BAB II

LANDASAN TEORI

2.1 Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu. (Jogiyanto, 2005)

Pendekatan sistem yang merupakan jaringan kerja dari prosedur lebih menekankan urut-urutan operasi di dalam sistem. Menurut Richard F. Neuschel (Jogiyanto, 2005),” Prosedur adalah sutu urut-urutan operasi klerikal (tulis menulis) yang biasanya melibatkan beberapa orang didalam satu atau lebih departemen, yang diterapkan untuk menjamin penanganan yang seragam dari transaksi-transaksi bisnis yang terjadi”.

Suatu sistem mempunyai tujuan (*goal*) atau sasaran (objektifitas). Tujuan biasanya dihubungkan dengan ruang lingkup yang lebih luas dan sasaran dalam ruang lingkup yang lebih sempit. Sasaran menentukan masukan dan keluaran yang dihasilkan. Sistem dikatakan berhasil jika mencapai suatu sasaran dan tujuan.

2.2 Aplikasi

Aplikasi adalah perangkat lunak yang ada pada komputer digunakan untuk melayani berbagai macam kebutuhan. Teknologi yang canggih dari perangkat keras akan berfungsi bila instruksi-instruksi tertentu telah diberikan kepadanya. Instruksi-instruksi tersebut disebut dengan perangkat lunak (Jogiyanto, 2005).

Definisi lain aplikasi adalah program yang dibuat oleh pemakai yang ditujukan untuk melakukan suatu tugas khusus. Program seperti ini biasa

dikelompokkan menjadi dua yaitu program aplikasi serbaguna dan program aplikasi spesifik.

Program aplikasi serbaguna adalah program aplikasi yang dapat digunakan oleh pemakai untuk melaksanakan hal-hal yang bersifat umum (misalnya untuk membuat dokumen atau untuk pengiriman surat secara elektronis) serta untuk mengotomasikan tugas-tugas individual yang bersifat berulang (misalnya untuk melakkan perhitungan yang bersifat rutin). Termasuk dalam kategori ini antara lain adalah DBMS sederhana, *Web browser*, surat elektronis, pengolah kata (*word processor*), dan lembar kerja (*spreadsheet*). Program aplikasi serbaguna seringkali disebut perangkat lunak pemakai akhir (*end-user-software*).

Program aplikasi spesifik adalah program yang ditujukan untuk menangani hal-hal yang sangat spesifik. Misalnya, program pada sistem POS (*point-of-sale*) dan ATM. Termasuk dalam kategori ini adalah program yang disebut sebagai paket aplikasi atau perangkat lunak paket (Kadir, 2014).

2.3 *Material Requirement Planning* (MRP)

2.3.1 Definisi *Material Requirement Planning* (MRP)

Menurut Indrajit (2007) bahwa *Material Requirement Planning* (MRP) adalah teknik penjadwalan yang digunakan oleh perusahaan manufaktur sebagai sarana bagaimana setiap pekerja yang terkait melakukan komunikasi perihal aliran material atau barang. Teknik atau metode MRP menitikberatkan pada perencanaan, karena pada dasarnya MRP adalah teknik perencanaan dan penjadwalan. Teknik ini sebetulnya sangat sederhana yaitu sekedar menggunakan logika matematik untuk merencanakan jumlah barang yang diperlukan dan menjadwalkan kapan barang dimaksud diperlukan. Meskipun sangat sederhana tetapi dari praktek diketahui

bahwa justru karena perencanaan dan penjadwalan inilah sering kali suatu proses produksi atau manufaktur itu dapat berhasil atau tidak. Perencanaan dengan MRP adalah tipikal perencanaan dan penjadwalan yang digunakan dalam suatu perusahaan manufaktur yang mengenai alur barang ke dan melalui proses pembuatan barang jadi.

2.3.2 *Input* dari Sistem MRP

Menurut Indrajit (2007) bahwa masukan adalah sesuatu yang dibutuhkan oleh proses perencanaan produksi agar suatu keluaran dapat dihasilkan. Ada beberapa inputan yang dibutuhkan untuk proses *Material Requirement Planning* (MRP) diantaranya adalah sebagain berikut:

1. *Master Production Schedule* (MPS)

Master Production Schedule atau jadwal induk produksi merupakan proses alokasi untuk membuat sejumlah produk yang diinginkan dengan memperhatikan kapasitas yang dimiliki. MPS didasarkan pada peramalan atas kebutuhan permintaan dependent dari setiap produk akhir yang akan dibuat.

2. Status Inventori (SI)

Status inventori berisikan informasi tentang status semua item yang ada dalam persediaan. Setiap item persediaan harus didefinisikan untuk menjaga kekeliruan perencanaan. Pencatatan-pencatatan ini harus dijaga agar selalu menggambarkan keadaan yang paling akhir dengan selalu melakukan pencatatan tentang transaksi-transaksi yang terjadi, seperti penerimaan, pengeluaran produk gagal, *lead time*, persediaan cadangan, dan catatan-catatan penting lainnya dari semua item.

3. *Bill of Material* (BoM)

Bill of Material berisikan informasi tentang hubungan komponen satu dengan yang lainnya dalam suatu perakitan, juga menginformasikan kebutuhan tiap komponen untuk membentuk setiap produk akhir. Informasi ini sangat penting dalam penentuan kebutuhan kotor dan kebutuhan bersih suatu produk akhir.

2.3.3 Proses dari Sistem MRP

Menurut Indrajit (2007) bahwa proses adalah cara atau dengan apa masukan itu dirubah menjadi keluaran. Untuk mengelola masukan tersebut dibutuhkan suatu alat atau metode untuk membentuk *output* sesuai yang diinginkan dalam hal ini metode yang diguanakan adalah *Material Requirement Planning* (MRP).

2.3.4 *Output* dari Sistem MRP

Menurut Indrajit (2007) bahwa keluaran adalah hasil dari masukan dan diproses yang akan menghasilkan suatu produk atau aplikasi yang dapat membantu berjalannya suatu sistem informasi perencanaan produksi. *Output* yang dihasilkan yaitu solusi perencanaan produksi berupa informasi perencanaan produksi atau jumlah kebutuhan bahan baku yang akan diproduksi yang didukung oleh *Bill of Material* (BoM), pelaksanaan produksi atau kebutuhan waktu dan sumber daya manusia (SDM) dalam pelaksanaan produksi yang didukung oleh MPS, dan Status Inventori (SI) barang untuk diproduksi.

2.4 Diagram *Unified Modeling Language (UML)*

2.4.1 Diagram *Use Case* Bisnis

Diagram *use case* bisnis digunakan untuk mempresentasikan bisnis yang dilakukan organisasi. Diagram ini menjawab pertanyaan: “apa yang bisnis lakukan?’ dan “mengapa kita membangun sistem untuk itu?’. Diagram *use case*

bisnis digunakan untuk memodelkan aktivitas bisnis organisasi sebagai landasan pembuatan *use case* sistem. Diagram *use case* bisnis juga digunakan untuk mendefinisikan bisnis apa saja yang dilakukan organisasi dalam rangka menjalankan visi organisasi (Sholiq, 2010).

Diagram *use case* bisnis digambarkan menurut perspektif organisasi. Ia tidak membedakan apakah aktivitas tersebut dilakukan secara manual atau otomatis menggunakan perangkat lunak. Komponen pembentuk diagram *use case* bisnis:

a. Aktor Bisnis

Aktor bisnis adalah sesorang atau sesuatu yang ada di luar organisasi. Ia berinteraksi dengan organisasi dan terlibat dalam kegiatan bisnis organisasi. Contoh aktor bisnis, antara lain: pelanggan, kreditor, investor atau pemasok. Jadi posisi mereka di luar organisasi yang sedang dimodelkan, tetapi terlibat dalam kegiatan organisasi. Aktor bisnis dimodelkan dengan menggunakan ikon berikut:

Gambar 2.1 Notasi Aktor Bisnis

b. Pekerja Bisnis

Pekerja bisnis adalah suatu peran (*role*) di dalam organisasi, bukan posisi atau jabatan. Seseorang bisa memainkan banyak peran tetapi memegang hanya satu posisi.

Gambar 2.2 Notasi Pekerja Bisnis

Memodelkan pekerja bisnis digunakan untuk memahami peran di dalam aktivitas bisnis organiasi dan bagaimana peran tersebut berinteraksi dengan proses bisnis organisasi. Sangat jelas perbedaan antara aktor bisnis dan pekerja bisnis, jika aktor bisnis berada di luar ruang lingkup bisnis yang sedang didefinisikan, sedangkan pekerja bisnis berada di dalam bisnis yang didefinisikan.

c. *Use Case* Bisnis

Sebuah *use case* bisnis adalah model yang digunakan untuk menggambarkan sebuah proses bisnis organisasi. Dengan kata lain, *use case* bisnis menginformasikan tentang aktivitas bisnis utama yang organisasi lakukan.

Gambar 2.3 Notasi *Use Case*

Use case bisnis memiliki format penulisan nama kata kerja atau frase kata kerja, seperti pada kata "Memberi harga produk'. Tentu saja, *use case* bisnis "Memberi harga produk' tidak memberikan banyak informasi tanpa adanya bberaoa penjelasan tambahan. Setiap *use case* bisnis, dapat dibuat penjelasan tambahan untuk menjelaskan secara rinci apa yang terjadi di dalam *use case* bisnis. Untuk hal tersebut didokumentasikan secara spesifik di dalam sebuat *workflow*.

d. Relasi

Untuk membuat diagram *use case* bisnis digunakan penghubung (relasi) antara aktor bisnis dan atau pekerja bisnis dengan *use case* bisnis. Ada dua jenis relasi yang mungkin terjadi, pertama adalah relasi antara aktor bisnis atau pekerja bisnis dengan *use case* bisnis, relasi jenis ini disebut asosiasi. Relasi asosiasi adalah relasi regular yang sering atau hampir terjadi pada kegiatan pemodelan bisnis.

Kedua, relasi pewarisan struktur antara elemen-elemen pemodelan bisnis sendiri yang disebut generalisasi.

e. Entitas Bisnis

Entitas bisnis adalah objek digunakan atau yang dihasilkan oleh organisasi saat melakukan aktivitas bisnis. Entitas bisnis meliputi sesuatu yang pekerja bisnis hadapi sehari-hari. Setiap entitas bisnis harus diberi nama yang unik yang menggambarkan tanggung jawabnya. Nama berbentuk kata benda atau frase kata benda.

Gambar 2.4 Notasi Entitas Bisnis

2.4.2 Diagram *Use Case*

Diagram *use case* menyajikan interaksi antara *use case* dan actor dalam sistem yang akan dikembangkan. *Use case* sendiri adalah fungsionalitas atau persyaratan-persyaratan sistem yang harus dipenuhi oleh sistem yang akan dikembangkan tersebut menurut pandangan pemakai sistem. Sedangkan aktor bisa berupa orang, peralatan, atau sistem lain yang berinteraksi terhadap sistem yang akan dibangun (Sholiq, 2010).

2.4.3 Diagram Aktivitas

Diagram aktivitas menggambarkan aliran fungsionalitas sistem. Ada dua kegunaan diagram aktivitas dalam pemodelan dengan UML, yaitu:

1. Pada tahap pemodelan bisnis, diagram aktivitas dapat digunakan untuk menunjukan alur kerja bisnis (*business workflow*).

2. Pada tahap pemodelan sistem, diagram aktivitas dapat digunakan untuk menjelaskan aktivitas yang terjadi didalam sebuah use case.

Diagram aktivitas mendefinisikan darimana *workflow* dimulai, dimana *workflow* berakhir, aktivitas apa saja yang terjadi di dalam *workflow*, dan apa saja yang dilakukan saat aktivitas terjadi. Aktivitas adalah tugas yang dilakukan selama dalam *workflow*.

2.4.4 Diagram Sekuensial

Diagram sekuensial digunakan untuk menunjukkan alur (*flow*) fungsionalitas yang melalui sebuah *use case* yang disusun dalam urutan waktu.

2.4.5 Diagram Kelas

Diagram kelas menunjukkan interaksi antar kelas-kelas dalam sistem. Kelas juga dapat dianggap sebagai cetak biru dari objek-objek di dalam sistem.

2.5 MySQL

MySQL adalah sebuah perangkat lunak *database* (basis data) sistem terbuka yang sangat terkenal di kalangan pengembang sistem *database* dunia yang digunakan untuk berbagai aplikasi terutama untuk aplikasi berbasis web. MySQL mempunyai fungsi sebagai SQL (*Structured Query Language*) telah diperluas. MySQL umumnya digunakan bersama dengan PHP untuk membuat aplikasi yang dinamis dan powerful.

2.6 PHP (*Hypertext Preprocessor*)

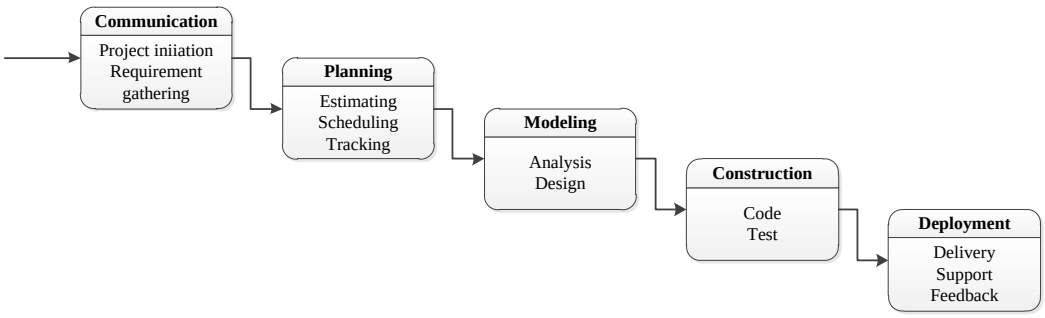
Menurut Saputra (2011) PHP atau yang memiliki kepanjangan PHP *Hypertext Preprocessor* merupakan suatu bahasa pemrograman yang difungsikan untuk membangun suatu website dinamis. PHP menyatu dengan kode HTML, maksudnya adalah beda kondisi. HTML digunakan sebagai pembangun atau

pondasi dari kerangka layout web, sedangkan PHP difungsikan sebagai prosesnya sehingga dengan adanya PHP tersebut, web akan sangat mudah di-*maintenance*. PHP berjalan pada sisi *server* sehingga PHP disebut juga sebagai bahasa *server Side Scripting*. Artinya bahwa dalam setiap menjalankan PHP, wajib adanya web server.

2.7 Siklus Hidup Pengembangan Sistem (SDLC)

Menurut pressman (2015), Model *System Development Life Cycle (SDLC)* ini biasa disebut juga dengan model *waterfall* atau disebut juga *classic life cycle.* Adapun pengertian dari SDLC ini adalah suatu pendekatan yang sistematis dan berurutan. Tahapan-tahapannya adalah *Requirements* (analisis sistem), *Analysis* (analisis kebutuhan sistem), *Design* (perancangan), *Coding* (implementasi), *Testing* (pengujian) dan *Maintenance* (perawatan).

Model eksplisit pertama dari proses pengembangan perangkat lunak, berasal dari proses-proses rekayasa yang lain. Model ini memungkinkan proses pengembangan lebih terlihat. Hal ini dikarenakan bentuknya yang bertingkat ke bawah dari satu fase ke fase lainnya, model ini dikenal dengan model *waterfall*, seperti pada gambar 2.5



Gambar 2.5 Bagan *System Development Life Cycle* (SDLC) Model *Waterfall*

Gambar 2.5 menunjukkan tahapan umum dari model proses *waterfall*. Model ini disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus

menunggu selesainya tahap sebelumnya dan berjalan berurutan. Akan tetapi, Pressman (2015) memecah model ini meskipun secara garis besar sama dengan tahapan-tahapan model *waterfall* pada umumnya.

Model ini merupakan model yang paling banyak dipakai dalam *Software Engineering*. Model ini melakukan pendekatan secara sistematis dan urut mulai dari level kebutuhan sistem lalu menuju ke tahap *Communication, Planning, Modeling, Construction,* dan *Deployment.*

Berikut ini adalah penjelasan dari tahap-tahap yang dilakukan di dalam Model *Waterfall* menurut Pressman (2015):

a. *Communication*

Langkah pertama diawali dengan komunikasi kepada konsumen/pengguna. Langkah awal ini merupakan 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, risiko 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.