

Sistem Informasi Persediaan Sparepart Kapal

Dina Angraini dan Widiastuti

Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma
 Jl. Margonda Raya no. 100, Depok Indonesia
 email : dina_anggraini@staff.gunadarma.ac.id, widiastuti@staff.gunadarma.ac.id

ABSTRACT

This Inventory Information System was created with the main objective to facilitate the people concerned with the ship spare parts inventory decisions. This system can also be used to avoid the possibility of misuse of goods. Inventory Information System is needed to know the amount and type of ship spare parts quickly and accurately as well as ease in making the report. The process of making begins with needs analysis system, making users level then designing the system using tools Unified Modeling Language (UML) and software used for making inventory information system application is VB.Net and MySql. Concludes with an application implementing information systems ship spare parts inventory. Ship Spare Parts Inventory Information System generates the management of incoming and outgoing goods in more detail to the name and quantity of goods. Application of this system accelerates the processing of inventory data that will facilitate in making the report goods in and goods out reports in a reasonably short period.

Key words

Inventory, Information System, Spare parts, UML

1. Pendahuluan

Pada saat ini banyak hal yang memerlukan perubahan untuk mencapai keadaan yang lebih baik dari sebelumnya. Perubahan itu sangat didorong oleh perkembangan teknologi komputer. Sebelum adanya perkembangan komputer semua pencatatan transaksi masih dilakukan secara manual, yang menyebabkan ketidak efisienan waktu dan membutuhkan banyak kertas untuk pencatatan yang berbeda-beda.

Demikian juga yang dialami oleh perusahaan spare part kapal yang masih mengalami banyak hambatan karena pencatatan, dalam hal ini oleh bagian gudang, masih dilakukan secara manual. Kendala utamanya adalah segala pencatatan transaksi yang terjadi harus ditulis berulang kali sebagai back up data, apabila kertas transaksi suatu hari hilang. Selain itu juga, waktu yang dibutuhkan jadi lebih

lama, karena setiap pemasukan dan pengeluaran barang harus dicatat secara manual, dan dilakukan oleh satu orang. Belum lagi jika terjadi kesalahan pemasukan nama sales, atau penghitungan salah tentang jumlah barang yang masuk dan yang keluar data harus dicatat ulang, itu semua menyebabkan borosnya jumlah kertas yang digunakan, waktu yang banyak terbuang, dan tenaga kerja yang terbuang. Oleh karena itu komputerisasi dalam sistem persediaan barang di perusahaan spare part kapal sangat diperlukan. Karena dengan adanya komputerisasi, pencatatan transaksi jadi lebih praktis, lebih hemat waktu, dan dapat dilakukan oleh orang lain, apabila suatu saat diperlukannya pengganti dalam pencatatan transaksi. Atas dasar pemikiran diatas maka dicoba dibuat suatu perancangan sistem persediaan barang untuk perusahaan spare part kapal.

Dalam penulisan ini dibuat sistem informasi persediaan barang spare part kapal dengan harapan dapat membantu perusahaan dalam pencatatan transaksi barang di gudang dengan lebih mudah dan menghemat waktu.

2. Konsep Metodologi Unified Modeling Language (UML)

Secara resmi *Unified Modeling Language (UML)* dibuat pada bulan oktober 1994 ketika Rumbaugh bergabung dengan Booch membuat proyek pendekatan metoda *uniform* atau seragam dari masing-masing metoda mereka.

Unified Modeling Language (UML) adalah sebuah bahasa untuk menentukan, visualisasi, konstruksi dan mendokumentasikan deskripsi dari sistem software untuk memodelkan bisnis.

Unified Modeling Language (UML) merupakan sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasi, membangun dan pendokumentasian dari sebuah sistem pengembangan software berbasis objek (object oriented).

2.1 Model Statis Dan Model Dinamis

Model adalah sebuah sistem yang direpresentasikan secara abstrak dibuat untuk mengerti sistem dalam membangun atau memodifikasi sistem. Membangun model untuk suatu sistem *software* sangat bergantung pada konstruksinya atau kemudahannya dalam memperbaikinya. Oleh karena itu pembuatan model dalam sama pentingnya dengan membuat cetak biru (*blue print*) untuk membuat bangunan yang besar.

Model Statis

Model statis merepresentasikan struktur atau aspek sistem yang tetap atau tidak berubah-ubah. Contohnya *Class Diagram*.

Model Dinamis

Model dinamis merupakan kebalikan dari model statis. Model dinamis dipandang sebagai kumpulan dari prosedur sitem yang berubah-ubah.

2.2 Diagram didalam UML

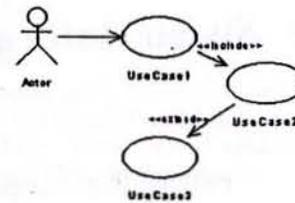
Untuk membuat suatu model UML memiliki diagram grafis sebagai berikut:

1. *Use-case Diagram*
2. *Class Diagram*
3. *Behavior Diagram*
 - a. *Statechart Diagram*
 - b. *Activity Diagram*
 - c. *Interaction Diagram*
 - *Sequence Diagram*
 - *Collaboration Diagram*
4. *Implementation Diagram*
 - a. *Component Diagram*
 - b. *Deployment Diagram*

Diagram-diagram tersebut diberi nama berdasarkan sudut pandang yang berbeda terhadap sistem. Dibuatnya berbagai jenis diagram diatas karena setiap sistem yang kompleks paling baik jika didekati melalui himpunan berbagai sudut pandang yang kecil.

Use-case Diagram

Use-case Diagram menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (*actor*). Diagram ini menunjukkan fungsionalitas suatu sistem dan bagaimana sistem berinteraksi dengan dunia luar. Diagram ini juga menggambarkan perilaku *software* aplikasi termasuk di dalamnya interaksi antara *actor* dengan *software* aplikasi.



Gambar 1. Contoh Use-case Diagram

Elemen-elemen dalam Use-case Diagram

1. Actor

Actor menggambarkan pengguna *software* aplikasi (*User*). *Actor* membantu memberikan suatu gambaran jelas tentang apa yang harus dikerjakan *software* aplikasi. Sebuah *actor* dapat terdiri dari sistem *external* yang membutuhkan informasi dari sistem *internal*.



Gambar 2. Contoh Actor

2. Use-case

Use-case adalah skenario yang menggambarkan bagaimana User menggunakan sistem. Pemberian nama *Use-case* harus dapat mendeskripsikan fungsi *Use-case* secara umum.



Use Case Name

Gambar 3. Contoh Use-case

3. Interaksi antara Actor dengan Use-case

Asosiasi dalam Use-case Diagram

1. <<uses>>

Terjadi ketika ada aktivitas yang sama, untuk menghindari penggambaran *Use-case* yang sama dengan cukup membuat satu buah *Use-case*. *Use-case* tersebut dapat digunakan oleh *Use-case* lainnya dengan menggunakan <<uses>> asosiasi. <<uses>> asosiasi menghindari redundansi pembuatan *Use-case* yang sama.

2. <<include>>

Relationship ini digunakan bila ada sebuah *Use-case* selalu termasuk dalam alur kerja dari *Use-case* lainnya.

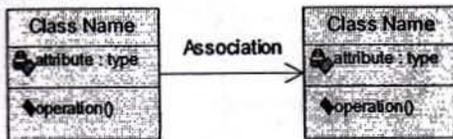
3. <<extend>>

Relationship ini digunakan bila sebuah Use-case yang hampir sama dengan Use-case lainnya tetapi agak lebih khusus. Relationship <<extend>> menunjukkan bahwa Use-case merupakan kelanjutan dari Use-case lainnya jika suatu syarat tertentu dipenuhi.

Class Diagram

Class Diagram memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas dalam suatu sistem. Selama proses analisis, *Class Diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Dan selama proses desain, *Class Diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

Kelas dalam model analisis disebut kelas analisis. Kelas analisis terdiri dari *boundary*, *control* dan *entity*. Dimana kelas analisis ini akan menjadi kelas didalam model desain.



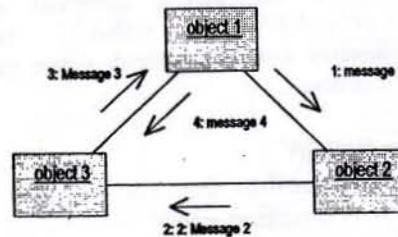
Gambar 4. Contoh Class Diagram

Dalam *Class Diagram relationship* dapat berupa:

1. **Asosiasi**
Hubungan antara dua kelas yang ditunjukkan oleh sebuah garis yang menggabungkan keduanya. Sebuah *Association* dapat diberi nama. Dalam *UML* menggunakan asterik (*) untuk merepresentasikan banyak (*many/more*) dan dua titik (..) atau koma(,) untuk merepresentasikan atau (*or*). Sejumlah objek dari satu kelas berelasi dengan sejumlah objek dalam kelas lain disebut *Multiplicity*.
2. **Aggegrasi**
Terjadi ketika sebuah kelas terdiri dari sejumlah kelas atau satu kelas disusun oleh kelas lainnya. Kunci dari *aggregation* adalah kata bagian dari (*part of*) dan terdiri dari (*consist of*) atau terbuat dari (*is made up of*) dari problem statement. *Aggregation* direpresentasikan dalam bentuk diamond.
3. **Generalisasi**
Generalisasi adalah keterhubungan membuat umum, direpresentasikan dengan garis yang ujungnya diberi sebuah segitiga dan terhubung dengan *super class*-nya.
4. **Komposisi**
Komposisi adalah bentuk keterhubungan yang lebih kuat lagi kepemilikannya dan mempunyai jangka waktu yang timbul sesuai kebutuhan.

Collaboration Diagram

Diagram yang melihat pada interaksi dan hubungan terstruktur antar objek. *Collaboration Diagram* merepresentasikan sebuah kolaborasi yang merupakan kumpulan dari objek yang berhubungan (*related*) dan konteks dan berinteraksi dimana kumpulan pesan saling bertukar diantara objek-objek untuk mendapatkan hasil atau keluaran yang diinginkan



Gambar 5. Contoh Collaboration Diagram

Elemen-elemen dalam Collaboration Diagram

1. **Boundary**
Memodelkan interaksi antara satu *actor* atau lebih dengan sistem. *Boundary* dapat berupa :
 - o *User interface* yang merupakan sarana komunikasi antara sistem dengan *User*.
 - o Sistem *interface* yang merupakan sarana komunikasi antara sistem dengan sistem informasi lainnya.
 - o *Device interface* yang merupakan sarana komunikasi antara sistem dengan device (alat)
2. **Control**
Digunakan untuk memodelkan "perilaku mengatur" untuk satu atau beberapa *Use-case* saja. Biasanya *Control* memasangkan *boundary* dengan *entity*.
3. **Entity**
Memodelkan informasi yang harus disimpan oleh sistem. Tanggung jawab dari *entity* yaitu untuk menyimpan dan mengatur informasi dalam sistem.

2.3 Konsep Dasar Rational Rose

Rational Rose adalah software yang memiliki perangkat pemodelan secara visual untuk membangun suatu solusi dalam rekayasa software dan pemodelan bisnis. Bahasa yang digunakan dalam *Rational Rose* adalah bahasa pemodelan *UML*.

View Dalam Rational Rose

Saat membuat model untuk membangun suatu software terdapat berbagai cara pandang (*view*).

Dalam *Rational Rose view* tersebut dibagi menjadi :

- a) *Use-Case View*
View yang memperlihatkan bagaimana *actor* dan *Use-case* berinteraksi.
- b) *Logical View*
Logical view mengarah pada persyaratan fungsional sistem. View ini melihat pada kelas-kelas dan hubungan antar kelas-kelas tersebut.
- c) *Component View*
View yang mengarah pada pengaturan *software*. View ini mengandung informasi mengenai komponen-komponen *software*, komponen tereksekusi dan *library* untuk sistem yang akan dimodelkan.

3. Analisa Sistem

Pada sistem persediaan spare part kapal ini ada dua buah lokasi yang digunakan sebagai tempat penyimpanan, dikenal dengan gudang besar dan gudang kecil. Gudang besar adalah sama pengertian dengan gudang kecil, namun berbeda dalam hubungan dengan bagian terkait dan dalam hal jumlah ketersediaan barang yang disimpan. Gudang besar lebih kepada berhubungan dengan bagian luar perusahaan sedangkan gudang kecil lebih kepada hubungan dengan bagian yang ada dalam perusahaan. Dilihat dari ketersediaan barang, maka gudang besar tentu saja akan menyimpan persediaan barang lebih banyak dari pada gudang kecil.

Aktivitas kegiatan persediaan barang dimulai dari adanya surat permintaan dari unit kapal ke gudang kecil, gudang kecil akan melihat ketersediaan barang, bila ada maka akan dibuat surat keluar barang, bila tidak ada akan meminta ke gudang besar, untuk selanjutnya gudang besar akan melakukan pengecekan mengenai ketersediaan barang. Apabila permintaan barang tersedia di gudang besar, maka gudang besar akan mendistribusikan barang yang diminta sehingga barang dari gudang besar dapat dikirim ke gudang kecil. Pengiriman barang ini akan mengakibatkan persediaan barang di gudang besar menjadi berkurang, sementara persediaan barang di gudang kecil sebaliknya akan menjadi bertambah. Barang di gudang besar akan menjadi bertambah apabila ada pengiriman barang dari supplier, di mana pengiriman ini akan terjadi apabila ada purchase order (PO) dari bagian pembelian. Sehingga, apabila persediaan barang di gudang kecil berkurang, akan menghubungi gudang besar, Pada saat barang diterima dari supplier ke gudang besar akan ada beberapa aktivitas, di antaranya adalah pemeriksaan barang apakah diterima dalam kondisi baik atau tidak, apakah barang dikirim oleh supplier sesuai dengan tanggal pengiriman yang seharusnya atau tidak dan yang terutama adalah apakah barang yang dikirim sesuai

dengan purchase order (PO) dari bagian pembelian atau tidak.

Langkah-langkah pembuatan :

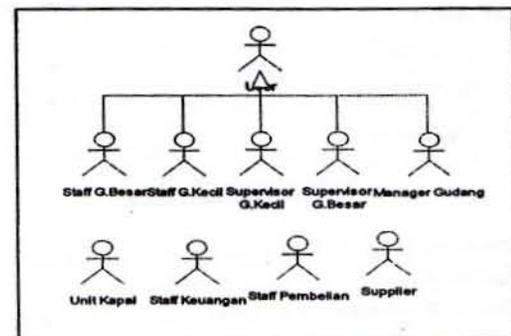
1. Analisa masalah
2. Pemodelan sistem
3. Model perancangan
4. Desain Class Diagram
5. Rancangan Database

Usecase Model

Usecase model terdiri dari dua bagian utama yaitu actor dan use-case.

Actor

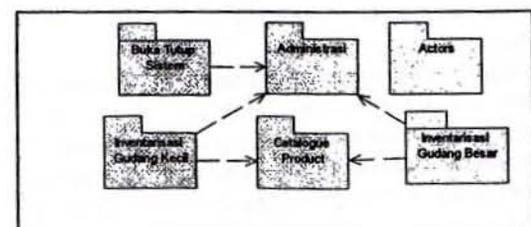
Terdapat 9 actor yang dapat digeneralisasi menjadi user, unit kapal, staff keuangan, staff pembelian, supplier dan Manager Gudang



Gambar 6. Paket Actor

Paket Use-case

Sistem ini terdiri dari 6 package usecase agar dapat dengan mudah menguraikan kegiatan.

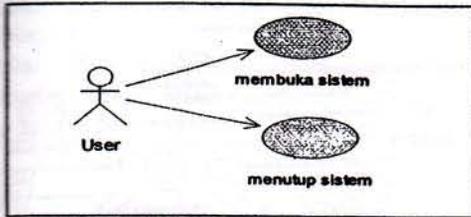


Gambar 7. Paket Use-case

Use-case Diagram

Use-case Paket Buka-Tutup Sistem

Staff gudang, Supervisor, Manger dapat membuka dan menutup aplikasi sistem informasi persediaan sparepart kapal ini, yang diawali dengan memasukan *username* dan *password*.



Gambar 8. Usecase Diagram Buka tutup

Use-case Paket Catalogue Management

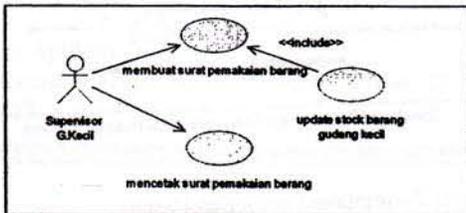
Pada paket ini menggambarkan proses yang bertujuan untuk menambahkan dan memelihara produk-produk perusahaan



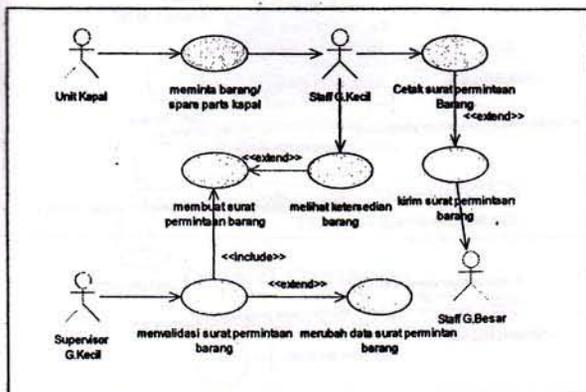
Gambar 9. Usecase Diagram Catalogue

Use-case Paket Inventarisasi Gudang Kecil

Pada paket ini berisi proses pembuatan surat pemakaian barang, permintaan barang,



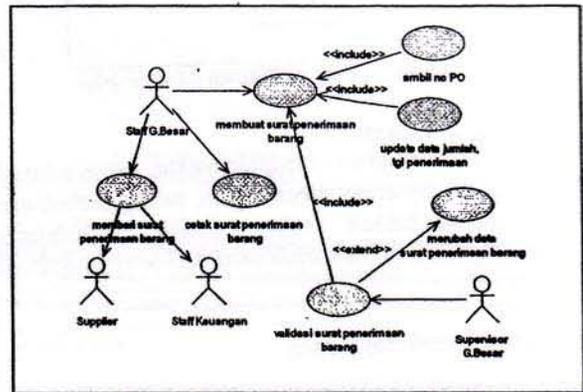
Gambar 10. Usecase Diagram Catalogue



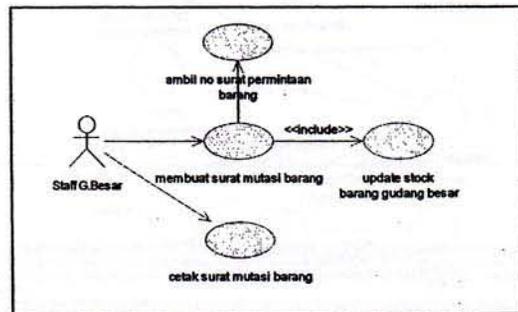
Gambar 11. Usecase Diagram Permintaan Barang

Use-case Paket Inventarisasi Gudang Besar

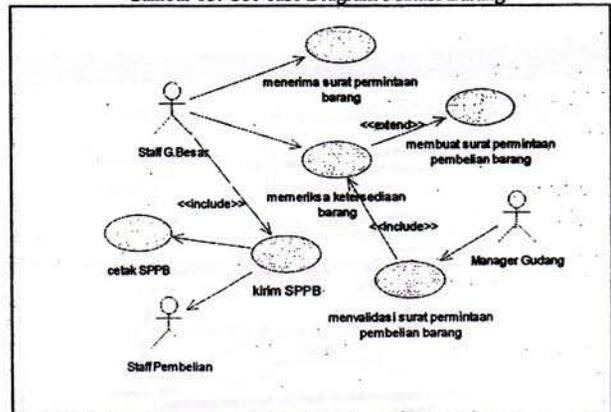
Paket ini berisi proses inventarisasi barang yang ada di gudang besar yaitu proses mutasi barang, permintaan pembelian, Dapat terlihat mulai dari gambar 12 sampai dengan gambar 14.



Gambar 12. Usecase Diagram Penerimaan barang



Gambar 13. Use-case Diagram Mutasi Barang

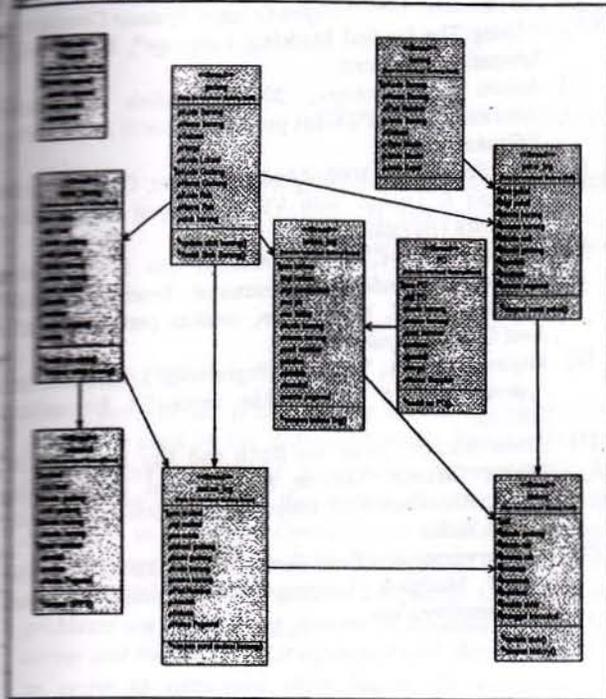


Gambar 14. Use-case Diagram Permintaan Pembelian

Use-case Paket Administration

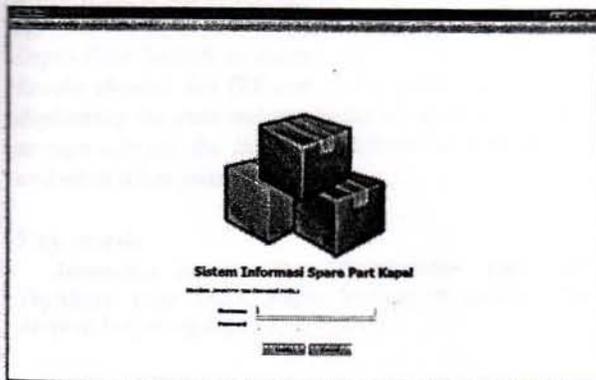
Pada paket use-case ini menggambarkan pengelolaan dalam data user.

Gambar 20. Use-case Diagram Permintaan Pembelian Barang
 Class Diagram
 Substansi akan database dari sistem informasi ini digambarkan dalam class diagram seperti terlihat pada gambar 21.

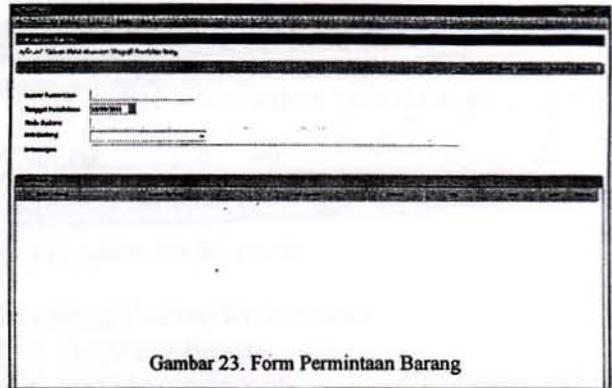


Gambar 21. Class Diagram

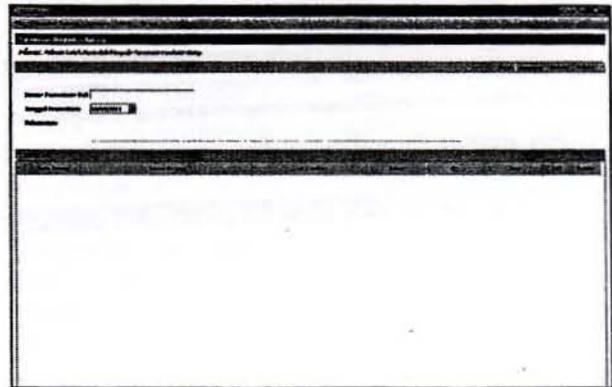
Tampilan Aplikasi
 Implementasi aplikasi terwakili oleh gambar 22 hingga gambar 27.



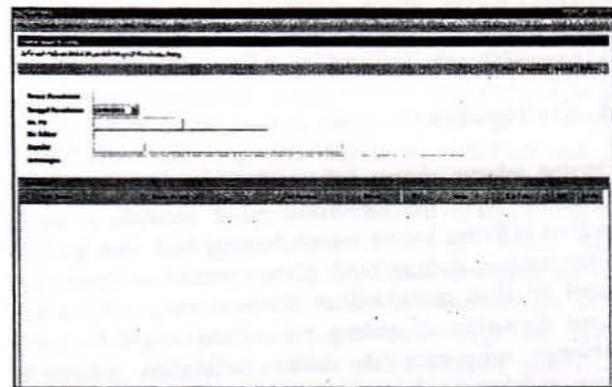
Gambar 22. Class Diagram



Gambar 23. Form Permintaan Barang



Gambar 24. Form Permintaan Pembelian Barang



Gambar 25. Form Penerimaan Barang

Gambar 26. Form Distribusi/Mutasi Barang

Gambar 27. Form Pemakaian Barang

4. Kesimpulan

Dengan adanya sistem informasi persediaan spare part kapal ini dapat mempermudah pihak perusahaan untuk melihat aktivitas keluar masuk barang, baik dari gudang besar maupun gudang kecil. Sistem persediaan spare part kapal ini akan menghasilkan database yang sama untuk dapat digunakan di gudang besar maupun gudang kecil sehingga mempermudah dalam melakukan pendataan maupun dalam pembuatan laporan yang berkenaan dengan persediaan spare part kapal.

REFERENSI

- [1] A.Suhendar dan Hariman Gunadi, 2002, "Visual Modeling menggunakan UML dan Rational Rose", Penerbit Informatika Bandung
- [2] Ali Bahrami., 1999, "Object Oriented Systems Development : Using The Unified Modeling Language", Mc Graw-Hill International Editions
- [3] Ariesto Hadi Sutopo., 2002, "Analisis dan Desain Berorientasi Objek", Edisi pertama, Penerbit J&J Learning Yogyakarta, 2002.
- [4] Indra Sosrodjojo, 2002, "Andal Software, Object Oriented Analysis & Design With Visio", Kampus Anggrek Bina Nusantara University, 2002.
- [5] Jogiyanto H. M., 1999, "Analisis dan Desain Sistem Informasi : Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis", Edisi kedua, cetakan pertama, Penerbit Andi Offset Yogyakarta, 1999.
- [6] Rogers Pressman., "Software Engineering : A Practitioner's Approach", third edition, Mc Graw-Hill International Editions.
- [7] Simon Bennett, Steve Mc Robb and Ray Farmer, 2000, "Object Oriented Systems Analysis and Design Using UML", Mc Graw-Hill International Editions, Computer Science Series.
- [8] Sri Dharwiyanti dan Romi Satria Wahono, 2003, "Pengantar Unified Modeling Language (UML)", Kuliah Umum IlmuKomputer.Com.