BAB II

LANDASAN TEORI

2.1 Sistem Informasi

Aplikasi adalah penggunaan atau penerapan suatu konsep yang menjadi suatu pokok pembahasan. Aplikasi dapat diartikan juga sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu (Noviansyah, 2008). Aplikasi *software* yang dirancang untuk suatu tugas khusus dapat dibedakan menjadi dua jenis, yaitu:

a. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
b. Aplikasi *software* paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu.

Menurut Jogiyanto (2005:12), aplikasi adalah penggunaan dalam suatu komputer, instruksi (*instructiom*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses *input* menjadi *output.*

2.2 Penjadwalan Produksi

Penjadwalan produksi dapat didefinisikan sebagai proses pengalokasian sumber daya dan mesin yang ada untuk menyelesaikan semua pekerjaan dengan mempertimbangkan batasan-batasan yang ada. Pada saat merencanakan suatu jadwal produksi, ketersediaan sumber daya yang dimiliki harus dipertimbangkan dengan baik (Nasution, 2003). Tujuan dari penjadwalan produksi adalah sebagai berikut :

1. Meningkatkan penggunaan sumber daya atau mengurangi waktu tunggunya, sehingga total waktu proses dapat berkurang dan produktivitasnya dapat meningkat.
2. Mengurangi persediaan barang setengah jadi atau mengurangi sejumlah pekerjaan yang menunggu dalam antrian ketika sumber daya yang ada masih mengerjakan tugas yang lain.
3. Mengurangi beberapa keterlambatan pada pekerjaan yang mempunyai batas waktu penyelesaian sehingga akan meminimaliasi biaya keterlambatan.
4. Membantu pengambilan keputusan mengenai perencanaan kapasitas pabrik dan jenis kapasitas yang dibutuhkan sehingga penambahan biaya dapat dihindarkan.

2.3 Istilah-Istilah dalam Penjadwalan Produksi

Beberapa istilah umum yang digunakan dalam penjadwalan produksi antara lain (Rizky, 2011):

1. *Processing time* (ti) (waktu proses), waktu yang diperlukan untuk mengerjakan suatu pekerjaan. Dalam waktu proses ini sudah termasuk waktu yang dibutuhkan untuk persiapan dan pengaturan (setup) selama proses berlangsung
2. *Due date* (di) (batas waktu), adalah batas waktu yang diberikan untuk menyelesaikan suatu tugas atau operasi terakhir dari suatu pekerjaan. Apabila tugas tersebut tidak terselesaikan hingga batas waktu, maka terjadi keterlambatan.
3. *Lateness* (Li) (keterlambatan), adalah selisih waktu penyelesaian tugas *(Completion time)* dengan batas waktunya *(Due date)*. Apabila tugas

diselesaikan sebelum batas waktu (due date) maka terjadi nilai keterlambatan positif. Apabila tugas diselesaikan setelah batas waktu, maka terjadi keterlambatan negatif.

4. *Tardiness* (Ti) (ukuran keterlambatan), adalah ukuran waktu terlambat yang bernilai positif jika suatu pekerjaan dapat diselesaikan lebih cepat dari due date.
5. *Slack* (SLi) (kelonggaran), adalah waktu tersisa yang muncul akibat dari waktu proses *(time processing)* lebih kecil dari batas waktunya *(Due date).*
6. *Completion time* (Ci) (waktu penyelesaian), adalah waktu yang dibutuhkan untuk menyelesaikan pekerjaan mulai dari saat tersedianya pekerjaan (t = 0) sampai dengan tugas ke-n selesai.
7. *Flow time* (Fi) (waktu alir), adalah waktu yang dibutuhkan oleh suatu pekerjaan dari saat pekerjaan tersebut masuk ke dalam suatu tahap proses sampai pekerjaan yang bersangkutan selesai dikerjakan. Dengan kata lain, *flow time* adalah waktu proses ditambah dengan waktu menunggu sebelum diproses.
8. *Makespan* (M), adalah total waktu penyelesaian pekerjaan-pekerjaan mulai urutan pertama yang dikerjakan pada mesin atau *work center* pertama sampai urutan pekerjaan terakhir *work center* terakhir.

2.4 Kriteria Proses Penjadwalan

Teknik penjadwalan yang benar tergantung pada volume pesanan, sifat alami operasi, dan kompleksitas pekerjaan secara keseluruhan, serta kepentingan dari keempat kriteria (Heizer & Render, 2008). Berikut keempat kriteria tersebut :

1. Meminimalkan waktu penyelesaian. Kriteria ini dievaluasi dengan menentukan waktu penyelesaian rata-rata untuk setiap pekerjaan.
2. Memaksimalkan *utilisasi*. Kriteria ini dievaluasi dengan menghitung presentase waktu suatu fasilitas yang digunakan.
3. Meminimalkan persediaan barang setengah jadi. Kriteria ini dievaluasi dengan menentukan jumlah pekerjaan rata-rata dalam sistem. Hubungan antara banyaknya pekerjaan dalam sistem dan persediaan barang setengah jadi akan tinggi. Oleh karena itu, jika terdapat lebih sedikit pekerjaan dalam sistem, maka persediaan yang ada lebih rendah.
4. Meminimalkan waktu tunggu pelanggan. Kriteria ini dievaluasi dengan menentukan jumlah keterlambatan rata-rata. Empat kriteria ini untuk mengevaluasi kinerja penjadwalan. Selain itu, pendekatan penjadwalan yang baik harus sederhana, jelas, mudah dipahami, mudah dilakukan, fleksibel, dan realistis.

2.5 Metode-metode Penjadwalan Produksi

Pembuatan penjadwalan produksi, khususnya penjadwalan produksi di pabrik biasanya dilakukannya dengan bantuan *sequencing* yaitu menentukan urutan pekerjaan yang harus dikerjakan terlebih dahulu agar setiap aktivitas produksi dapat berjalan lebih efektif dan efisien. Terdapat beberapa macam aturan dalam *sequencing* tergantung pada tujuannya (Subagyo, 2005). Metode – metode yang dapat digunakan sebagai berikut (Russell & Taylor, III 2006):

1. FCFS (*First Come First Server*)

   Metode yang memprioritaskan operasi atau pekerjaan yang datang terlebih dahulu. Pekerjaan atau proses yang datang akan langsung diproses terlebih

dahulu. Metode ini sangat cocok untuk perusahaan atau organisasi yang pelanggannya lebih mementingkan waktu pelayanan.

2. SPT (*Shortest Processing Time*)

   Metode ini memprioritaskan operasi atau pekerjaan yang waktu prosesnya terpendek. Metode ini meminimalkan *work in proces*, rata-rata keterlambatan (*mean latenes)* dan waktu penyelesaian rata-rata (*mean flow time)* produk.

3. EDD (*Earliest Due Date*)

   Metode ini memprioritaskan operasi atau pekerjaan yang batas penyelesaiannya (*Due Date*) terpendek. Metode ini berjalan baik pada pekerjaan yang waktu prosesnya relatif sama.

4. LPT (*Largest Processing Time*)

   Metode ini memprioritaskan operasi atau pekerjaan yang mempunyai waktu proses yang terlama dulu yang akan dikerjakan terlebih dahulu.

Dalam buku yang dibuat oleh Render dan Heizer (2008), tidak ada satu pun aturan pengurutan yang unggul dalam semua kriteria. Pengalaman menunjukkan hal berikut:

1. SPT biasanya merupakan teknik terbaik untuk meminimalkan aliran pekerjaan dan meminimalkan jumlah pekerjaan rata-rata dalam sistem. Kelemahan urutanya adalah pekerjaan yang memiliki waktu pemrosesan panjang dapat tidak dikerjakan secara terus-menerus, karena pekerjaan yang memiliki waktu pemrosesan pendek selalu didahulukan.
2. FCFS tidak menghasilkan kinerja yang baik pada hamper semua kriteria. Bagaimanapun, FCFS memiliki kelebihan karena terlihat adil oleh pelanggan.

3. EDD meminimalkan keterlambatan maksimal yang mungkin perlu untuk pekerjaan yang memiliki penalti setelah tanggal tertentu.

Setiap metode yang digunakan dapat menghasilkan ukuran efektifitas dengan menggunakan rumus sebagai berikut:

a. Waktu Penyelesaian Rata-Rata

$$\text{Rumus : } \frac{Jumlah\ aliran\ waktu\ total}{jumlah\ pekerjaan}$$

b. Utilisasi

$$\text{Rumus : } \frac{Jumlah\ waktu\ proses\ total}{jumlah\ aliran\ waktu\ total}$$

c. Jumlah Pekerjaan Rata-Rata Dalam Sistem

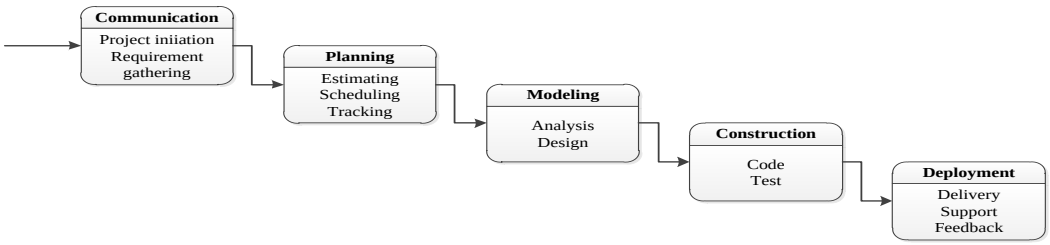
$$\text{Rumus : } \frac{Jumlah\ aliran\ waktu\ total}{waktu\ proses\ pekerjaan\ total}$$

d. Keterlambatan Rata-Rata

$$\text{Rumus : } \frac{Jumlah\ hari\ keterlambatan}{jumlah\ pekerjaan}$$

2.6 *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) adalah suatu pendekatan yang berurutan atau sistematis yang digunakan untuk pengembangan perangkat lunak. Model ini juga disebut model waterfall yaitu model air terjun yang mempunyai tahapan - tahapan yaitu *communication*, *planning*, *modeling*, *construktion*, dan *deployment*.



Gambar 2.1 *SDLC* dengan metode *Waterfall*

Pada Gambar 2.1 menggambarkan tahapan – tahapan dari model *waterfall*. Model ini berjalan secara sistematis dari tahap *communication, planning, modeling, construction,* dan *deployment.* Pressman menjelaskan tahapan-tahapan model *Waterfall* sebagai berikut (Pressman, 2015):

a. *Comunication*

Dalam model *waterfall* langkah pertama diawali dengan komunikasi dengan pihak konsumen/pengguna. Komunikasi ini adalah langkah penting karena menyangkut pengumpulan informasi tentang kebutuhan konsumen/pengguna.

b. *Planning*

Setelah proses *communication* ini, kemudian menetapkan rencana untuk pengerjaan *software* yang meliputi tugas-tugas teknis yang akan dilakukan, resiko yang mungkin terjadi, sumber yang dibutuhkan, hasil yang akan dibuat, dan jadwal pengerjaan.

c. *Modeling*

Pada proses *modeling* ini menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada rancangan struktur data, arsitektur *software*, representasi *interface*, dan *detail* (algoritma) prosedural.

d. *Construction*

Construction merupakan proses membuat kode (*code generation*). *Coding* atau pengkodean merupakan penerjemahan desain dalam bahasa yang bisa dikenali oleh komputer. *Programmer* akan menerjemahkan transaksi yang diminta oleh user. Tahapan inilah yang merupakan tahapan secara nyata

dalam mengerjakan suatu *software*, artinya penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan testing terhadap sistem yang telah dibuat. Tujuan testing adalah menemukan kesalahan-kesalahan terhadap sistem tersebut untuk kemudian bisa diperbaiki.

e. *Deployment*

Tahapan ini bisa dikatakan final dalam pembuatan sebuah *software* atau sistem. Setelah melakukan analisis, desain dan pengkodean maka sistem yang sudah jadi akan digunakan *user*. Kemudian *software* yang telah dibuat harus dilakukan pemeliharaan secara berkala.

2.7 *Visual Basic*.Net

Visual Basic terkenal sebagai bahasa pemrograman yang mudah digunakan untuk membuat aplikasi yang berjalan di atas *platform* Windows. Pada tahun 90-an, *Visual Basic* menjadi bahasa pemrograman yang paling populer dan menjadi pilihan utama untuk mengembangkan program berbasis Windows. Versi *Visual Basic* yang terakhir sebelum berjalan di atas .NET *Framework* adalah VB6 (Kurniawan, 2011).

Visual Basic.NET dirilis pada bulan Februari tahun 2002 bersamaan dengan *platform* .NET 1.0. Kini sudah ada beberapa versi dari *Visual Basic* yang berjalan pada *platform* .NET, yaitu VB 2002 (VB7), VB 2005 (VB8), VB 2008 (VB9), dan VB 2010 (VB10) yang dirilis bersamaan dengan *Visual Studio* 2010. Selain *Visual Basic* 2010, *Visual Studio* 2010 juga mendukung beberapa bahasa lain yaitu C#, C++, F# (bahasa baru untuk functional programming), IronPhyton, dan IronRuby (bahasa baru untuk *dynamic programming*).

2.8 *MyStructured Query Language* (MySql)

My Structured Query Language (MySQL) atau yang bisa di baca mai-sekuel adalah program pembuat dan pengelola *database* (Kadir, 2003). Selain itu data MySQL juga merupakan program pengakses *database* yang bersifat jaringan, sehingga dapat digunakan untuk membuat Aplikasi Multi *User* (banyak pengguna). Kelebihan dari MySQL adalah menggunakan bahasa *query* (permintaan) standar SQL (*Structured Query Language*). SQL adalah salah satu bahasa permintaan yang terstruktur. Dalam hal ini penulis menggunakan *database* MySQL dikarenakan *database* yang sudah ada pada pengadilan tinggi Surabaya menggunakan MySQL.

2.9 *Testing*

Testing adalah proses pemantapan kepercayaan akan kinerja program atau sistem sebagaimana yang diharapkan. *Testing Software* adalah proses pengoperasikan *software* dalam suatu kondisi yang dikendalikan untuk verifikasi, mendeteksi *error* dan validasi. Verifikasi adalah pengecekkan atau pengetesan entitas-entitas, termasuk *software*, untuk pemenuhan dan konsistensi dengan melakukan evaluasi hasil terhadap kebutuhan yang telah ditetapkan. Validasi adalah melihat kebenaran sistem apakah proses yang telah dituliskan sudah sesuai dengan apa yang dibutuhkan oleh pengguna. Deteksi *error* adalah testing yang berorientasi untuk membuat kesalahan secara intensif, untuk menentukan apakah suatu hal tersebut tidak terjadi. *Test case* merupakan suatu tes yang dilakukan berdasarkan pada suatu inisialisasi, masukan, kondisi ataupun hasil yang telah ditentukan sebelumnya (Romeo, 2003). Adapun kegunaan dari *test case* ini, adalah sebagai berikut:

1. Untuk melakukan *testing* kesesuaian suatu komponen terhadap desain *White Box Testing*.
2. Untuk melakukan *testing* kesesuaian suatu komponen terhadap spesifikasi *Black Box Testing*.

2.9.1 *White Box Testing*

White Box Testing adalah suatu metode desain *test case* yang menggunakan struktur kendali dari desain prosedural (Romeo, 2003). Seringkali *white box testing* di asosiasi kan dengan pengukuran cakupan tes, yang mengukur persentase jalur-jalir dari tipe yang dipilih untuk dieksekusi oleh test cases. *White box testing* dapat menjamin semua struktur internal data dapat dites untuk memastikan validasinya.

Cakupan pernyataan, cabang dan jalur adalah suatu teknik *white box testing* yang menggunakan alur logika dari program untuk membuat *test case* alur logika adalah cara dimana suatu bagian dari program tertentu dieksekusi saat menjalankan program. Alur logika suatu program dapat di representasi kan dengan *flowgraph*.

2.9.2 *Black Box Testing*

Pengertian dari *Black Box Testing* adalah suatu tipe *testing* yang memperlakukan perangkat lunak yang tidak diketahui kinerja internalnya. Berdasarkan hal tersebut, para *tester* memandang perangkat lunak seperti layaknya "kotak hitam" yang tidak terlihat isinya, tetapi mendapat proses *testing* bagian luarnya saja. *Black Box Testing* hanya memandang perangkat lunak dari sisi spesifikasi dan kebutuhan yang telah ditentukan pada awal perancangan. Keuntungan dari jenis *testing* ini antara lain (Romeo, 2003):

1. Anggota tim *tester* tidak harus darri seseorang yang memiliki kemampuan teknis program.

2. Kesalahan dari perangkat lunak ataupun *bug* sering ditemukan oleh komponen *tester* yang berasal dari pengguna.
3. Hasil dari *black box testing* dapat memperjelas kontradiksi ataupun kerancuan yang mungkin timbul dari eksekusi sebuah perangkat lunak.
4. Proses *testing* dapat dilakukan lebih cepat dibandingkan *white box testing.*
5. Alasan penulis menggunakan *black box testing* dikarenakan dapat dengan mudah mengetahui fungsi aplikasi yang salah, dapat dengan mudah mengetahui kesalahan pada tampilan, dapat dengan mudah mengetahui kesalahan pada *database*, dan dapat dengan mudah mengetahui kesalah inisialisasi dan tujuan akhir.