BAB III

OBJEK DAN METODE PENELITIAN

3.1 Objek Penelitian

3.1.1 Sejarah Singkat Perusahaan

Pada tanggal 24 Juni 1859 di kota Solferino, Italia Utara, pasukan Perancis dan Italia sedang bertempur melawan pasukan Austria dalam suatu peperangan yang mengerikan. Pada hari yang sama, seorang pemuda warganegara Swiss, Henry Dunant , berada di sana dalam rangka perjalanannya untuk menjumpai Kaisar Perancis, Napoleon III. Puluhan ribu tentara terluka, sementara bantuan medis militer tidak cukup untuk merawat 40.000 orang yang menjadi korban pertempuran tersebut. Tergetar oleh penderitaan tentara yang terluka, Henry Dunant bekerjasama dengan penduduk setempat, segera bertindak mengerahkan bantuan untuk menolong mereka.

Beberapa waktu kemudian, setelah kembali ke Swiss, dia menuangkan kesan dan pengalaman tersebut kedalam sebuah buku berjudul "Kenangan dari Solferino", yang menggemparkan seluruh Eropa. Dalam bukunya, Henry Dunant mengajukan dua gagasan;

1. Membentuk organisasi kemanusiaan internasional, yang dapat dipersiapkan pendiriannya pada masa damai untuk menolong para prajurit yang cedera di medan perang.

2. Mengadakan perjanjian internasional guna melindungi prajurit yang cedera di medan perang serta perlindungan sukarelawan dan organisasi tersebut pada waktu memberikan pertolongan pada saat perang.

Pada tahun 1863, empat orang warga kota Jenewa bergabung dengan Henry Dunant untuk mengembangkan gagasan pertama tersebut. Mereka bersama-sama membentuk "Komite Internasional untuk bantuan para tentara yang cedera", yang sekarang disebut Komite Internasional Palang Merah atau International Committee of the Red Cross (ICRC).Dalam perkembangannya kelak untuk melaksanakan kegiatan kemanusiaan di setiap negara maka didirikanlah organisasi sukarelawan yang bertugas untuk membantu bagian medis angkatan darat pada waktu perang. Organisasi tersebut yang sekarang disebut Perhimpunan Nasional Palang Merah atau Bulan Sabit Merah.

Berdasarkan gagasan kedua, pada tahun 1864, atas prakarsa pemerintah federal Swiss diadakan Konferensi Internasional yang dihadiri beberapa negara untuk menyetujui adanya "Konvensi perbaikan kondisi prajurit yang cedera di medan perang". Konvensi ini kemudian disempurnakan dan dikembangkan menjadi Konvensi Jenewa I, II, III dan IV tahun 1949 atau juga dikenal sebagai Konvensi Palang Merah . Konvensi ini merupakan salah satu komponen dari Hukum Perikemanusiaan Internasional (HPI) suatu ketentuan internasional yang mengatur perlindungan dan bantuan korban perang.

3.1.2 Visi dan Misi Perusahaan

3.1.2.1 Visi

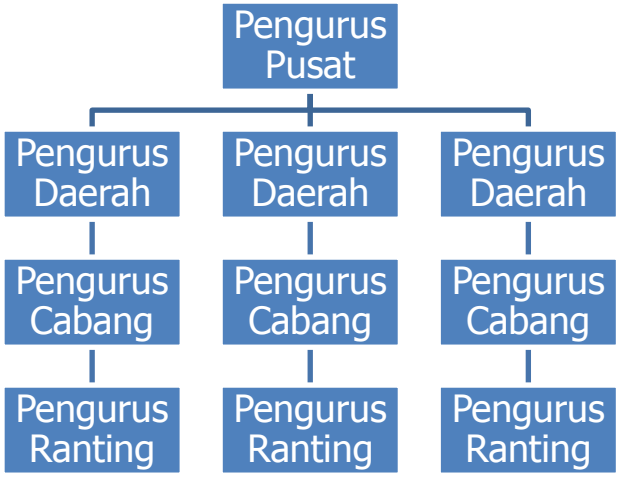
Mampu dan siap menyediakan pelayanan kepalangmerahan dengan cepat dan tepat dengan berpegang teguh pada Prinsip Dasar Gerakan Palang Merah dan Bulan Sabit Merah Internasional

3.1.2.2 Misi

1. Menyebarluaskan dan mendorong aplikasi prinsip dasar Gerakan Palang Merah dan Bulan Sabit Merah Internasional
2. Melaksanakan tugas-tugas :
    - Penanganan bencana yang mencakup tanggap darurat dan kesiapsiagaan berbasis masyarakat dan "TMS"
    - Pelayanan kesehatan dan Program Bantuan kesehatan berbasis masyarakat
3. Pengembangan dan Penguatan kapasitas organisasi dan mewujudkan program yang berkesinambungan.

3.1.3 Struktur Organisasi Perusahaan

Dapat kita lihat struktur organisasi yang ada di PMI diantaranya adalah:



- Pengurus Pusat
- Pengurus Daerah (31 di tingkat Propinsi)
- Pengurus Cabang (370 di tingkat Kabupaten/kotamadya)
- Pengurus Ranting (2200, di tingkat Kecamatan)

Gambar 3.1 Struktur Organisasi

3.2 Metode Penelitian

Metode penelitian merupakan cara penelitian yang digunakan untuk mendapatkan data selengkap-lengkapnya untuk mencapai tujuan tertentu.

3.2.1 Desain Penelitian

Dalam melakukan suatu penelitian perlu dilakukan perencanaan penelitian agar penelitian yang dilakukan dapat berjalan dengan baik dan sistematis. Metode yang akan digunakan dalam penelitian ini adalah metode *deskriptif*. Metode deskriptif adalah suatu metode dalam meneliti status sekelompok manusia, suatu

objek, suatu set kondisi, suatu sistem pemikiran, ataupun suatu kelas peristiwa pada masa sekarang. Tujuan dari penelitian deskriptif ini adalah untuk menjelaskan aspek-aspek yang relevan tentang fenomena perseorangan, organisasi, perusahaan berorientasi industri atau perspektif lainnya, membuat deskripsi, gambaran atau lukisan secara sistematis, faktual dan akurat mengenai fakta-fakta, sifat-sifat serta hubungan antara fenomena yang diselidiki.

3.2.2 Jenis dan Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan penulis dalam penelitian ini yaitu menggunakan sumber data primer dan sumber data sekunder, berikut penjelasannya :

3.2.2.1 Sumber Data Primer

Sumber data primer adalah informasi yang diperoleh pertama kali oleh peneliti menyangkut variabel yang menjadi tujuan utama penelitian, seperti wawancara dan observasi langsung pada objek yang diteliti dalam hal ini adalah **PMI**.

1. Observasi

Observasi pengumpulan data melalui pengamatan langsung atau peninjauan secara cermat dan langsung di lapangan atau lokasi penelitian. Dengan cara mengunjungi lokasi penelitian untuk mengamati langsung berbagai hal atau kondisi yang ada di PMI

2. Wawancara

Merupakan salah satu metode pengumpulan data dimana peneliti bertemu langsung (tatap muka) atau tidak langsung yaitu kepada bagian Dokumentasi PMI.

3.2.2.2 Sumber Data Sekunder

Data sekunder adalah data yang digunakan untuk mendukung data primer. Data tersebut dapat kita gunakan sebagai pemicu untuk memahami persoalan yang muncul dalam perusahaan tersebut dan yang akan kita gunakan sebagai masalah penelitian. Data sekunder yang penulis ambil yaitu profil perusahaan, struktur organisasi, sejarah **PMI** serta dokumen-dokumen yang diperoleh dari tempat penelitian.

3.2.3 Metode Pendekatan dan Pengembangan Sistem

Metode pendekatan dan pengembangan sistem menggambarkan tahapan-tahapan dalam proses penelitian guna memecahkan masalah penelitian dari awal perencanaan hingga tercapainya tujuan penelitian dan pengembangan system.

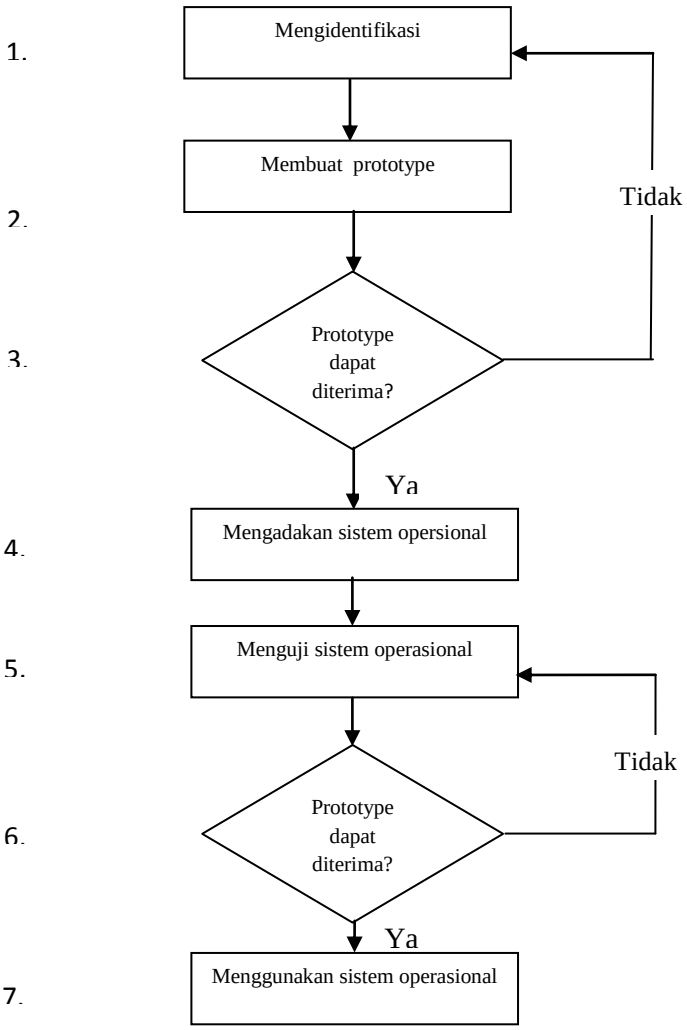
3.2.3.1 Metode Pendekatan Sistem

Metode pendekatan sistem yang penulis gunakan dalam perancangan sistem adalah metode pendekatan sistem yang berorientasi objek (*Object-Oriented),* yang mencakup analisis dan desain yang disebut *OOAD* (*Object Oriented Analysis and Design*). *Object-oriented analysis* adalah metode analisis yang memeriksa *requirements* (syarat/keperluan yang harus dipenuhi suatu sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang

lingkup permasalahan. Sedangkan *Object-oriented analysis* adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau subsistem. Alat bantu analisis dan perancangan berorientasi objek adalah *Unified Modelling Language (UML).* UML adalah sebuah bahasa yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem dengan berfokus pada objek. Berikut ini adalah delapan jenis diagram didalam UML yaitu : Use case diagram, Class diagram, Statechart diagram Activity diagram, Sequence diagram, Collaboration diagram, Component diagram, Deployment diagram.

3.2.3.1 Metode Pengembangan Sistem

Untuk metodelogi pengembangan sistem, penulis menggunakan pendekatan prototype (Prototyping Paradigma) karena metode ini terdiri dari tahap-tahap yang memberikan kemudahan, jika pada satu tahap tidak sesuai maka dapat kembali ke tahap sebelumnya serta pengujian dilakukan oleh pembuat atau programmer. Pendekatan Prototype dapat dilihat pada gambar dibawah ini :



Gambar 3.2 Pendekatan Prototype

Sumber : Jogianto, 1990

3.2.3.3 Alat Bantu Analisis dan Perancangan

Dengan metode pendekatan system yang berorientasi objek, maka penulis akan menggambarkan bagaimana karakteristik system tersebut dengan menggunakan pemodelan yang disebut *Unifield Modelling Language* (UML).

1. Use Case Diagram

Use case menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar dan menjelaskan sistem

secara fungsional yang terlihat user. Biasanya dibuat pada awal pengembangan. *Use case* diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case*merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua feature yang ada pada sistem. Sebuah *use case* dapat meng-include fungsionalitas use case lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-include akan dipanggil setiap kali *use case* yang meng-include dieksekusi secara normal. Sebuah *use case* dapat di-include oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common. Sebuah *use case* juga dapat meng-extend use case lain dengan behaviour-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2. Activity diagram

Diagram ini menjelaskan alur kerja suatu sistem. *Activity diagram* mirip dengan *state diagram* karena sejumlah aktifitas menggambarkan keadaan suatu proses dengan memperlihatkan urutan aktifitas yang dijalankan baik berupa

pilihan maupun paralel. Diagram ini juga berguna untuk menganalisis sebuah *use case* dengan menggambarkan aksi-aksi yang diperlukan dan kapan aksi-aksi tersebut dijalankan. Selain itu, *activity diagram* dapat menjelaskan urutan algoritma yang kompleks dan memodelkan sejumlah aplikasi dengan proses paralel.

3. Collaboration Diagram

Secara fungsional digram ini hampir mirip dengan *sequence diagram*. *Collaboration diagram* memfokuskan pada interaksi dan hubungan diantara sekumpulan objek yang berkolaborasi. Hubungan-hubungan tersebut memperlihatkan objek actual dan relasi yng terjadi diantara mereka yang digambarkan dengan sebuah garis. Diatas garis terdapat alur pesan yang dikirim objek yang berhubungan tersebut.

4. Class Diagram

Class diagram menggambarkan struktur dan penjelasan class, paket, dan objek serta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. Class diagram juga menjelaskan hubungan antar class dalam sebuah sistem yang sedang dibuat dan bagaimana caranya agar mereka saling berkolaborasi untuk mencapai sebuah tujuan.

Class juga memiliki 3 area pokok (utama) yaitu : nama,atribut,dan operasi. Nama berfungsi untuk member identitas pada sebuah kelas, atribut fungsinya adalah untuk member karakteristik pada data yang dimiliki suatu objek di dalam kelas, sedangkan operasi fungsinya adalah memberikan sebuah fungsi ke sebuah objek. Dalam mendefinisikan metode yang ada di dalam kelas harus diperhatikan

yang namanya Cohesion dan Coupling, Cohesion adalah ukuran keterkaitan sebuah instruksi di sebuah metode, Coupling adalah ukuran keterkaitan antar metode. Di dalam class diagram terdapat hubungan antar kelas secara konseptual, yang disebut Relasi antar Class, di UML disediakan macam-macam relasi antar Class, diantaranya: Asosiasi (Hubungan statis antar kelas), Agregasi (hubungan dari keseluruhan objek), Generalisasi (relasi beberapa subkelas ke super kelas), Dependency (keterhubungan tiap kelas.)

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. *Public*, dapat dipanggil oleh siapa saja

Hubungan Antar *Class:*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan

4. menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
5. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

Langkah – langkah untuk membuat visualisasi system dari sebuah perusahaan seperti gambar diatas yaitu :

1. Tentukan terlebih dahulu class–class yang dibutuhkan dalam visualisasi ini class–class yang dibutuhkan diantaranya pegawai, department, transaksi, pelanggan, produk, perlengkapan, makanan dan minuman.
2. Lalu langkah selanjutnya menentukan atribut-atribut yang dibutuhkan dan sesuai untuk masing-masing class. Sebagai contoh untuk class pelanggan memiliki atribut-atribut seperti id_pelanggan, nama_pelanggan dan alamat_perusahaan.

5. Sequence Diagram

Sequence diagram menggambarkan interaksi antar obyek dan mengindikasikan komunikasi diantara obyek-obyek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.

Pada diagram ini, dimensi vertikal merepresentasikan waktu. Bagian paling atas dari diagram menjadi titik awal dan waktu berjalan ke bawah sampai dengan bagian dasar dari diagram. Garis Vertical, disebutlifeline, dilekatkan pada setiap obyek atau aktor. Kemudian lifeline tersebut digambarkan menjadi kotak ketika obyek melakukan suatu operasi, kotak tersebut disebut activation. Obyek dikatakan mempunyai live activation pada saat tersebut. Pesan yang dipertukarkan antar obyek digambarkan sebagai sebuah anak panah antara activation box pengirim dan penerima. Kemudian diatasnya diberikan label pesan.

1. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

2. Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen peranti lunak, termasuk ketergantungan (*dependency*) diantaranya.

1. Komponen peranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time* maupun *run time*.
2. Pada umumnya komponen terbentuk dari bebrapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil.

3. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

Tipe-tipe Component :

Bentuk-bentuk *component* ada 3 yaitu:

1. *Deployment Component*: Yang menjadi basis dari *executable system*. Contoh *deployment component* diantaranya: dan lain-lain (*Dynamic Library Link*) *file exe, Active X Control, Java Bean* dan lain-lain.
2. *Work Product Component*: Yaitu *file-file* yang dibutuhkan untuk pembuatan *deployment component*. Contoh untuk *component* kedua ini diantaranya *file* data, *file source code* dan lain-lain.
3. *Execution Component*: Yang dibuat sebagai hasil dari sistem yang akan dijalankan.
4. Menurut Fowler(2004) hal penting pada *component* adalah *component* mewakili potongan-potongan yang independen yang bisa dipesan dan diperbaharui sewaktu-waktu. Dengan demikian, pembagian sistem kedalam *component-component* lebih banyak didorong oleh kepentingan marketing dari pada kepentingan teknis. Meskipun demikian harus juga diingat bahwa terlalu banyak *component* juga kurang bagus, karena susah mengatur dan memeliharanya khususnya yang menyangkut masalah versioning.

8. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya.

3.2.4 Pengujian Software

Metode pengujian software adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan Perangkat lunak dapat diuji dengan dua cara, yaitu :

1. Pengujian dengan menggunakan data uji untuk menguji semua elemen program (data internal,loop, logika, keputusan dan jalur). Data uji dibangkitkan dengan mengetahui struktur internal (kode sumber) dari perangkat lunak.
2. Pengujian dilakukan dengan mengeksekusi data uji dan mengecek apakah fungsional perangkat lunak bekerja dengan baik. Data uji dibangkitkan dari spesifikasi perangkat lunak.

Faktor pengujian software yang digunakan dalam penelitian ini adalah *black box*. Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian *black box* merupakan metode peracangan data uji yang didasarkan pada spesifikasi perangkat lunak.

Faktor Pengujian Black Box :

1. *Reliability*

   Menekankan bahwa aplikasi akan dilaksanakan dalam fungsi sesuai yang diminta dalam periode waktu tertentu. Pembetulan proses tersangkut kemampuan sistem untuk memvalidasi proses secara benar.

2. *Access control*

   Menekankan sumberdaya sistem harus dilindungi dari kemungkinan modifikasi, pengrusakan, penyalahgunaan dan prosedur keamanan harus dijalankan secara penuh untuk menjamin integritas data dan program aplikasi.

3. *Correctness*

   Menjamin pada data yang dimasukkan, proses dan output yang dihasilkan dari aplikasi harus akurat dan lengkap. Kelengkapan dan akurasi akan dicapai melalui kontrol transaksi dan elemen data.