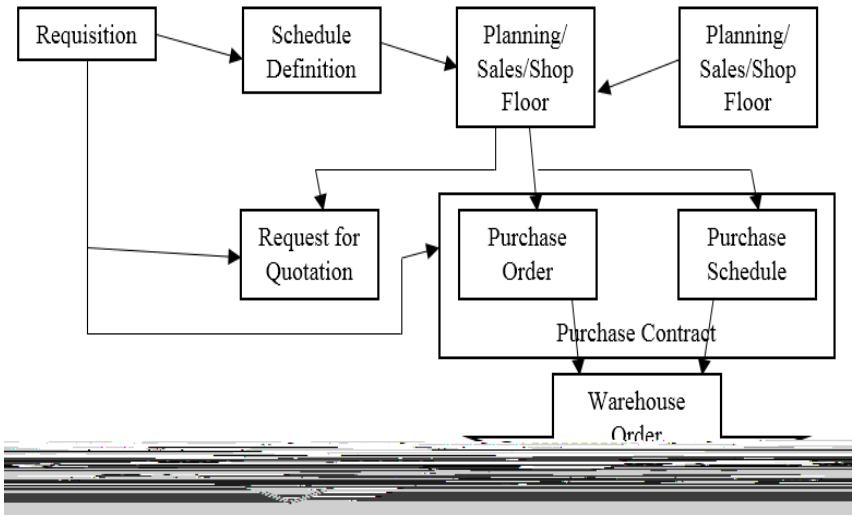
aktivitas bisnis yang lintas fungsi, lintas unit bisnis, lintas dunia. Sistem ERP dirancang berdasarkan proses bisnis yang dianggap 'best practice' proses umum yang paling layak ditiru. Misalnya, bagaimana proses umum yang sebenarnya berlaku untuk pembelian (purchasing), penyusunan stok di gudang dan sebagainya. Untuk mendapatkan manfaat yang sebesar-besarnya dari sistem ERP, maka industri kita juga harus mengikuti 'best practice process' (proses umum terbaik) yang berlaku. sasaran ERP meliputi customer service

chase order management dapat dilihat pada ilustrasi Gambar 2.2.



Gambar 2.2. Alur aktivitas pada *purchase order management* (Dhewanto, 2007)

Pada gambar terlihat bahwa proses pembelian dipicu dari adanya permintaan (*Requisition*) yang juga disertai dengan definisi jadwal kapan barang tersebut diperlukan *(schedule definition)*. Jadwal ini disesuaikan atau dijadikan dasar bagi pembuatan rencana produksi dan renana penjualan serta dengan mempertimbangkan informasi mengenai dari mana barang/material itu akan diperoleh (*sourcing information*). Berbagai masukan tadi dijadikan landasan untuk membuat penawaran kepada pihak luar (*request for quotation*). Selanjutnya, setelah penawaran diajukan, dibuatlah form pembelian *(purchase order*) dan jadwal pengiriman yang dikehendaki *(purchase schedule*), semuanya itu dituangkan dalam sebuah kontrak pembelian (*purchase contract*). Kontrak pembelian inilah yang akan dijadikan acuan oleh bagian gudang ketika menerima kiriman material dari pemasok.

2.3 PIECES

Metode PIECES merupakan kerangka kerja yang digunakan untuk klasifikasi permasalahan yang adas berdasarkan kategori yang disebutkan dalam tiap hurufnya *Performance, Information, Economic, Control, Efficiency, Service* (Adiguna, 2015).

2.3.1 *Performance*

Kinerja adalah suatu kemampuan sistem dalam menyelesaikan tugas dengan cepat sehingga sasaran dapat segera tercapai. Indikator-indikator yang dapat menunjukkan kinerja suatu sistem informasi dapat dilihat pada Tabel 2.1.

Tabel 2.1. Indikator *performnce* sistem informasi

No	Indikator Performance	Keterangan
1.	*Throughput*	Penilaian sistem dilihat dari banyaknya pekerjaan yang dilakukan berdasarkan waktu periode tertentu.
2.	*Respon time*	Delay rata-rata antara transaksi dan respon dari transaksi tersebut.
3.	*Audibilitas*	Kecocokan dimana keselarasan terhadap standar dapat diperiksa.
4.	*Kelaziman Komunikasi*	Tingkat dimana interface standar, protokol, dan *bandwith* digunakan.
5.	*Kelengkapan*	Drajat dimana implementasi penuh dari fungsi yang diharapkan tercapai.
6.	*Konsistensi*	Penggunaan desain dan teknik dokumentasi yang seragam pada keseluruhan proyek pengembangan perangkat lunak dengan toleransi kesalahan, yang terjadi pada saat program mengalami kesalahan.

2.3.2 *Information*

Informasi merupakan hal penting karena dengan informasi tersebut pihak manajemen dan user dapat melakukan langkah selanjutnya. apabila kemampuan sistem informasi baik, maka user akan mendapatkan informasi yang akurat, tepat waktu dan relevan sesuai dengan yang diharapkan.

Indikator yang dapat menunjukan informasi suatu sistem informasi yang akurat, tepat waktu dan relevan dapat dilihat pada Tabel 2.2.

Tabel 2.2. Indikator *information* sistem informasi

No	Indikator Performance	Keterangan
1.	*Accuracy*	Informasi yang dihasilkan memiliki ketepatan yang tinggi.
2.	Relevansi informasi	Informasi yang dihasilkan sesuai dengan kebutuhan.
3.	Penyajian informasi	Informasi disajikan dalam bentuk yang sesuai dan mudah diimplementasikan.
4.	Fleksibelitas data	Informasi mudah disesuaikan dengan kebutuhan.

2.3.3 *Economic*

Pemanfaatan biaya yang digunakan dari pemanfaatan informasi. Peningkatan terhadap kebutuhan informasi yang ekonomis dapat mempengaruhi pengendalian biaya dan peningkatan manfaat terhadap sistem informasi. Indikator *eco-*

nomic sistem informasi dapat dilihat pada Tabel 2.3.

Tabel 2.3. Indikator *economic* sistem informasi

No	Indikator Performance	Keterangan
1.	*Reusabilitas*	Tingkat dimana sebuah program atau bagian dari program tersebut dapat digunakan kembali di dalam aplikasi yang lain.
2.	Sumber daya	Jumlah sumber daya yang digunakan dalam pengembangan sistem, meliputi sumber daya manusia serta sumber daya ekonomi.

2.3.4 *Control*

Analisis ini digunakan untuk membandingkan sistem yang dianalisa berdasarkan pada segi integritas sistem, kemudahan akses, dan keamanan data. Indikator control sistem informasi dapat dilihat pada Tabel 2.4.

Tabel 2.4. Indikator *control* sistem informasi

No	Indikator Performance	Keterangan
1.	Integritas	Tingkat dimana akses ke perangkat lunak atau data oleh orang yang tidak berhak dapat dikontrol.
2.	Keamanan	Mempunyai mekanisme yang mengontrol atau melindungi program.

2.3.5 *Efficiency*

Efisiensi berhubungan dengan bagaimana sumber tersebut dapat digunakan secara optimal. Operasi pada suatu perusahaan dikatakan efisien atau tidak biasanya didasarkan pada tugas dan tanggung jawab dalam melaksanakan kegiatan. Indikator *efficiency* sistem informasi dapat dilihat pada Tabel 2.5.

Tabel 2.5. Indikator *efficiency* sistem informasi

No	Indikator Performance	Keterangan
1.	Usabilitas	Usaha yang dibutuhkan untuk mempelajari, mengoperasikan, menyiapkan input, dan mengintepretasikan *output* suatu program.
2.	Maintanabilitas	Usaha yang diperlukan untuk mencari dan membetulkan kesalahan pada sebuah program.

2.3.6 *Service*

Peningkatan pelayanan memperlihatkan kategori yang beragam. Peningkatan pelayanan yang lebih baik bagi manajemen, user dan bagian lain merupakan simbol kualitas dari suatu sistem informasi. Indikator *service* sistem informasi dapat dilihat pada Tabel 2.6.

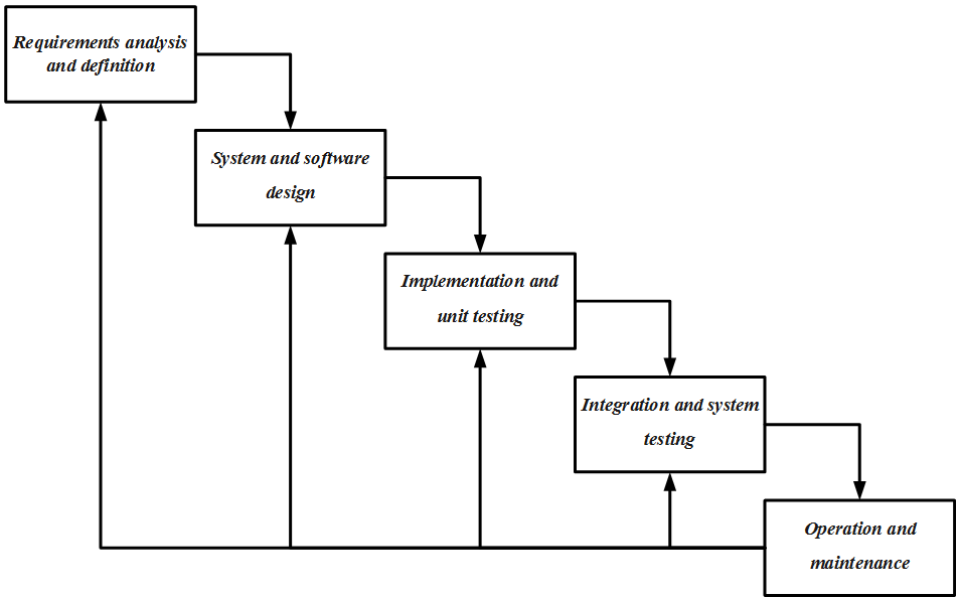
Tabel 2.6. Indikator *service* sistem informasi

No	Indikator Performance	Keterangan
1.	Akurasi	Ketelitian komputasi dan *control*.
2.	Reliabilitas	Tingkat dimana sebuah program dapat dipercaya melakukan fungsi yang diminta.
3.	Kesederhanaan	Tingkat dimana sebuah program dapat dipahami tanpa kesukaran.

2.4 Metodologi Pengembangan Rekayasa Perangkat Lunak

Model yang digunakan dalam menganalisa perangkat lunak memanfaatkan model atau paradigma siklus hidup pengembangan sistem *System Development Life Cycle* (SDLC). SDLC sering disebut sebagai pendekatan air terjun (*Waterfall Aproach*), karena terdiri dari serangkaian tugas yang mengikuti suatu pola yang teratur dan dilakukan secara *top-down*.

Metode penelitian yang diterapkan pada penelitian ini adalah dengan pengembangan metode *waterfall*. Metode *waterfall* merupakan model pengembangan sistem informasi yang sistematik dan sekuensial (Sasmito, 2017). Metode *Waterfall* memiliki tahapan-tahapan pada Gambar 2.3.



Gambar 2.3. Metode *waterfall* (Sasmito, 2017)

1. *Requirements analysis and definition*
   Layanan sistem, kendala, dan tujuan ditetapkan oleh hasil konsultasi dengan pengguna yang kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.
2. *System and software design*
   Tahapan perancangan sistem mengalokasikan kebutuhan-kebutuhan sistem baik perangkat keras maupun perangkat lunak dengan membentuk arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan penggambaran abstraksisistem dasar perangkat lunak dan hubungannya.
3. *Implementation and unit testing*
   Pada Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian melibatkan verifikasi bahwa setiap unit memenuhi spesifikasinya.
4. *Integration and system testing*
   Pada unit-unit individu program atau program digabung dan diuji sebagai sebuah sistem lengkap untuk memastikan apakah sesuai dengan kebutuhan perangkat lunak atau tidak. Setelah pengujian, perangkat lunak dapat dikirimkan ke *customer*. Pada fase ini pengujian sistem menggunakan *blackbox testing*.
5. *Operation and maintenance*
   Biasanya (walaupun tidak selalu), tahapan ini merupakan tahapan yang pal-

ing panjang. Sistem dipasang dan digunakan secara nyata. *Maintenance* melibatkan pembetulan kesalahan yang tidak ditemukan pada tahapan-tahapan sebelumnya, meningkatkan implementasi dari unit sistem, dan meningkatkan layanan sistem sebagai kebutuhan baru.

2.5 Metode Pengumpulan Data

Metode penelitian membicarakan bagaimana secara berurut suatu penelitian dilakukan, yaitu dengan alat apa dan prosedur bagaimana suatu penelitian dilakukan. Oleh karena itu, Metode penelitian dapat memandu peneliti mengetahui tentang urutan-urutan dalam penelitian. Adapun data penelitian ini terbagi dua yaitu:

1. Data Primer

   Data primer ialah data yang bersal dari sumber asli atau pertama. Data ini tidak tersedia dalam bentuk terkompilasi ataupun dalam bentuk file-file. Ada dua macam model pengambilan data yang sering digunakan dalam pengumpulan data primer yaitu:

   (a) Observasi

   Observasi merupakan salah satu teknik pengumpulan data yang tidak hanya mengukur sikap dari responden (wawancara dan angket), namun juga dapat digunakan untuk merekam berbagai fenomena yang terjadi (situasi, kondisi). Metode pengumpulan data menggunakan observasi ini terbagi menjadi dua bagian yaitu *participant* Observation dan *Non-participant Observation*. Dalam *participant Observation* peneliti secara langsung terlibat dalam kegiatan sehari-hari orang atau situasi yang diamati sebagai sumber data, sedangkan *Non-participant* Observation penelitinya tidak ikut secara langsung dalam kegiatan atau proses yang sedang diamati.

   (b) Wawancara

   Metode wawancara melibatkan presentasi rangsangan *oral-verbal* dan dijawab dalam kondisi respon *oral-verbal*. Metode dapat digunakan melalui Wawancara personal dan jika memungkinkan melalui telepon. Dalam wawancara personal memerlukan seseorang yang bertindak sebagai pewawancara yang menanyakan pertanyaan secara umum dalam pertemuan *face-to-face* dari satu orang/narasumber ke orang/narasumber lainnya. Sedangkan wawancara melalui telepon merupakan metode pengumpulan informasi dengan cara menghubungi responden.

2. Data Sekunder

   Data Sekunder merupakan data yang telah tersedia yang mengacu pada data yang telah dikumpulkan maupun dianalisa oleh seseorang. Untuk mendapatkan data yang tepat dan sesuai dengan tujuan penelitian, diperlukan tiga pertimbangan (Siska, 2015):

   (a) Jenis data harus sesuai dengan tujuan penelitian yang sudah kita tentukan sebelumnya.

   (b) Data sekunder yang dibutuhkan bukan menekankan pada jumlah tetapi pada kualitas dan kesesuaian, oleh karena itu peneliti harus selektif dan hati-hati dalam memilih dan menggunakannya.

   (c) Data sekunder biasanya digunakan sebagai pendukung data primer, oleh karena itu kadang-kadang kita tidak dapat hanya menggunakan data sekunder sebagai satu-satunya sumber informasi untuk menyelesaikan masalah penelitian.

2.6 *Website*

Hasugian (2018) *Website* adalah salah satu aplikasi yang berisikan dokumen-dokumen multimedia (teks, gambar, suara, animasi, video) didalamnya yang menggunakan protokol HTTP (hypertext transfer protokol) dan untuk mengakses menggunakan perangkat lunak yang disebut browser. *Website* juga dapat diartikan sebagai kumpulan halaman-halaman yang digunakan untuk mempublikasikan informasi berupa teks, gambar, dan program multimedia lainnya berupa animasi (gambar gerak, tulisan gerak), suara, dan atau gabungan dari semuanya itu baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait antara satu halaman dengan halaman yang lain yang sering disebut sebagai *hyperlink*. Dalam pengertian yang lain bahwa *website* adalah Kumpulan dari halaman- halaman sius yang biasanya terangkum dalam sebuah *domain* atau *subdomain*, yang tempatnya berada di dalam *World Wide Web* (WWW) di Internet.

Sebuah web page adalah dokumen ayang ditulis dalam format *Hyper Text Markup Language* (HTML), yang hampir selalu bisa diakses melalui HTTP, yaitu protokol yang menyampaikan informasi dari server *website* untuk ditampilkan kepada para pemakai melalui web *browser*. Semua publikasi *website* tersebut dapat membentuk sebuah jaringan informasi yang sangat besar. Aplikasi *website* merupakan tipe sistem yang menggunakan arsitektur *client-server*. Dimana sebuah program *client* dihubungkan ke sebuah server untuk mendapatkan informasi yang dibutuhkan, dan melengkapi permintaan yang sudah diatur oleh user. *Thin clien-*

t merupakan *clients* hanya memiliki sedikit informasi diperlukan dari pengalaman user. Informasi bisnis dan data semua berada di *server* kecuali yang hanya disediakan oleh user.*Thick clients* merupakan *clients* berisi beberapa logika pengolahan yang dibutuhkan dalam pekerjaan user yang menyeluruh (Marisa, 2015).

Dari definisi di atas, kita dapat menyimpulkan bahwa *client* digunakan untuk aplikasi web *thin clients*. Program *client*, pada hal ini adalah *browser*, hanya sebuah *interface* yang oleh user digunakan untuk melaksanakan tugas-tugas. Yang lainnya, dari data yang user perlukan untuk dioperasikan, logika yang menentukan aliran program dan eksekusi, berada pada *server*. Dari suatu perspektif web base disini adalah tugas-tugas dari *server client* (Marisa, 2015).

1. *Website server*

   server menerima permintaan-permintaan dari para *client* web *browser* dan kemudian meresponnya. Beberapa permintaan yang datang dari *client* disertai nama dan alamat item yang *client* cari, sebagaimana beberapa data user yang disediakan. *Server* menerima permintaan tersebut, memprosesnya, dan kemudian merespon data yang dicari oleh *client* atau sebuak kode *error* yang mengindikasikan bahwa item tidak terdapat pada *server*.
2. *Website client*

   Tugas *browser* menyediakan tampilan untuk user, *browser* akan menampilkan tangapan dari *server* ke *client*. Jika user meminta *server* mendapatkan dokumen, atau mungkin mengirim submit sebuah form, browserlah yang menjalankan permintaan tersebut ke dalam sesuatu yang dapat dimengerti oleh *server*. Setelah *server* selesai memenuhi proses permintaan kemudian merespon, *browser* mengambil data yang dibutuhkan dari *server* selanjutnya dikirim dan ditambilkan ke user .

2.7 *XAMPP*

XAMPP adalah sebuah *software* yang berfungsi untuk menjalankan *website* berbasis PHP dan menggunakan pengolah data *MySQL* di komputer lokal. *XAMPP* berperan sebagai *server* web pada komputer Anda. *XAMPP* juga dapat disebut sebuah *CPanel server virtual*, yang dapat membantu Anda melakukan *preview* sehingga dapat memodifikasi *website* tanpa harus *online* atau terakses dengan internet (Y. Wicaksono, 2008).

XAMPP adalah salah satu aplikasi web *server apache* yang terintegrasi dengan *mysql* dan *phpmyadmin*. *XAMPP* adalah singkatan dari X, *Apache Server, MySQL, PHPMyadmin*, dan *Phyton*. Huruf X di depan menandakan *XAMPP* bisa diinstal di berbagai operating system. *XAMPP* dapat diinstal pada *Windows*, *Linux*,

MacOS, dan Solaris. Sampai saat ini, *XAMPP* masih diberikan secara gratis, bebas didownload dan digunakan tanpa harus membayar. XAMPP pada linux juga disebut dengan LAMPP, di mana huruf L merupakan singkatan dari Linux. *XAMPP* adalah buatan dari *team apache friends*. *XAMMP* dapat didownload langsung dari situs resminya http://www.apache-friends.org. (Setyawati, 2019).

Beberapa fitur yang terdapat pada *XAMPP* 1.8.3 di antaranya:

1. *Apache 2.4.7*
2. *MySQL 5.6.14*
3. *PHP 5.5.6*
4. *phpMyAdmin 4.0.9*
5. *FileZilla FTP Server 0.9.41*
6. *Tomcat 7.0.42*
7. *Strawberry Perl 5.16.3.1 Portable*
8. *XAMPP Control Panel 3.2.1*

2.8 *MySQL*

MySQL adalah program database yang mampu mengirim dan menerima data dengan sangat cepat dan *multi user*. *MySQL* memiliki dua bentuk lisensi, yaitu *free software* dan *shareware*. Penulis sendiri dalam menjelaskan buku ini menggunakan *MySQL* yang *free software* karena bebas menggunakan database ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensi, yang berada di bawah lisensi *general public license* (GNU/GPL), yang dapat Anda download pada alamat resminya http://www.mysql.com (Komputer, 2010).

MySQL memiliki beberapa kelebihan dan keuntungan, antaranya adalah:

1. Banyak ahli mengatakan bahwa *MySQL* merupakan *server* tercepat.
2. *MySQL* adalah system manajemen *database* yang *OpenSource* (kode sumbernya terbuka), dimana software bersifat *free* atau bebas untuk digunakan oleh perseorangan atau instansi tanpa harus membeli atau membayar kepada pembuatnya.
3. *MySQL* memiliki performa yang tinggi tapi simpel.
4. *Database MySQL* memiliki bahasa *Structured Query Language* (SQL).
5. MySQL mudah diakses melalui *protocol Open Database Connectivity* (ODBC) buatan *Microsoft*. Ini menyebabkan *MySQL* dapat diakses oleh banyak *software*.
6. Semua *client* dapat mengakses server dalam satu waktu, tanpa harus menunggu yang lain untuk mengakses *database*.
7. *Database MySQL* bisa diakses dari semua tempat di internet dengan hak

akses tertentu.

8. *MySQL* adalah database yang mampu menyimpan data berkapasitas besar, sampai berukuran *Gigabyte*.
9. *MySQL* bisa berjalan di berbagai operating system seperti *Linux, Windows, Solaris,* dan lain-lain.

2.9 PHP

Menurut Anhar (2010) PHP adalah singkatan dari: *Hypertext Preprocessor* yaitu bahasa pemrograman web *server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada server (*server side HTML embedded scripting*). PHP adalah *script* yang digunakan untuk membuat halaman *website* yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru/*up to date*. Semua *script* PHP dieksekusi pada *server* di mana *script* tersebut dijalankan.

Beberapa alasan bagi kita untuk mempelajari PHP adalah karena (Anhar, 2010):

1. Kesederhanaan. User yang baru belajar pemrograman, alasan ini pasti merupakan alasan utama untuk mulai belajar PHP. Karena kesederhanaan tersebut, maka kita menjadi merasa mudah untuk belajar PHP. User yang sedikit tahu atau bahkan sama sekali tidak mengerti tentang pemrograman PHP bisa dengan cepat belajar dan mencoba membuat aplikasi web PHP. Selain itu, PHP memiliki banyak sekali fungsi built-in untuk menangani kebutuhan standar pembuatan aplikasi web. Dengan adanya fungsi-fungsi tersebut, maka tentu saja proses belajar PHP terutama dalam pengembangan aplikasi akan jauh lebih mudah karena semua sudah tersedia. Kita sebagai user tinggal memakai dan mengembangkannya.
2. Dalam sisi pemahamanan, PHP adalah bahasa *scripting* yang paling mudah karena memiliki referensi yang banyak.
3. PHP adalah bahasa open source yang dapat digunakan di berbagai sistem operasi seperti: *Linux, Unix, Macintosh,* dan *Windows*. PHP dapat dijalankan secara *runtime* melalui *console* serta dapat menjalankan perintah-perintah system. *Open source* artinya code-code PHP terbuka untuk umum dan kita tidak harus membayar biaya pembelian atas keaslian *license* yang biasanya cukup mahal. Karena *source code* PHP tersedia secara gratis, maka hal tersebut memungkinkan komunitas milis-milis dan *developer* untuk selalu melakukan perbaikan, pengembangan, dan menemukan *bug* dalam

bahasa PHP.

4. Web *Server* yang mendukung PHP dapat ditemukan di mana-mana, mulai dari *Apache*, IIS, *Lighttpd*, hingga *Xitami* dengan configurasi yang relatif mudah.
5. PHP juga dilengkapi dengan berbagai macam pendukung lain seperti *support langsung* ke berbagai macam *database* yang populer, misal: *Oracle, PostgreSQL, MySQL,* dan lain-lain.

2.10 *Unifield Modeling Language* (UML)

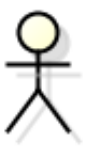
UML singkatan dari Unifield Modeling Language yang berarti bahasa permodelan standar, UML memiliki *sintaks* dan semantik, ketika membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. Bagaimana menggunakan elemen pada model-model yang dibuat berhubungan satu dengan yang lainnya harus mengikuti standar yang ada. Intinya UML merupakan alat komunikasi yang konsiten dalam menunjang pengembangan sistem saat (Nawang, Kurniawati, dan Duta, 2017). UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk:

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabat sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem. UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori.

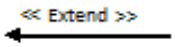
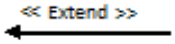
2.10.1 *Use Case Diagram*

Use case diagram atau diagram *use case* merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu (Nawang dkk., 2017). Berikut adalah simbol-simbol yang ada pada *use case* diagram dapat dilihat pada Tabel 2.7.

Tabel 2.7. Simbol-simbol *use case diagram*

No	Simbol	Notasi	Keterangan
1.		*Actor*	Actor adalah *abcstraction* dari orang atau sistem lain yang mengaktifkan fungsi dari terget.

Tabel 2.7 Simbol-simbol *use case diagram* (Tabel lanjutan...)

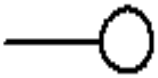
No	Simbol	Notasi	Keterangan
2.		*Use case*	menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dan actor.
3.		*Association*	Asosiasi antar actor dan use case, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
4.		*Extend*	Extend merupakan perluasan dari use case lain jika kondisi atau syarat terpenuhi.
5.		*Generalization*	Generalization digunakan untuk memperlihatkan bahwa beberapa actor atau use case memiliki suatu hal yang bersifat umum.

2.10.2 *Class Diagram*

Class Diagram atau diagram kelas adalah menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang dibuat untuk membangun sistem. *Class* memiliki apa yang disebut atribut dan metode atau operasi sebagai berikut (Nawang dkk., 2017).

1. Atribut merupakan variabel-variabel yang dimiliki oleh kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.
3. Simbol yang digunakan dalam class diagram terdapat pada Tabel 2.8.

Tabel 2.8. Simbol-simbol *class diagram*

No	Simbol	Notasi	Keterangan
1.		*Class*	Kelas pada struktur system.
2.		*Interface*	Sama dengan kosep interface dalam pemograman berorientasi objek.
3.		*Association*	Relasi antar kelas dengan makna umum,asosiasi biasanya juga disertai dengan multiplicity.

Tabel 2.8 Simbol-simbol *class diagram* (Tabel lanjutan...)

No	Simbol	Notasi	Keterangan
4.		*Diracted Association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity.
5.		*Dependency*	Kebergantungan antarkelas.

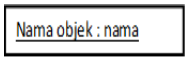
2.10.3 *Sequence Diagram*

Sequence diagram atau diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu, untuk menggambar diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

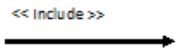
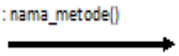
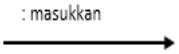
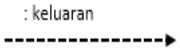
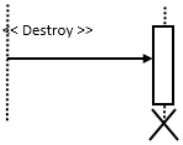
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak (Nawang dkk., 2017).

Simbol-simbol yang digunakan dalam sequence diagram dapat dilihat pada Tabel 2.9.

Tabel 2.9. Simbol-simbol *sequence diagram*

No	Simbol	Notasi	Keterangan
1.		*Actor*	Aktor merepresentasikan semua yang berinteraksi dengan sistem. Aktor bisa berupa orang, mesin, atau sistem lain.
2.		*lifeline*	Menyatakan kehidupan suatu objek.
3.		*Object*	objek yang berinteraksi diartikan pesan.
4.		Waktu aktif	Mengambarkan waktu aktif, yang terhubung oleh waktu aktif ini merupakan sebuah tahapan yang dilaksanakan.

Tabel 2.9 Simbol-simbol *sequence diagram* (Tabel lanjutan...)

No	Simbol	Notasi	Keterangan
5.		Pesan tipe *create*	Menyatakan suatu objek membuat objek lain, arah panah mengarah pada objek yang dibuat.
6.		Pesan tipe *call*	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
7.		Pesan tipe *send*	Menyatakan suatu objek mengirimkan data/masukan/ informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
8.		Pesan tipe *return*	Menyatakan suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
9.		Pesan tipe *destory*	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destory.

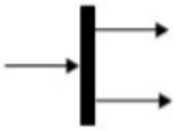
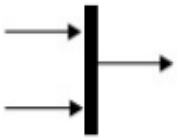
2.10.4 *Activity Diagram*

Activity Diagram menjelaskan aliran kerja dari sebuah sistem atau proses bisnis. Perlu diperhatikan bahwa *Activity* Diagram menggambarkan alur kerja sistem bukan apa yang dilakukan aktor. Simbol-simbol yang digunakan dalam activity diagram dilihat pada Tabel 2.10.

Tabel 2.10. Simbol-simbol *activity diagram*

No	Simbol	Notasi	Keterangan
1.		*Start State*	Start state adalah kondisi awal dari sebuah object terjadinya perubahan keadaan. Start state digambarkan dengan sebuah lingkaran solid.
2.		*End State*	End state adalah gambaran ketika objek berhenti memberi respon terhadap sebuah event. End state digambarkan dengan lingkaran solid di dalam sebuah lingkaran kosong.
3.		*State/Activities*	State atau activities merepresentasikan kondisi dari sebuah entitas, dan digambarkan dengan segiempat yang pinggirnya.

Tabel 2.10 Simbol-simbol *activity diagram* (Tabel lanjutan...)

No	Simbol	Notasi	Keterangan
4.		*Fork* (Percabangan)	Fork atau percabangan adalah pemisalah beberapa aliran konkuren dari sebuah aliran tunggal.
5.		*Join* (Penggabungan)	Join atau penggabungan merupakan pengabungan beberapa aliran konkuren dalam seuatu aliran tunggal.
6.		*Decision*	Decisionadalah suatu logika aliran konkuren yang mempunyai dua cabang aliran konkuren.

2.11 Pengujian Sistem

Pengujian *software* sangat diperlukan untuk memastikan *software*/aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengembang atau penguji *software* harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Pengujian atau *testing* sendiri merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merupakan bagian yang tidak terpisah dari siklus hidup pengembangan *software* seperti halnya analisis, desain, dan pengkodean (Mustaqbal, Firdaus, dan Rahmadi, 2016).

Pengujian *software* haruslah dilakukan dalam proses rekayasa perangkat lunak atau *software engineering*. strategi pengujian *software* telah diusulkan dalam literatur. Semuanya menyediakan template untuk pengujian bagi pembuat *software*.

2.11.1 *Black Box Testing*

Black Box Testing berfokus pada spesifikasi fungsional dari perangkat lunak. *Tester* dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. *Black Box Testing* bukanlah solusi alternatif dari *White Box Testing* tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *White Box Testing* (Mustaqbal dkk., 2016). *Black Box* Testing cenderung untuk menemukan hal-hal berikut:

1. Fungsi yang tidak benar atau tidak ada.
2. Kesalahan antarmuka (*interface errors*).
3. Kesalahan pada struktur data dan akses basis data.

4. Kesalahan performansi (*performance errors*).
5. Kesalahan inisialisasi dan terminasi.

Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana fungsi-fungsi diuji agar dapat dinyatakan *valid*?
2. *Input* seperti apa yang dapat menjadi bahan kasus uji yang baik?
3. Apakah sistem sensitif pada *input-input* tertentu?
4. Bagaimana sekumpulan data dapat diisolasi?
5. Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
6. Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Saat ini terdapat banyak metoda atau teknik untuk melaksanakan *Black Box Testing*, antara lain:

1. *Equivalence Partitioning*
2. *Boundary Value Analysis/Limit Testing*
3. *Comparison Testing*
4. *Sample Testing*
5. *Robustness Testing*
6. *Behavior Testing*
7. *Requirement Testing*
8. *Performance Testing*
9. Uji Ketahanan (*Endurance Testing*)
10. Uji Sebab-Akibat (*Cause-Effect Relationship Testing*)

2.11.2 *User Acceptance Test* (UAT)

Uji penerimaan pengguna merupakan proses pengujian oleh pengguna yang ditujukan untuk menghasilkan dokumen yang dijadikan bukti bahwa *software* yang telah dikembangkan telah dapat diterima oleh pengguna, apabila hasil pengujian (*testing*) sudah bisa dianggap memenuhi kebutuhan dari pengguna. Proses UAT didasarkan pada dokumen *requirement* yang disepakati bersama. Dokumen *requirement* adalah dokumen yang berisi lingkup pekerjaan *software* yang harus dikembangkan, dengan demikian maka dokumen ini semestinya menjadi acuan untuk pengujian. Proses dalam UAT adalah pemeriksaan dan pengujian terhadap hasil pekerjaan. Diperiksa apakah item-item yang ada dalam dokumen *requirement* sudah ada dalam *software* yang diuji atau tidak. Diuji apakah semua item yang telah ada dapat memenuhi kebutuhan penggunanya.

2.12 Penelitian Terdahulu

Dalam penelitian diperlukannya referensi dari penelitian terdahulu, penelitian terdahulu yang di gunakan pada penelitian ini adalah:

1. Rhiyananta Catur Yudowicitro (2014) dimana penelitian ini menghasilkan aplikasi ERP modul *Inventory Management* yang dapat dijadikan referensi untuk penelitian yang relevan dengan penelitian ini untuk dapat dikembangkan lagi.
2. Akbar dan Juliastrioza (2015) yang menghasilkan Penerapan *Enterprise Resource Planning* (ERP) untuk Sistem Informasi Pembelian, Persediaan, dan Penjualan Barang yang telah terkomputerisasi perusahan dapat mengatasi masalah lambatnya informasi ketersediaan barang, proses bisnis menjadi lebih cepat dan menghasilkan laporan pembelian, persediaan, dan penjualan barang serta laporan keuangan secara *real time*.
3. Zaidir (2017) Sistem yang dihasilkan peneliti layak untuk diimplementasikan karena kelengkapan rancangan yang baik untuk rancangan input maupun rancangan *output*, sehingga dapat membantu pimpinan dalam mengambil keputusan.
4. Anas (2016) sistem yang terorganisir sesuai dengan nomor seri, diaktifkan dengan mudah meyakinkan konsep FIFO, dan diserahkan kepada dealer secara akurat dengan paling sedikit kesalahan yang mungkin terjadi.
5. Chavarria-Barrientos dan Chen (2017) model riset terapan dalam membantu menilai posisi stakeholder selama proses adopsi sistem manajemen persediaan. Meskipun masih tentatif dari alam, model dapat memberikan manajer proyek dengan pemahaman yang mendalam tentang mekanisme kompleks selama pelaksanaan inventarisasi sistem manajemen dengan menilai kekuatan dan posisi kepentingan dari para pemangku kepentingan di basis yang sering. Semoga, di dalam waktu dekat ini akan berkontribusi pada proyek - proyek yang lebih efektif di Indonesia area sistem manajemen persediaan.