang lebih menarik.

Guna membangun aplikasi Blackberry berbasis *push service* diperlukan langkah-langkah sebagai berikut.

1. Pelajari cara kerja BlackBerry *push service*.

2. Lihat Panduan Pengembangan BlackBerry *Push Service SDK* untuk informasi tambahan.
3. *Download* Blackberry *Push Service SDK v1.0* (versi Windows atau versi Linux) untuk mengembangkan komponen sisi *server*. Gunakan program pengembangan aplikasi Java atau BlackBerry*Widget SDK* untuk mengembangkan bagian klien dari Blackberry *push service*.
4. Daftar untuk mengevaluasi Blackberry *push service*. Pendaftaran ini memungkinkan menguji aplikasi atau layanan di lingkungan pengujian.
5. Daftar untuk meluncurkan aplikasi saat masih dalam proses pembuatan.

Catatan: Jikamenggunakan Blackberry *push service* di aplikasi yang berkomunikasi secara eksklusif melalui BlackBerry® *Enterprise Server*, pendaftaran tidak diperlukan.

2.3.2. BlackBerry Push Application Programming Interface

Menggunakan Blackberry *Push Application Programming Interface* (API), vendor *software* yang terdaftar dalam *Alliance Program* ataupun developer yang terdaftar untuk memakai Blackberry *push service* dapat membuat jangkauan aplikasi yang lebih luas untuk kegunaan BlackBerry bagi pemakai *software* (Research In Motion, 2008).

Blackberry Push API mengoptimasi waktu (*time*) dan lokasi (*location*) dalam memberikan peringatan atau notifikasi, seperti berita, perbankan dan saham, skor pertandingan olahraga, dan lainnya. *Push* data ke BlackBerry *Smartphone* adalah

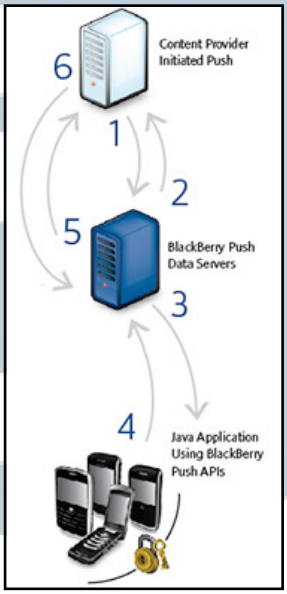
cara paling efektif bagi pemakai untuk mendapatkan suatu informasi. Data dapat dikirim ke aplikasi Java di dalam BlackBerry pemakai dengan menggunakan Blackberry Push API. Aplikasi yang dikembangkan dengan Blackberry Push API tidak perlu secara berkala menarik informasi baru dari *server*. Mungkin saja penarikan informasi dan respon yang diterima secara satu persatu memang dalam jumlah yang kecil, namun biaya ataupun daya yang dibutuhkan akan menanjak dengan cepat jika banyak aplikasi yang berjalan juga sedang menarik data (Fabris, 2010).

Blackberry Push API didesain untuk *push* sejumlah data dalam suatu besaran. *Content provider* dapat melakukan *push* data hingga 8 *Kilobytes* ke satu BlackBerry untuk satu kali pengiriman. Untuk menjaga kontrol tetap pada *user* dan sebagai pertimbangan dalam menyampaikan data yang di-*push* maka, jika data yang dikirim masih belum lengkap dengan 8 *Kilobytes* maka *user* dapat memutuskan untuk menerima atau tidak (Research In Motion, 2008).

Aplikasi yang menggunakan Blackberry Push API juga dapat mereduksi akibat yang ditimbulkan dari penggunaan jaringan berlebihan. Blackberry Push API mengirim data lewat suatu *background process* tanpa campur tangan *user*. Data yang paling baru akan diterima dan sudah tersinkronisasi ketika mereka membuka aplikasi. Blackberry Push API juga meminimalisir akibat dari pemakaian daya baterai. Dibanding mengecek data secara aktif dimana menimbulkan kerja yang lebih dan daya yang lebih besar, Blackberry Push API dengan sederhana hanya menggunakan listener dalam *background process* untuk menerima data *push* dari *server*. Setelah

berhasil dikirim, aplikasi mempunyai kekuatan untuk mengolah informasinya kembali jika diperlukan (Research In Motion, 2008).

Cara kerja BlackBerry *Push Service* dan BlackBerry *Push* API sebagaimana yang ditunjukkan pada gambar 2.3.



Gambar 2.3 Cara kerja BlackBerry Push Service (Research In Motion, 2011)

1. *Content provide r*mengirimkan *push request*.
2. Blackberry smartphonemengembalikan respons.
3. BlackBerry *infrastructure* melakukan *push* data ke Blackberry smartphone.
4. BlackBerry *smartphone* mengembalikan respons ke Blackberry smartphone.
5. BlackBerry *infrastructure* melanjutkan *acknowledgment* ke *content provider*.
6. *Content provider* mengembalikan status bahwa *acknowledgment* sudah diterima ke Blackberry smartphone.

2.4. Sistem Basis Data

2.4.1. Oracle 11g Database

Sekarang ini Oracle 11g adalah *databasesoftware* paling kuat yang ada di pasaran *software*. Oracle 11g juga sebagai memimpin pemakaian dan penjualan *databasesoftware* di seluruh dunia dan telah menjadi standar dalam arsitektur perusahaan untuk mengatur data tanpa mempermasalahkan seberapa besar data dan kompleksitasnya (Wessler, Ruel, & Zeis, 2009).

Oracle adalah *database software* yang mengorganisasikan data secara efisien dalam suatu keterkaitan yang beraturan. Model relational adalah konsep dimana data disimpan secara logika. Desain elemen-elemen dalam bentuk tabel-tabel. Tabel mempunyai kolom-kolom dan kolom tersebut punya atribut. Tabel yang telah diorganisasi digunakan untuk menyimpan data yang spesifik. Tabel-tabel yang dibuat saling berkaitan satu sama lain dengan menggunakan *primary key*(Singh & Pottle, 2009).

Oracle dalam bahasa pemrograman *database* memakai Procedural Language/Structured Query Language (PL/SQL). PL/SQL bersifat *portable*, performa tinggi dalam pemrosesan transaksi dan juga mempunyai beberapa keunggulan (Singh & Pottle, 2009).

a. Mendukung SQL
b. Mendukung pemrograman berorientasi objek
c. Performa yang lebih baik
d. Produktivitas yang tinggi

e. Integrasi yang sangat erat dengan Oracle *Database*

f. Keamanan yang rapat.

2.4.2. ActiveX Data Objects Database(ADOdb)

ADOdb adalah sebuah kumpulan *tools* (*library*) yang berupa abstraksi untuk berbagai macam *database*. ADOdb menjembatani bahasa pemrograman aplikasi dengan bahasa pemrograman *database*. Contohnya jika PHP adalah bahasa pemrograman aplikasi berbasis *web* dan *database*-nya terdiri dari satu *server* Oracle dan satu *server* MySQL maka ADOdb digunakan sebagai *library* untuk mengkoneksikan PHP dengan data dari MySQL maupun PHP, jadi cukup dengan satu *library* ADOdb tidak perlu *library* MySQL khusus dan Oracle khusus. Cara kerja secara umum untuk mengambil data lewat ADOdb (Vasudevan, 2001).

a. Buat suatu koneksi ADO ke sebuah *database*.

b. Buka koneksi tersebut.

c. Buat suatu *recordset* dengan mengeksekusi SQL *command*.

d. Buka *recordset*.

e. Ekstraksi data yang diperlukan dari *recordset*.

f. Tutup *recordset*.

g. Tutup koneksi ADO.

2.5. Pemrograman Aplikasi

2.5.1. Hypertext Preprocessor (PHP)

PHP Singkatan dari PHP: *Hypertext Preprocessor*, merupakan bahasa *scripting* yang sering digunakandalam pengembangan aplikasi berbasis *web*. PHP merupakan bahasa pemprograman berbasis *web* yang populer dan banyak digunakan oleh pengembang, dikarenakan kemudahan penggunaannya dan PHP termasuk perangkat lunak sumber terbuka (*open source software*). Dengan model pengembangan perangkat lunak sumber terbuka, membuat PHP dapat diambil dengan bebas di Internet, merubah *source code* (kode sumber) dan mendistribusikan kembali. PHP juga bisa dijalankan di berbagai *webserver*, seperti Apache, IIS, Lighthttpd, Nginx, dan lain sebagainya, dan mendukung berbagai *platform* sistem operasi. Selain itu, PHP mendukung berbagai *database* populer seperti MySQL, MsSQL, Sybase, Oracle, PostgreSQL, Generic ODBC dan lain sebagainya (Hanan, 2011).

2.5.1.1. Client URL (cURL)

cURL (dibaca: si URL) singkatan dari *Client* URL dan dikembangkan oleh Daniel Stenberg pada tahun 1998 sebagai alat bantu *commandline* untuk transfer *files* dengan sintaks URL melalui bermacam-macam protokol (FTP, HTTP, HTTPS, SCP, SFTP, TELNET, LDAP, dsb). Sedangkan libcurl adalah *libraryportable* yang menyediakan *interface* (untuk berbagai bahasa pemrograman, seperti Perl, Python,PHP, dsb) terhadap fungsionalitas cURL (Corno, 2009).

2.5.2. Java

Java sebagai salah satu bahasa pemrograman baru menjanjikan banyak kemudahan bagi programer junior maupun senior. Java merupakan bahasa pemrograman berorientasi objek yang dikembangkan dengan model yang mirip dengan bahasa C++ dan Smalltalk, namun dirancang agar lebih mudah dipakai dan *platform independent*, yaitu dapat dijalankan di berbagai jenis sistem operasi dan arsitektur komputer (Pujiadi, 2011). Bahasa ini juga dirancang untuk pemrograman di Internet sehingga dirancang agar aman dan *portable*. BlackBerry Java *Standard Development Kit* (SDK) digunakanuntuk pembuatan aplikasi pada BlackBerry.

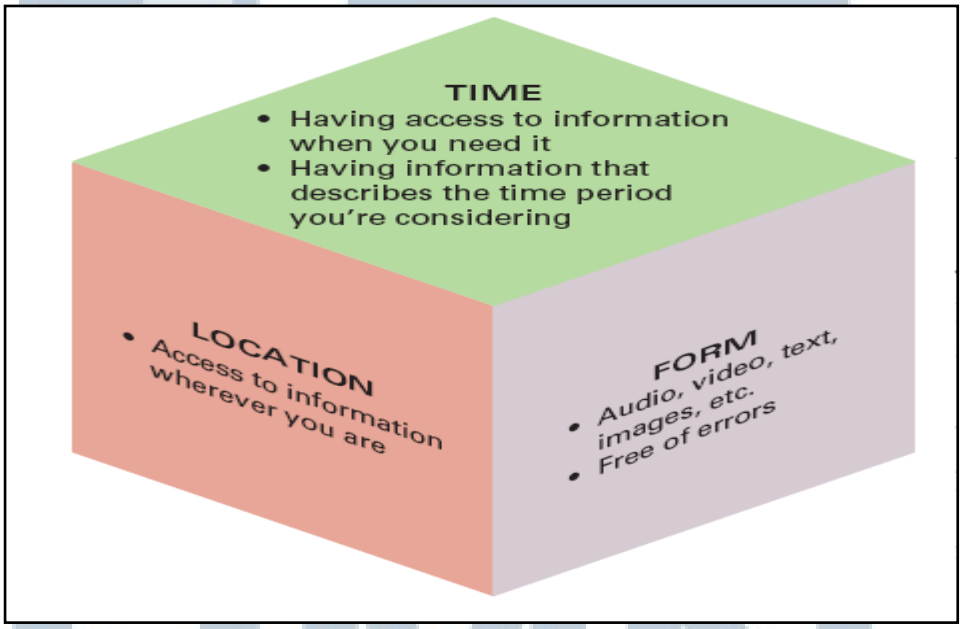
2.5.2.1. BlackBerry Java Development Environment

BlackBerry® Java® *Development Environment* (BlackBerry JDE) adalah lingkungan dan program simulasi pengembangan yang benar-benar terintegrasi untuk membuat aplikasi Java Platform, Micro Edition (Java® ME) untuk *smartphone* BlackBerry® berbasis Java®(Research In Motion).

BlackBerry JDE adalah lingkungan Java ME yang kompatibel dengan *Mobile Information Device Profile* (MIDP) untuk pengembang yang ingin aplikasi nirkabelnya benar-benar portabel. Selain itu, BlackBerry JDE menyediakan paket lengkap antarmuka dan utilitas untuk memanfaatkan beberapa fitur unik di *smartphone* BlackBerry, salah satunya adalah Blackberry Push API.

2.6. Dimensi Informasi

Informasi mempunyai berbagai macam karakteristik untuk mendeskripsikan kualitas dari informasi tersebut. Perbedaan antara baik dan buruknya suatu informasi dapat diidentifikasikan dengan mempertimbangkan apakah informasi tersebut mempunyai semua atau sebagian atribut dari informasi yang berkualitas. informasi dalam dimensi personal atau organisasional (Fuad, 2010).



Gambar 2.4 Dimensi Personal Informasi (Fuad, 2010)

Dimensi personal dari suatu informasi mempertimbangkan tiga sudut pandang: waktu, lokasi, dan bentuk (Fuad, 2010).

a. Dimensi Waktu Informasi

Informasi seperti sebagian besar sumber daya organisasi lainnya, dapat menjadi kadaluarsa dan tidak terpakai lagi. Contohnya ketika ingin membuat suatu transaksi

saham, kita harus tahu harga terkini. Jika kita menunggu sehari saja untuk melihat harga saham, maka mungkin kita tidak bisa bertahan dalam aktivitas pasar saham. Tidak diragukan lagi bahwa hampir semua tranasksi dan informasi tentang saham di saat ini selalu muncul di Internet (Laudon & Laudon, 2004).

Informasi dapat berguna dan relevan hanya jika mendeskripsikan periode waktu yang tepat untuk isi informasi di dalamnya. Dimensi waktu mendeskripsikan periode waktu dimana informasi akan digunakan dan frekuensi waktu dimana informasi tersebut harus diterima (Laudon & Laudon, 2004).

1. *Timeliness*

   Informasi harus tersedia saat dibutuhkan. Jika informasi disediakan terlalu cepat, maka mungkin informasi tersebut tidak lagi menjadi informasi terkini saat dibutuhkan. Jika terlalu terlambat maka tidak akan ada gunanya.

2. *Currency*

   Informasi haru merefleksikan kondisi terkini saat disediakan. Suatu informasi juga mungkin digunakan untuk mengindikasikan bahwa informasi yang ada seiring dengan waktu dapat di-*update* sehingga dapat disesuaikan kondisi atau kenyataan yang berubah.

3. *Frequency*

   Untuk mendukung informasi harus ada saat dibutuhkan, maka informasi juga harus tersedia sesering kebutuhan akan informasi itu muncul. Normalnya informasi disediakan dalam suatu interval waktu yang ditentukan, contohnya

sejumlah organisasi mungkin membutuhkan laporan penjualan mingguan sedangkan organisasi lain hanya perlu laporan penjualan bulanan.

4. *Time Period*

   Informasi harus menutupi kebutuhan periode waktu yang tepat. Misalnya dalam *sales forecasting*, informasi harus menyediakan laporan tentang performa waktu lampau, penjualan saat ini dan prediksi penjualan ke depan sehingga penerima informasi dapat melihat kondisi lampau, sekarang dan ke depannya.

b. Dimensi Lokasi Informasi

Dimensi lokasi dari informasi artinya bahwa akses ke informasi dimanapun si penerima berada. Idealnya, dimanapun lokasi informasi tersebut diletakkan dan dimanapun penerimanya bukan menjadi suatu masalah (Fuad, 2010). Penerima harus bisa mengakses informasi dari sebuah kamar hotel, rumah, pusat kegiatan mahasiswa di kampus, di tempat kerja, saat berjalan kaki di tengah jalanan, ataupun saat sedang di dalam pesawat terbang. Dimensi lokasi ini berkaitan erat dengan *mobile and wireless computing* (Laudon & Laudon, 2004).

Untuk mempertahankan informasi yang ada tetap bersifat pribadi dan aman saat menyediakan remote access untuk karyawan, banyak bisnis membuat suatu intranet. Intranet adalah suatu Internet organisasi yang bersifat internal dan dijaga aksesnya oleh sebuah fitur yang dinamakan dengan *firewall* (bentuknya bisa berupa *software*, hardware ataupun kombinasi keduanya). Jadi, jika organisasi mempunyai suatu intranet dan orang di dalamnya mau untuk mengakses informasi saat pergi dari

kantor, yang dibutuhkan hanya akses ke dalam *Web* dan masuk ke dalam Intranet yang memungkinkan informasi dapat diterima (Fuad, 2010).

c. Dimensi Bentuk Informasi

Dimensi bentuk informasi mempunyai dua aspek utama. Hal pertama cukup sederhana yaitu kemas informasi dalam bentuk yang paling berguna dan dapat dimengerti seperti audio, video, grafis dan bentuk lainnya (Laudon & Laudon, 2004). Aspek kedua adalah akurasi dimana informasi dibutuhkan terbebas dari segala kesalahan atau *error*. Bayangkan jika informasi adalah sebuah barang dan barang itu ternyata rusak, kita akan menjadi tidak puas. Seperti menerima informasi yang salah, kita juga tidak akan senang. Dimensi bentuk mendeskripsikan bagaimana informasi dipresentasikan kepada penerima (Laudon & Laudon, 2004).

1. *Clarity*

   Informasi harus dipresentasikan dalam bentuk yang memadai untuk calon penerima. Penerima bisa mengalokasikan informasi dengan cepat dan mengerti informasi dengan mudah.

2. *Detail*

   Informasi harus berisi tingkatan detil yang tepat untuk mencapai kebutuhan penerima. Contohnya, dalam beberapa kasus dibutuhkan informasi yang sangat detil sedangkan penerima lain membutuhkan informasi berupa rangkumannya saja.

3. *Order*

   Informasi harus disediakan dalam urutan yang benar. Sebagai contoh biasanya bagian manajemen akan menginformasikan rangkuman pada awal laporan mereka. Ini membuat para manajer bisa menempatkan dan mengerti aspek terpenting dari laporan sebelum melihat ke detil informasinya.

4. *Presentation*

   Informasi harus bisa dipresentasikan dalam sebuah bentuk yang bisa dimengerti, beberapa metode berbeda dapat digunakan untuk membuat informasi lebih jelas dan lebih bisa diakses untuk penerima. Contohnya untuk mempresentasikan informasi numerik maka bisa digunakan bentuk grafis, *chart* ataupun tabel.

5. *Media*

   Informasi harus bisa dipresentasikan menggunakan media yang tepat dalam penyampaiannya. Contohnya informasi formal normalnya dipresentasikan dalam bentuk laporan tercetak sedangkan *slide show* dipresentasikan menggunakan proyektor.

2.7. Push Mobile Technology dalam Pendidikan Tingkat Lanjut

Dalam dunia pendidikan tingkat lanjut sekarang ini hampir semuanya berpusat pada *personal computer* (PC). Sesungguhnya, *push* tidak terbatas pada PC semata, bahkan pembelajaran dapat di-*push* dengan berbagai cara. Salah satu contoh aspek yang paling bisa diperhatikan dalam generasi saat ini adalah mereka menggunakan *mobile devices* untuk mengakses informasi digital. Statistik

penggunaan jaringan mengindikasikan kuat dan meyakinkan penurunan akses melalui PC secara relatif dibandingkan akses *mobile*(Jones, 2010).

Dalam sebuah penelitian telah dicoba untuk mengaplikasikan *push* dengan *Short Message Services* (SMS) (Jones, 2010). Setiap hari, mahasiswa menerima bimbingan bahan belajar di handset mereka. Peneliti memonitor penyerapan informasi dan penggunaan dalam periode waktu 20 minggu. Kemudian peneliti memfokuskan ke arah tugas per grup di dalam panduan yang diterima mahasiswa lewat SMS. Peneliti menemukan bahwa 90% dari rata-rata mahasiswa membaca informasi tersebut. Sedangkan kontras dengan penggunaan *email* sebagai media, hanya di bawah 30% yang membaca informasi itu. Sangat sulit untuk memahami sejauh mana hal-hal baru tersebut mendukung hasil penelitian ini tetapi peneliti tidak menerima respon untuk menyarankan penyesuaian supaya menjadi terbiasa pada hal tersebut di minggu-minggu berikutnya setelah penelitian berakhir. Apa yang sebenarnya kita temukan adalah saran yang lebih spesifik lebih baik daripada saran yang lebih umum. Ada suatu preferensi yang kuat bagi mahasiswa untuk menerima informasi lewat SMS yang berkaitan dengan modul pembelajaran. Secara umum hasilnya mendorong penyerapan informasi. Hal ini memperkuat anggapan bahwa kustomisasi dari isi informasi adalah aspek kunci dari *e-learning* yang efektif.

Meskipun SMS murah dan berguna, SMS terbatas pada besar teks dan pendeknya (Jones, 2010).Keuntungannya teks memungkinkan untuk mengakomodasi *Uniform Resource Locator* (URL). Kebanyakan URL kurang dari 168 karakter dan ada juga yang lebih kecil lagi jika disiapkan dengan format ‘bit.ly’. Dari penelitian,

hampir semua mahasiswa memiliki *smartphone*. Bahkan sebagian besar memiliki lebih dari satu *mobile devices*. Berarti kebanyakan dari mereka dapat dengan mudah mengakses konten media melalui *web browser* di dalam *smartphone*.

2.8. Teknik Sampling

Untuk merepresentasikan populasi objek penelitian yang berjumlah lebih dari 2000 orang, digunakan teknik *sampling*. Alasannya karena mahasiswa dapat dianggap sebagai obyek yang homogen sehingga *random sample* dapat digunakan.

Teknik *sampling* yang dipakai sesuai kaidah teori Taro Yamane. Perhitungan *sample* ditunjukkan dengan rumus (Yamane, 1967).

$$n = \frac{N}{Nd^2 \div 1} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

Keterangan:

n = Ukuran *sample*.

N = Jumlah populasi.

d = Tingkat presisi.

Berikut adalah penjelasan teknik sampling yang digunakan dalam penelitian ini.

1. Simple *Random Sampling* atau *Sample* Acak Sederhana

   Cara atau teknik ini dapat dilakukan jika analisis penelitiannya cenderung deskriptif dan bersifat umum. Perbedaan karakter yang mungkin ada pada setiap unsur atau elemen populasi tidak merupakan hal yang penting bagi

rencana analisisnya. Misalnya, dalam populasi ada wanita dan pria, atau ada yang kaya dan yang miskin, ada manajer dan bukan manajer, dan perbedaan-perbedaan lainnya. Selama perbedaan gender, status kemakmuran, dan kedudukan dalam organisasi, serta perbedaan-perbedaan lain tersebut bukan merupakan sesuatu hal yang penting dan mempunyai pengaruh yang signifikan terhadap hasil penelitian, maka peneliti dapat mengambil *sample* secara acak sederhana. Dengan demikian setiap unsur populasi harus mempunyai kesempatan sama untuk bisa dipilih menjadi *sample*. Prosedurnya, susun "*sampling frame*", tetapkan jumlah *sample* yang akan diambil, tentukan alat pemilihan *sample*, dan pilih *sample* sampai dengan jumlah terpenuhi (Mustafa, 2003).

2. *PurposiveSampling*

Sesuai dengan namanya, *sample* diambil dengan maksud atau tujuan tertentu. Seseorang atau sesuatu diambil sebagai *sample* karena peneliti menganggap bahwa seseorang atau sesuatu tersebut memiliki informasi yang diperlukan bagi penelitiannya. Dua jenis *sample* ini dikenal dengan nama *judgement* dan *quota sampling* (Mustafa, 2003).*Sample* dipilih berdasarkan penilaian peneliti bahwa dia adalah pihak yang paling baik untuk dijadikan *sample* penelitiannya. Misalnya untuk memperoleh data tentang bagaimana satu proses produksi direncanakan oleh suatu perusahaan, maka manajer produksi merupakan orang yang terbaik untuk bisa memberikan informasi. Jadi,

judment sampling umumnya memilih sesuatu atau seseorang menjadi *sample* karena mereka mempunyai "*information rich*" (Mustafa, 2003)

2.9. Strategi Teknologi Pendidikan Tingkat Lanjut

Tiga dimensi perubahan yaitu: organisasi, teknologi dan ekonomi menjadi momentum besar di dalam bidang pendidikan tingkat lanjut (Ernst, Katz, & Sack, 1994). Setiap perubahan menyebabkan para pelaku bidang pendidikan mengalami diskusi, pembelajaran, rasa frustasi dan pada sebagian kasus terjadi ketakutan akan perubahan itu.

Kekuatan organisasional akan membawa perguruan tinggi atau universitas ke arah koordinasi yang lebih baik antar departemen dan meningkatkan kepercayaan individu di dalamnya. Peran teknologi adalah mendorong suatu distribusi data baik di *client* ataupun *server* dan suatu lingkungan pemrosesan data yang lebih kooperatif terhadap pengguna (Ernst, Katz, & Sack, 1994).

Pimpinan universitas dan para profesional IT di kampus telah memperhitungkan besarnya perubahan lingkungan universitas dan segi institusionalnya di dalam era informasi modern ini. Dalam berbagai macam publikasi dan pertemuan di perguruan tinggi sekarang ini yang menjadi topik pembahasan adalah seputar transformasi, restrukturisasi, *reengineeering*, *rethinking* dan inovasi.

Hidup dan bekerja dalam konteks sebuah perubahan yang sangat cepat dan mempunyai banyak pilihan untuk maju terdapat pemilihan yang tepat dan ada juga yang sesat sehingga arah penentuan kebijakan dalam universitas harus kritis dan

waktu menjadi hal yang esensial. Momentum perubahan terjadi sangat cepat dan terus meningkat, tidak cukup jika hanya membuat suatu keputusan yang tepat saja karena jika keputusan tepat tetapi tidak dilakukan dengan cepat maka sebagai suatu organisasi perguruan tinggi menanggung resiko untuk tertinggal dengan perubahan itu sendiri dibandingkan dengan menjadi sebuah organisasi pendidikan yang menyediakan, memfasilitasi atau bahkan menjadi pencipta perubahan itu. Hal ini seperti diungkapkan dalam pemikiran Lee Iaccoca, mantan presiden Ford dan Chrysler, yang mengatakan, "Pimpin, ikuti, atau keluar dari persaingan"

Salah satu tren perubahan yang terjadi adalah konsumer dalam hal ini adalah seluruh fakultas dan staf yang ada di dalam universitas dan departemen lain menginginkan suatu layanan mutakhir yang membutuhkan akses data lebih banyak. Contoh kasusnya adalah staf dan pihak fakultas termasuk mahasiswa terus menerus mengkritik tidak bisa diaksesnya kebutuhan data dari sistem informasi universitas (Ernst, Katz, & Sack, 1994).

Untuk merespon tekanan dalam kebutuhan akses informasi, universitas membutuhkan *customer orientation*. Model kontrol birokratif dan model pemecahan masalah inkremental membantu perkembangan budaya organisasi yang berkarakter seperi penjaga pintu, *bodyguard*, mengontrol dan memberikan aturan. Bandingkan jika mengadakan pembicaraan bisnis baik kepada partner bisnis ataupun calon mahasiswa tentang bagaimana memuaskan konsumer daripada membicarakan tentang pengaturan sistem informasi yang ada (Ernst, Katz, & Sack, 1994).

Model tradisional dari pengembangan sistem informasi mengasumsikan bahwa petinggi operasional unviersitas menginginkan bagian pengembangan IT membuat suatu organisasi fungsional dari sistem yang mendefinisikan persayaratan apa saja yang dibutuhkan sistem. Sekarang ini visi dari pengembangan sistem informasi sendiri bergerak dari fungsi vertikal yang mementingkan permintaan petinggi kampus ke proses kebutuhan horizontal yang menaruh fokus kepada *end users* dari sistem yaitu mahasiswa, fakultas atau jurusan, karyawan, dan *stakeholder* lain. Strategi lama yaitu mementingkan produk apa yang harus dibuat untuk kebutuhan pihak universitas, berubah menjadi strategi baru yaitu mementingkan *end users* membutuhkan produk seperti apa (Ernst, Katz, & Sack, 1994).

Strategi lama dan baru dalam menghadapi tren seperti ini dapat disimpulkan dalam gambar berikut.

Old Strategies	Emergent Strategies
Do things right	Do the right things, right
Assure compliance	Become a problem solver
Foster specialization	Empower generalists
Manage by exception	Create centers of competency
Safeguard institutional data	Promote access to information

Gambar 2.5 Strategi dalam menghadapi tren di dunia pendidikan tingkat lanjut (Ernst, Katz, & Sack, 1994)

Untuk menghadapi tantangan masa kini dan ke depannya, identifikasi terhadap halangan-halangan yang dihadapi oleh *end users* harus secara terbuka dan

kritis. Bukan beralasan tidak bisa karena begitu adanya sistem yang berjalan namun lebih baik memulai pengembangan fitur dalam sistem yang belum ada secepat mungkin. *Stakeholder* secara bersama harus mendesain suatu arsitektur teknologi dan infrastruktur jaringan yang bisa menaikkan akses dalam berbagai kebutuhan. Penghilangan batas antar teknologi apa yang dipakai menjadi panduan dalam rencana dan program pengembangan informasi teknologi (Ernst, Katz, & Sack, 1994).