BAB II

LANDASAN TEORI

2.1 Sumber Daya Manusia

Dalam dunia bisnis, terdapat 4 jenis aset utama yang mempengaruhi kinerja sebuah perusahaan yaitu:

1. *Physical*, yang meliputi bangunan, tanah, *furniture*, komputer, kendaraan, alat, dan lainnya
2. *Financial*, yang meliputi *cash*, *financial resources*, *stocks*, dan *financial securities*, dan lainnya
3. *Intangible*, yang meliputi penelitian dan pengembangan, paten, sistem informasi, proses operasi, dan lainnya
4. *Human*, yang meliputi orang dengan talenta, kemampuan, pengalaman, keahlian, hubungan, dan lainnya

Meskipun asset yang disebutkan memiliki tingkat kepentingan yang berbeda untuk setiap perusahaan, *Human* merupakan penghubung dan penggerak asset lainnya sehingga dapat menggerakkan dan menggembangkan perusahaan (Jackson, 2008).

Organisasi adalah entitas yang terdiri dari sekelompok orang yang memiliki suatu tujuan. Organisasi merupakan bagian yang cukup penting dalam perusahaan dalam mengatur aset *Human* sehingga perusahaan dapat berjalan dan berkembang. Produktivitas dari sebuah perusahaan dapat dinilai dari bagaimana cara mengatur

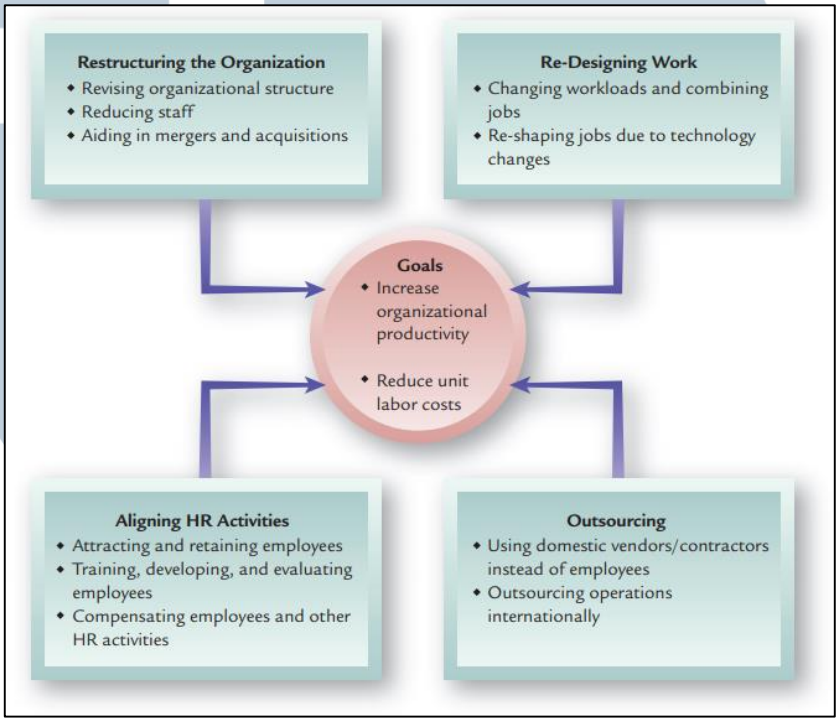
dan mengolah aset yang dimiliki perusahaan sehingga dapat berjalan dengan baik dan efisien.

Menurut John H. Jackson (2018), sebuah produktivitas yang lebih baik bukan berarti mengeluarkan lebih banya *output* ataupun mengurangi jumlah *resource* yang digunakan untuk menghasilkan produk dengan jumlah yang sama. Produktivitas diartikan sebagai sebuah tolak ukur untuk mengukur jumlah dan kualitas pekerjaan yang dihasilkan berdasarkan dari *resource* yang digunakan terutama aset *Human*.

Terdapat empat cara utama yang dilakukan pada organisasi sehingga dapat meningkatkan produktivitas sebuah perusahaaan seperti pada Gambar 2.1, yaitu:

1. *Organizational Restructuring,* kegiatan untuk melihat kembali sebuah struktur organisasi dan melakukan perubahan struktur baik dari mengurangi tenaga kerja, menggabungkan beberapa bagian dari organisasi.
2. *Redesigning Work,* kegiatan yang dilakukan untuk mengubah pekerjaan yang dilakukan oleh suatu individu dengan mengerjakan beberapa pekerjaan baru maupun menambah suatu teknologi dan alat sehingga suatu individu dapat bekerja dengan lebih cepat dan efisien.
3. *Aligning HR Activities,* kegiatan yang dilakukan untuk memastikan yang dilakukan pada organisasi untuk meningkatkan produktivitas seperti memberikan pelatihan, mengukur performa karyawan (*Performance Management*), mengatur kompensasi karyawan, dan segala aktivitas yang tidak melawan dari produktivitas.
4. *Outsourcing Analyses,* kegiatan yang menganalisa kebutuhan untuk melakukan *outsourcing* seperti memperhitungkan kerugian dan keuntungan, negosiasi

dengan vendor *outsourcing*, memastikan bahwa orang yang dikontrak dapat bekerja secara legal dan wajar, dan menghubungkan antara karyawan dengan orang *outsource*.



Gambar 2.1 Aktivitas Meningkatkan Produktivitas Organisasi (Jackson, 2008)

2.2 Human Resource Information System

Menurut Endri H (2019), *Human Resource Information System* (HRIS) merupakan perangkat lunak yang digunakan untuk mengatur Sumber Daya Manusia pada perusahaan oleh *Human Resources Department* (HRD) secara sentralisasi. Dengan adanya HRIS, kegiatan yang berkaitan dengan Sumber Daya Manusia salah satunya seperti rekrutmen, *learning* & d*evelopment*, kehadiran, ketidak hadiran (libur, ijin atau cuti), evaluasi kinerja karyawan. remunerasi, penggajian dan pembayaran pajak akan menjadi lebih. Tanpa adanya HRIS, pekerjaan yang dilakukan oleh HRD secara *manual* cukup merepotkan dan tidak efisien terutama apabila karyawan pada perusahaan bertambah banyak.

Terdapat beberapa keuntungan dengan menggunakan HRIS sebagai alat untuk membantu kegiatan *Human Resources Management* (HRM) yaitu sebagai berikut:

2.2.1 Memonitor Kinerja Karyawan

HRIS memudahkan manajemen untuk mengidentifikasi karyawan yang cepat berkembang dan karyawan yang perkembangannya lambat. Manajer HRD dapat mengukur pencapaian karyawan, dan memudahkan manajemen dalam mengevaluasi karyawan.

2.2.2 Menghemat Biaya Operasi

HRIS membantu meringankan biaya operasional HRD dengan menghilangkan penggunaan kertas atau aplikasi-aplikasi lainnya yang tidak efisien. Perangkat lunak ini juga mengurangi kebutuhan akan tenaga kerja untuk mengelola SDM, karena seluruh proses dapat ditangani secara otomatis.

2.2.3 Membantu Proses Pengambilan Keputusan

HRIS membantu manajemen mengambil keputusan dengan lebih baik. Karena seluruh informasi terkait SDM dapat ditinjau melalui satu sistem terpusat, manajemen dapat dengan lebih cepat dan mudah mengetahui siapa saja yang sudah cukup berkontribusi ke perusahaan dan siapa yang belum. Manajemen dapat melakukan tindakan dengan lebih cepat untuk mengatasi kesenjangan keterampilan karyawan.

2.2.4 Meningkatkan Keamanan Data Karyawan

Data karyawan bersifat personal dan seharusnya hanya boleh diakses oleh orang-orang yang berkepentingan saja. Sistem HRM memungkinkan manajer HRD

atau staff terkait untuk mengontrol siapa saja yang dapat mengakses informasi dalam sistem, sehingga resiko pencurian data dapat diminimalkan.

2.2.5 Memudahkan Pendistribusian Gaji

HRIS menjadikan proses pengelolaan data kekaryawanan lebih cepat, hemat tempat dan proses penggajian menjadi lebih cepat dan mudah. *Human Error* yang sering menjadi salah penyebab kesalahan dalam penghitungan gaji dapat diminimalkan, sebab sistem menghitungnya sesuai dengan informasi karyawan yang terkait dengan faktor pemberian gaji suatu perusahaan.

2.3 SAP – HCM Organizational Management Module

SAP Human Capital Management (SAP-HCM) merupakan salah satu modul utama dalam SAP yang berfokus pada membangun manajemen yang terorganisir dalam perusahaan (Habmann, dkk.,2010). SAP-HCM menangani seluruh kegiatan pengaturan SDM pada perusahaan. Salah satu kegiatan SDM yang dilakukan oleh SAP-HCM adalah kegiatan untuk mengorganisir karyawan yang bekerja dibawah perusahan untuk mencapai tujuan yang telah ditentukan. Kegiatan tersebut dikerjakan dalam modul *Organizational Management*.

Dalam modul Organizational Management, data disimpan dalam dua *infotype* utama yaitu Infotype 1000 yang digunakan untuk menyimpan seluruh data objek pada SAP-HCM, dan Infotype 1001 yang digunakan untuk menyimpan data relasi antar objek. Terdapat beberapa jenis objek yang digunakan pada SAP-HCM yaitu sebagai berikut:

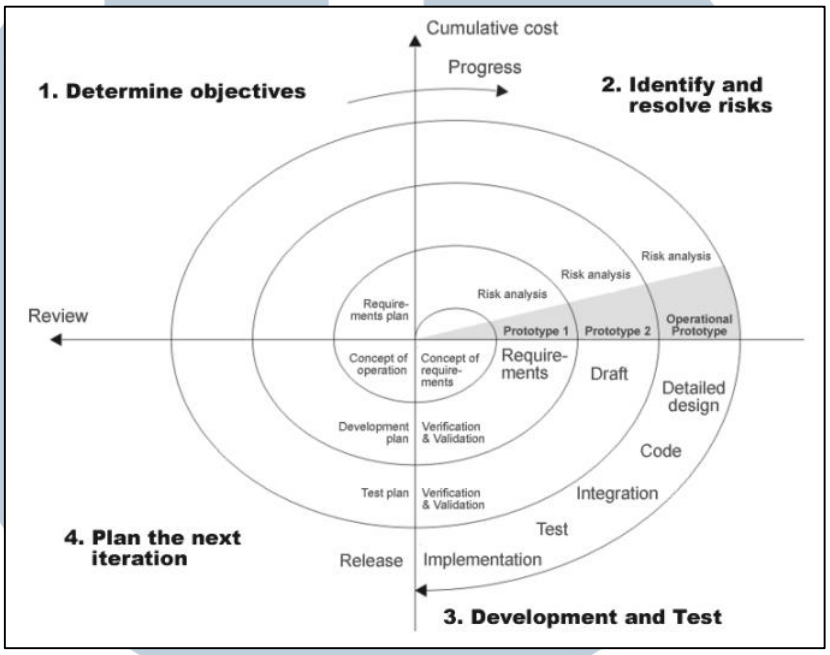
1. Organizational Unit (O), merupakan objek yang merepresentaskan sebuah bisnis, tim, sektor, maupun pengelompokan lainnya. Dengan adanya Organizational Unit, maka hirarki dalam sebuah perusahaan dapat dibentuk sehingga dapat terlihat dengan jelas kedudukan dari sebuah kelompok dalam suatu perusahaan.
2. Position (P), merupakan objek yang digunakan sebagai penyedia tempat yang nanti diisi oleh karyawan. Apabila tidak ada karyawan yang menempati Position, maka Position akan dinyatakan "Vacant" sehingga dapat ditempati oleh karyawan melalui E-Recruiting pada SAP-HCM yang diatur oleh modul Personel Adminstration. Position selalu direlasikan kepada sebuah *Organizationa Unit* pada satu waktu. Terdapat dua dasar relasi yang dapat dilakukan Position terhadap Organizational Unit yaitu "Belongs To" yang berarti ditugaskan kepada sebuar Organizational Unit, dan "Chief" yang berarti Position menjadi kepala sebuah Organizational Unit.

2.4 System Development Life Cycle Spiral

System Development Life Cycle (SDLC) merupakan sebuah metode yang digunakan dalam merancang, membangun, dan mengontrol sebuah Sistem Informasi. Terdapat beberapa jenis SDLC yang digunakan sesuai dengan kondisi proyek yang digunakan, salah satunya adalah metode Spiral.

Spiral merupakan metode SDLC yang merupakan solusi dari lemahnya Waterfall (Mishra, 2013). Metode ini diperkenalkan oleh Boehm pada tahun 1988 dengan memodifikasi Waterfall menjadi beberapa bagian iterasi dimulai dari iterasi

kecil seperti pada Gambar 2.2. Spiral ini didasari oleh filosofi *start small, think big* (Ruparelia, 2010).



Gambar 2.2 Boehm Spiral Life Cycle (Boehm, 2000)

Spiral dilakukan dengan memecah proyek menjadi *segment* kecil juga memberikan evaluasi dan mempertimbangkan untuk melanjutkan proyek yang dikerjakan. Setiap *cycle* melewati proses yang sama untuk setiap *segment* hingga proses *code* pada masing-masing *segment*. Metode Spiral dapat digabungkan dengan metode lainnya seperti *Waterfall*, *Prototyping*, dan *Incremental* karena memiliki dasar konsep yang serupa dengan *Spiral*.

Dalam setiap *cycle* dalam proses spiral, *prototype* akan dibangun lalu diverifikasi terhadap *requirements* yang telah didefinisikan, dan divalidasi melalui tahapan *testing*. Terdapat dua kategori dalam risk yang ditemukan, yaitu *performance related risk,* dan *development risk*. Apabila *development risk* lebih menonjol, maka proses pengembangan projek akan mengikuti *incremental waterfall* model. Apabila *performane related risk* lebih menonjol, maka spiral dapat

dilanjutkan ke perkembangan berikutnya. Dengan menerapkan mekanisme tersebut, diharapkan prototype yang dihasilkan pada iterasi selanjutnya memiliki resiko yang lebih kecil (Ruparelia, 2010).

Terdapat kelemahan dalam menggunaan Spiral sebagai metode SDLC pada sebuah proyek, yaitu:

1. Menentukan komposisi *segment* yang akan dibangun pada setiap *cycle* yang akan dilakukan dalam sebuah proyek.
2. Kurangnya *reuseability* karena setiap *cycle* baru bertambah kompleksitas dari proyek
3. Tidak ada kontrol antar *cycle* yang dilakukan pada Spiral.
4. Memungkinkan proyek berakhir menggunakan SDLC Waterfall.
5. *Cycle* dilakukan tanpa memiliki *deadline* yang jelas meningkatkan resiko *budget* dan waktu penyelesaian yang tidak sesuai.

Meskipun berbagai jenis proses model yang lebih baru bermunculan, terdapat beberapa kondisi dimana proses model Spiral dapat menjadi proses model yang cocok dan efektif untuk mengembangkan sebuah sistem (Alshamarni, 2015), yaitu:

1. Ketika pengguna sistem tidak mengetahui *requirements* yang pasti dan lengkap.
2. Sistem yang dibangun memiliki tingkat resiko sedang menuju resiko yang cukup tinggi.
3. Memprioritaskan pengatasan terhadap resiko yang ada, dan *cost* yang akan dibutuhkan.
4. Ketika adanya kemungkinan bahwa perubahan yang besar akan diperlukan pada sistem.

5. Sistem yang dibangun memiliki tingkat spesifikasi yang cukup tinggi.
6. Perlunya sebuah kontrol dalam pembangunan dan dokumentasi ketika membangun sistem.
7. Terdapat fitur maupun fungsionalitas yang dapat ditambahkan pada sistem untuk waktu kedepannya.

2.5 Software Testing

Software Testing merupakan sebuah kegiatan untuk mengidentifikasi kekurangan, kesalahan, dan kecacatan dari sebuah peranagkat lunak yang dibangun (Khan, 2012). Tujuan utama dari kegiatan *Software Testing* untuk menjaga kualitas, kehandalan, menjaga kontrol, dan melengkapi kekurangan sebuah perangkat lunak. *Box Approach Testing Technique* merupakan salah satu teknik *Software Testing* yang terdiri dari *Black Box Testing Technique* dan *White Box Testing Technique*.

2.5.1 Black Box Testing Technique

Merupakan teknik pengujian perangkat lunak secara kualitas. Pengujian dilakukan berdasarkan *requirements* yang telah ditentukan tanpa memerlukan pengetahuan mengenai *code* yang ada pada perangkat lunak. Umumnya dilakukan oleh pengguna yang tidak mengetahui bagaimana *code* bekerja di dalam perangkat lunak.

Black Box Testing memiliki peran yang penting dalam *Software Testing* untuk memastikan seluruh fungsionalitas pada perangkat lunak dapat bekerja sesuai dengan yang telah ditentukan para *requirements*. *Black Box Testing* juga menambah perspektif dari pengguna aplikasi yang tidak mengetahui *code* yang berada di dalam

dengan tujuan untuk menemukan kekurangan dalam perangkat lunak baik secara teknis maupun secara konsep.

Namun, menurut Nidra (2012) terdapat beberapa kekurangan ketika menggunakan *Black Box Testing* seperti:

1. Hanya beberapa uji skenario yang dilakukan dalam pengujian sehingga kualitas perangkat lunak tidak teruji secara menyeluruh.
2. Tanpa uji skenario yang jelas dan spesifik, pengujian tidak dapat dijalankan karena tidak memiliki validitas yang kuat.
3. Pengujian tidak dapat mengukur efisiensi dan efiktivitas dari perangkat lunak.

Salah satu metode *Black Box Testing* yang dapat digunakan adalah *Decision Tables*. *Decision Tables* merupakan aturan-aturan yang dapat dibaca dan dibuat oleh perancang untuk mendapatkan sebuah hasil yang diinginkan. *Decision Tables* dapat digunakan apabila terdapat pilihan dan aturan yang harus dijalani untuk mencapai suatu hasil yang diinginkan dalam sebuah perangkat lunak. *Decision Tables* terdiri dari 2 komponen utama yaitu *Condition*, dan *Action*. Berikut merupakan contoh *Decision Table* pada Gambar 2.3.

Condition Entries	Account No.	Signature	Money	Actions
1	Correct	Match	Yes	A1
2	Correct	Match	No	A2
3	Correct	Not match	Yes	?
4	Correct	Not match	No	?
5	Incorrect	Match	Yes	A3
6	Incorrect	Match	No	?
7	Incorrect	Not match	Yes	?
8	Incorrect	Not match	No	?

Gambar 2.3 Contoh Decision Table (Nidhra, 2012)

Salah satu penggunaan dari *Decision Table* dalam pembangunan sebuah perangkat lunak adalah untuk menguji integritas dari sebuat aplikasi web

berdasarkan dari kondisi maupun kriteria pada setiap komponen web seperti pada Gambar 2.4. Dengan menggunakan pengujian terhadap setiap komponen, maka memastikan kualitas dari aplikasi web dimana *Lines to be Covered* merupakan seluruh skenario yang teruji, dan *Covered Lines* merupakan skenario yang berhasil dijalankan dan sesuai ekspetasi(Di Lucca, 2002).

Web Page	# Covered Lines	# Lines to be covered	Coverage Ratio
Login.asp	8	8	100%
Chat.asp	2	3	67%
Read.asp	20	21	95%
Write.asp	4	5	80%
Logout.asp	5	5	100%
Total	39	42	93%

Gambar 2.4 Hasil Tes Integrasi Dari Sebuah Aplikasi Web (Di Lucca, 2002)

2.5.2 White Box Testing Technique

Merupakan teknik pengujian yang berfokus terhadap detail logika, dan struktur data yang ada pada perangkat lunak yang sedang dibangun. Dalam *White Box Testing*, diperlukan orang yang mengetahui cara bekerja *code* pada perangkat yang dibangun. *White Box Testing* dapat dilakukan dalam tahap perancangan maupun implementasi dalam bentuk *source code* perangkat lunak.

Dengan menggunakan *White Box Testing*, dapat memunculkan masalah yang kurang terlihat seperti logika yang kurang tepat, pengaturan struktur data yang redundansi, hingga mengidentifikasi fungsi yang tidak berjalan dengan semestinya. Namun *White Box Testing* memiliki kelemahan seperti membutuhkan orang yang ahli dalam bidangnya untuk menganalisa secara menyeluruh, perlunya penanganan

secara intensif untuk menjaga perancangan dan *source code* sehingga tidak merusak perangkat lunak secara keseluruhan.

Salah satu metode yang digunakan dalam *White Box Testing* adalah *Formal Inspection*. *Formal Inspection* merupakan metode tradisional untuk melakukan pengujian dengan mencari kesalahan dalam bentuk perancangan maupun implementasi berupa *source code* dalam bentuk *review*. Terdapat beberapa komponen yang berada dalam *Formal Inspection* seperti pada Gambar 2.5 yaitu sebagai berikut:

1. *Planning*, menentukan peserta yang tepat dalam melakukan *review*.
2. *Overview*, kegiatan untuk menunjukkan gambaran perangkat lunak secara besar, detail perancangan perangkat lunak, hingga logika yang diimplementasikan dalam perangkat lunak dalam bentuk dokumentasi.
3. *Preparation*, mempelajari perancangan dan logika yang telah diimplementasikan sebelumnya sehingga dapat berfokus kepada bagian yang memiliki kemungkinan terjadi sebuah kesalahan.
4. *Inspection*, proses review untuk menemukan kekurangan, dan kesalahan baik dalam perancangan dan implementasi yang dicatat dalam sebuah dokumentasi.
5. *Rework*, tindakan yang perlu dilakukan untuk menangani masalah yang ditemukan.
6. *Follow Up*, kegiatan yang dilakukan untuk memastikan bahwa perancangan, dan implementasi dapat berjalan tanpa masalah.

Assigned Date	09/02/10		Finished Date	04/03/10
Submitted to course coordinator				Wasif Afzal
Subject	Inspection Summary Report for Code Review			
Priority	High:15	Low: 22		Medium : 40
Number of Faults Found				77
Re-Inspection Required (Y/N)				N
Total Number of Inspection meetings				5
Total Inspection time (Hrs)				10
Moderator	Author	Recorder	Reader	Inspector
Srinivas	Maanasa	Matthias	Aparna	Sudhakar

Gambar 2.5 Contoh *Review* dari *Formal Inspeciton* (Nidhra, 2012)