

B A B 2

L A N D A S A N T E O R I

2.1 Sistem Informasi

Sistem adalah kumpulan komponen-komponen yang saling terkait dan saling berinteraksi untuk melakukan tugas dalam rangka mencapai suatu tujuan tertentu (William & Sawyer, 2015). Sedangkan informasi adalah aset terbesar sebuah organisasi dalam menciptakan, menangkap, mengatur, menyimpan, mengambil, menganalisis dan bertindak berdasarkan informasi sebagai kegiatan mendasar dalam setiap organisasi (Wallace, 2014). Sistem Informasi merupakan seperangkat komponen yang saling terkait untuk mengumpulkan, memanipulasi, menyimpan dan menyebarluaskan data dan informasi serta menyediakan mekanisme umpan balik untuk memenuhi tujuan (Stair & Reynolds, 2017).

Informasi tidak sama dengan data dan pengetahuan. Data terdiri dari fakta mentah, informasi adalah kumpulan data yang diorganisir dan diproses sehingga memiliki nilai tambahan di luar nilai fakta individu. Sedangkan pengetahuan merupakan kesadaran dan pemahaman tentang serangkaian informasi dan cara-cara agar informasi dapat berguna untuk mendukung tugas tertentu atau mencapai keputusan (Stair & Reynolds, 2018).

2.1.1 Komponen Sistem Informasi

Suatu sistem informasi memiliki beberapa komponen seperti (Stair & Reynolds, 2017) :

1. *Input*

Aktivitas mengumpulkan dan menangkap data mentah.

2. *Processing*

Mengubah atau merubah data menjadi *output* yang bermanfaat. Pemrosesan dapat melibatkan pembuatan perhitungan, membandingkan data, mengambil tindakan alternatif dan menyimpan data untuk penggunaan di masa mendatang. Proses data menjadi informasi yang berguna sangat penting dalam pengaturan bisnis.

3. *Output*

Output atau keluaran melibatkan pembuatan informasi yang bermanfaat, biasanya dalam bentuk dokumen dan laporan.

4. *Feedback*

Informasi dari sistem yang digunakan untuk membuat perubahan pada kegiatan *input* atau pemrosesan.

2.1.2 Nilai Sistem Informasi

Nilai sistem informasi secara langsung terkait dengan bagaimana membantu pengambilan keputusan demi mencapai tujuan organisasi. Informasi yang berharga dapat membantu orang melaksanakan tugas dengan lebih efisien dan efektif (Stair & Reynolds, 2018). Informasi menjadi hal yang sangat penting karena informasi merupakan salah satu penentu pengambilan keputusan yang berkaitan dengan proses bisnis suatu organisasi ataupun perusahaan.

2.1.3 Karakteristik Sistem Informasi

Suatu sistem dapat dikatakan sebagai sebuah sistem informasi jika sudah memenuhi karakteristik utama dari sebuah sistem informasi. Karakteristik utama ini menunjukkan sebuah sistem dapat benar-benar dikatakan sebuah sistem yang dapat memberikan arus informasi dari *host* ke seluruh penggunanya. Beberapa karakteristik yang dimiliki oleh sistem informasi yaitu (Stair & Reynolds, 2018) :

1. *Accessible*

Sebuah informasi harus mudah diakses oleh pengguna yang berwenang sehingga mereka dapat memperolehnya dalam format yang tepat dan pada waktu yang tepat untuk memenuhi kebutuhan mereka.

2. *Accurate*

Sebuah informasi yang akurat bebas dari kesalahan. Dalam beberapa kasus, informasi yang tidak akurat dihasilkan karena data yang tidak akurat dimasukkan ke dalam proses transformasi. Biasanya dapat disebut *garbage in, garbage out* (GIGO).

3. *Complete*

Informasi yang lengkap mengandung semua fakta penting, misalnya laporan investasi yang tidak mencakup semua biaya penting tidak lengkap.

4. *Economical*

Informasi juga harus relatif ekonomis untuk memproduksinya. Pembuat keputusan harus selalu menyeimbangkan nilai informasi dengan biaya produksinya.

5. *Flexible*

Informasi yang fleksibel dapat digunakan untuk berbagai kebutuhan. Misalnya informasi tentang berapa banyak persediaan yang tersedia untuk bagian tertentu dapat digunakan oleh perwakilan penjualan dalam menutup penjualan oleh manajer produksi untuk menentukan apakah lebih banyak persediaan diperlukan, dan oleh seorang eksekutif keuangan untuk menentukan nilai total perusahaan telah berinvestasi dalam persediaan.

6. *Relevant*

Informasi yang relevan penting bagi pembuat keputusan. Sebagai contoh, informasi yang menunjukkan bahwa harga kayu mungkin turun bisa jadi tidak relevan dengan produsen *chip* komputer.

7. *Reliable*

Sebuah informasi terpercaya yang dapat dipercaya oleh pengguna. Beberapa kasus keandalan informasi tergantung pada keandalan metode pengumpulan data. Dalam kasus lain, keandalan tergantung pada sumber informasi. Sebuah rumor dari sumber yang tidak diketahui bahwa harga minyak mungkin naik mungkin tidak dapat diandalkan.

8. *Secure*

Sebuah informasi harus aman dari akses oleh pengguna yang tidak sah.

9. *Simple*

Informasi harus sederhana, tidak rumit. Informasi yang canggih dan terperinci mungkin tidak diperlukan. Dalam kenyataannya, terlalu banyak informasi dapat menyebabkan pembuat tidak dapat menentukan apa yang benar-benar penting.

10. *Timely*

Informasi tepat waktu disampaikan ketika dibutuhkan. Sebagai contoh, mengetahui kondisi cuaca minggu lalu tidak akan membantu ketika mencoba memutuskan pakaian apa yang akan dikenakan hari ini.

11. *Verifiable*

Informasi harus dapat diverifikasi sehingga kita dapat memastikan bahwa itu benar, mungkin dengan memeriksa banyak sumber untuk informasi yang sama.

2.2 Sekolah

Sekolah adalah satuan pendidikan yang berjenjang dan berkesinambungan untuk menyelenggarakan kegiatan belajar mengajar (UU No. 2 Tahun 1989). Sekolah merupakan sistem sosial yang dibatasi oleh kegiatan-kegiatan untuk berinteraksi dan membentuk suatu kesatuan sosial agar dapat menghasilkan sesuatu yang bermanfaat bagi masyarakat. Pada tanggal 16 Mei 2005, Pemerintah menerapkan Standar Nasional Pendidikan (SNP) yang menyatakan bahwa seluruh sekolah di Indonesia yang memenuhi standar nasional dapat menyelenggarakan pendidikan. Pendidikan standar wajib dilakukan di sekolah dan oleh sekolah dengan delapan standar yang ditempuh secara bertahap dan harus dipenuhi oleh sekolah.

Delapan Standar Nasional Pendidikan (SNP) yang dinyatakan dalam Undang-undang yaitu (UU No. 20 Pasal 1 Ayat 17 Tahun 2003) :

1. Standar kompetensi kelulusan

Kualifikasi kemampuan lulusan yang mencakup sikap, pengetahuan, dan keterampilan. Standar kompetensi kelulusan digunakan sebagai pedoman penilaian dalam menentukan kelulusan peserta didik dari satuan pendidikan.

2. Standar isi

Ruang lingkup materi dan tingkat kompetensi yang dituangkan dalam kriteria tentang kompetensi tamatan, kompetensi bahan kajian, kompetensi mata pelajaran, dan silabus pembelajaran yang harus dipenuhi oleh peserta didik pada jenjang dan jenis pendidikan tertentu. Standar isi ini memuat kerangka dasar dan struktur kurikulum, beban belajar, kurikulum tingkat satuan pendidikan serta kalender pendidikan.

3. Standar proses

Standar yang berkaitan dengan pelaksanaan pembelajaran pada satuan pendidikan untuk mencapai standar kompetensi lulusan.

4. Standar pendidik dan tenaga pendidikan

Kriteria pendidikan prajabatan dan kelayakan fisik maupun mental serta pendidikan dalam jabatan, pendidik harus memiliki kualifikasi akademik dan kompetensi sebagai agen pembelajaran, sehat jasmani dan rohani serta memiliki kemampuan mewujudkan tujuan pendidikan nasional.

5. Standar sarana dan prasarana

Standar nasional pendidikan yang berkaitan dengan kriteria minimal tentang ruang belajar, tempat berolahraga, tempat beribadah, dan fasilitas lainnya.

6. Standar pengolahan

Standar nasional pendidikan yang berkaitan dengan perencanaan, pelaksanaan dan pengawasan kegiatan pendidikan pada tingkat satuan pendidikan.

7. Standar pembiayaan

Standar yang mengatur komponen dan besarnya operasi satuan pendidikan yang berlaku selama satu tahun dan pembiayaan pendidikan tersebut meliputi biaya investasi, biaya operasi dan biaya personal.

8. Standar penilaian

Standar nasional pendidikan yang berkaitan dengan mekanisme, prosedur dan *instrument* penilaian hasil belajar peserta didik.

Pendidikan nasional berfungsi untuk mengembangkan kemampuan dan membentuk watak serta peradaban bangsa yang bermartabat dalam rangka mencerdaskan kehidupan bangsa. Tujuan dari pendidikan tersebut untuk mengembangkan potensi peserta didik agar menjadi manusia yang beriman dan bertakwa kepada Tuhan Yang Maha Esa, berakhlak mulia, sehat, berilmu, cakap, kreatif, mandiri dan menjadi warga negara yang demokratis serta bertanggung jawab (UU No. 20 Pasal 3 Tahun 2003).

2.2.1 Penerimaan Siswa Baru

Penerimaan siswa baru merupakan langkah awal bagi peserta didik untuk memasuki jenjang pendidikan. Tidak hanya penting bagi peserta didik melainkan bagi sekolah juga karena ini merupakan titik awal dalam menentukan kelancaran tugas belajar dan mengajar di dalam sebuah sekolah.

Dalam menerima siswa baru, setiap sekolah menerapkan sistem zonasi dimana sekolah yang diselenggarakan oleh pemerintah daerah wajib menerima calon peserta didik yang berdomisili pada radius zona terdekat dari sekolah paling sedikit 90% dari total jumlah peserta didik yang diterima (Permendikbud No. 17 Tahun 2017). Domisili calon peserta didik berdasarkan alamat pada kartu keluarga yang diterbitkan paling lambat 6 bulan sebelum pelaksanaan Penerimaan Peserta Didik Baru (PPDB). Zonasi ini diterapkan agar terjadi pemerataan kualitas pendidikan.

Dalam penyeleksian PPDB ada kriteria khusus untuk kelas 10 (sepuluh) SMA atau SMK dengan memprioritaskan sesuai daya tampung masing-masing rombongan belajar di setiap sekolah yang sudah ditentukan. Pertama yang harus diprioritaskan adalah jarak tempat tinggal dari rumah ke sekolah sesuai dengan zonasi tersebut, kedua usia, ketiga nilai hasil ujian sekolah (bagi lulusan SD) dan Surat Keterangan Hasil Ujian Nasional (SKHUN) bagi lulusan SMP. Kemudian yang terakhir berdasarkan prestasi akademik dan non-akademik sesuai dengan ketentuan sekolah masing-masing.

Penelitian terkait penerimaan siswa baru pernah dilakukan di SMPN 1 Kelapa, Bangka Belitung. Proses bisnis yang terjadi, panitia menyediakan formulir yang masih menggunakan media kertas yang diisi dengan tulisan tangan. Seringkali tulisan tangan ini susah dibaca sehingga menghambat proses penerimaan siswa baru. Penelitian proses penerimaan siswa baru berfokus pada formulir *online* yang dapat diisi dengan data siswa, orang tua siswa, dan kelengkapan dokumen pendukung penerimaan siswa baru (Sarwindah, 2018).

Selain itu, penelitian penerimaan siswa baru juga pernah dilakukan di SMA Negeri 1 Pulokulon, Semarang. Proses penerimaan siswa baru dilakukan secara bertahap, yaitu seleksi dokumen, tes seleksi dan daftar ulang. Setiap tahap dapat menggugurkan jumlah siswa yang mengikuti seleksi. Prosedur penerimaan siswa baru ini dilakukan berulang namun sekolah masih mengalami kesulitan dan lambat dalam proses seleksi siswa baru (Ningtyas, Badrul, & Sulistyowati, 2018).

2.2.2 Pembayaran Administrasi Siswa

Setiap sekolah memiliki administrasi yang wajib dilakukan di antaranya adalah Sumbangan Pembinaan Pendidikan (SPP). Bagi peserta didik baru, ada

beberapa syarat dalam mendaftarkan diri selain dengan mengisi formulir pendaftaran yaitu melakukan pembayaran apabila siswa tersebut sudah diterima di sekolah dengan jenjang yang sesuai. Hal-hal yang termasuk di dalam administrasi sekolah bagi siswa baru meliputi pembayaran seragam sekolah, buku pelajaran, SPP dan kegiatan lainnya yang diselenggarakan oleh pihak sekolah.

Pembayaran seragam sekolah hanya dilakukan sekali ketika siswa pertama kali masuk sekolah sebagai siswa baru. Buku pelajaran dibayarkan setiap tahunnya ketika mereka naik ke jenjang berikutnya yang terdiri dari buku teks. Buku teks adalah buku acuan wajib untuk digunakan di satuan pendidikan dasar, menengah atau perguruan tinggi yang memuat materi pembelajaran dalam rangka peningkatan keimanan, ketakwaan, akhlak mulia, kepribadian, penguasaan ilmu pengetahuan dan teknologi, peningkatan kepekaan dan kemampuan estetis, peningkatan kemampuan kinestetis dan kesehatan yang disusun berdasarkan standar nasional pendidikan (Permendiknas No. 2 Tahun 2008).

SPP dibayarkan setiap bulannya pada periode tahun pelajaran selama satu tahun dimulai sejak bulan Juli hingga Juni tahun berikutnya. Iuran bulanan SPP wajib dibayarkan oleh setiap siswa selama satu tahun ajaran yaitu 12 (dua belas) bulan. Pembayaran SPP juga merupakan syarat pengambilan Raport, Ijazah, SKHUN, dokumen kelulusan lain dan atau permohonan surat keterangan pindah sekolah (mutasi) sesuai dengan peraturan masing-masing sekolah. Kegiatan-kegiatan sekolah lainnya seperti latihan dasar kependidikan (LDK), pramuka, dan lain sebagainya sesuai dengan sekolah masing-masing.

2.2.3 Pembagian Kelas

Sistem pembagian kelas dan jumlah siswa setiap tahunnya pasti terjadi perubahan. Setiap sekolah juga memiliki aturan yang sama dalam menentukan jumlah siswa per Rombongan Belajar (rombel) yang sudah diatur di dalam undang-undang pendidikan.

Jumlah peserta didik dalam satu Rombongan Belajar diatur sebagai berikut (Permendikbud No. 17 Pasal 24 Tahun 2017) :

- Jenjang SMK dalam satu kelas berjumlah paling sedikit 15 (lima belas) peserta didik dan paling banyak 36 (tiga puluh enam) peserta didik.

Adapula aturan jumlah rombel bagi setiap sekolah yang diatur sebagai berikut (Permendikbud No. 17 Pasal 26 Tahun 2017) :

- Jenjang pendidikan SM K atau bentuk lain yang sederajat, jumlah rombel paling sedikit 3 dan paling banyak 72 rombel. Setiap tingkat paling banyak 24 rombel.

Semua ketentuan di atas didukung dengan pernyataan dari Surat Edaran Mendikbud “*Ketentuan jumlah peserta didik dalam satu rombongan belajar dan jumlah rombongan belajar pada sekolah diberlakukan hanya untuk peserta didik baru pada kelas 1, kelas 7, dan kelas 10 untuk setiap sekolah*” (Permendikbud No. 17 Tahun 2017). Tetapi tidak semua sekolah di Indonesia dapat memenuhi ketentuan di atas karena masih belum bisa menampung peserta didik yang sudah tersedia berdasarkan ketentuan mengenai zonasi, jumlah peserta didik di dalam satu rombongan belajar, dan jumlah rombongan belajar pada setiap sekolah dapat dilakukan secara bertahap sesuai dengan kesiapan setiap provinsi / kabupaten / kota setempat.

2.2.4 Laporan Penerimaan Siswa Baru

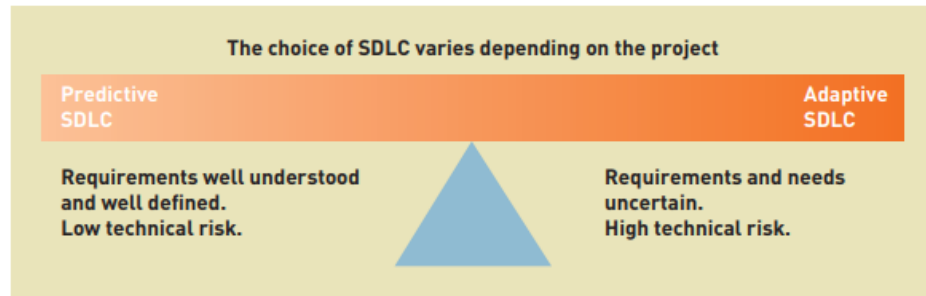
Sistem laporan untuk penerimaan siswa baru terdapat 2 bentuk yaitu laporan tentang banyaknya jumlah siswa yang masuk dan laporan tentang total biaya yang sudah calon siswa bayarkan kepada sekolah. Setiap sekolah memiliki tenaga administrasi yang handal dan sesuai dengan Standar BSNP (Badan Standar Nasional Pendidikan) dimana memenuhi standar tenaga administrasi yang berlaku secara nasional (Permendiknas No. 24 Tahun 2008).

2.3 System Development Life Cycle (SDLC)

System Development Life Cycle (SDLC) adalah kerangka kerja yang mengidentifikasi kegiatan pengembangan sistem informasi meliputi perencanaan, analisis sistem, desain sistem, pemrograman, pengujian, tahap pelatihan pengguna, kegiatan manajemen proyek lainnya yang diperlukan dalam pengembangan sistem informasi baru (Satzinger, Jackson, Burd, 2016).

Ada beberapa pendekatan pengembangan sistem, namun biasanya cukup dikelompokkan menjadi pendekatan prediktif dan pendekatan adaptif. Pendekatan adaptif digunakan ketika persyaratan sistem atau kebutuhan pengguna belum cukup dipahami. Proyek tidak dapat direncanakan sepenuhnya

di awal. Namun ditentukan poin - poin utamanya saja, selanjutnya pengembangan harus fleksibel dan cepat beradaptasi bila ada perubahan selama proses pengembangan (Satzinger, Jackson, Burd, 2016).



Gambar 2.1 SDLC dengan pendekatan prediktif vs pendekatan adaptif

(Satzinger, Jackson, Burd, 2016)

Di dalam SDLC terdapat enam proses inti yang harus selalu ada dalam pengembangan sistem informasi yaitu (Satzinger, Jackson, Burd, 2016) :

1. Identifikasi masalah atau menunggu persetujuan melanjutkan proyek.
2. Merencanakan dan memantau proyek serta menentukan apa yang harus dilakukan, bagaimana melakukannya, dan siapa yang melakukannya.
3. Temukan dan pahami detail masalah atau kebutuhan yang ada.
4. Desain komponen sistem yang memecahkan masalah atau memenuhi kebutuhan serta bagaimana cara kerjanya.
5. Membangun, menguji, dan mengintegrasikan komponen sistem, lebih banyak berkaitan dengan pemrograman dan integrasi komponen.
6. Selesaikan pengujian sistem kemudian menjalankan solusi yang dibuat.

Sebagian besar sistem informasi yang akan dikembangkan cenderung kompleks dari yang diperkirakan. Untuk mempermudah implementasi enam proses inti di atas, maka diperlukan pedoman metodologi yang komprehensif. Metodologi pengembangan sistem merupakan proses keseluruhan yang menentukan cara untuk melaksanakan pengembangan proyek. Setiap organisasi mengembangkan metodologi pengembangannya sendiri dari waktu ke waktu sesuai dengan kebutuhannya. Salah satu model pengembangan yang populer digunakan adalah *Agile Development* (Satzinger, Jackson, Burd, 2016).

Filosofi dasar *Agile* adalah bahwa anggota tim maupun pengguna tidak sepenuhnya memahami masalah dan kompleksitas sistem baru, sehingga rencana proyek dan pelaksanaan proyek harus responsif terhadap masalah yang tidak diantisipasi. Tim pengembang harus siap menerima perubahan dan persyaratan baru yang muncul selama proses pengembangan secara cepat dan fleksibel (Satzinger, Jackson, Burd, 2016).

Enam proses inti SDLC masih terlibat dalam *Agile Development*, tetapi dilakukan secara iteratif dimana proses pengembangan inti diulang untuk setiap komponen. Komponen utama dikembangkan pertama lalu komponen tambahan dikerjakan berikutnya (Satzinger, Jackson, Burd, 2016).

Core processes	Iterations					
	1	2	3	4	5	6
Identify the problem and obtain approval.	[Progress bar across all iterations]					
Plan and monitor the project.	[Progress bar across all iterations]					
Discover and understand details.	[Progress bar across all iterations]					
Design system components.	[Progress bar across all iterations]					
Build, test, and integrate system components.	[Progress bar across all iterations]					
Complete system tests and deploy the solution.	[Progress bar across all iterations]					

Gambar 2.2 Enam Proses Inti SDLC dengan Iterasi

(Satzinger, Jackson, Burd, 2016)

Setiap iterasi melibatkan enam proses inti ditampilkan sebagai baris dalam tabel. Di akhir iterasi, kerja sistem selesai dan dievaluasi. Iterasi berlangsung selama periode pendek biasanya dua hingga empat minggu. Kelebihan pengembangan secara iteratif yaitu sistem bisa lebih cepat digunakan karena fungsi utama dikerjakan dahulu (Satzinger, Jackson, Burd, 2016).

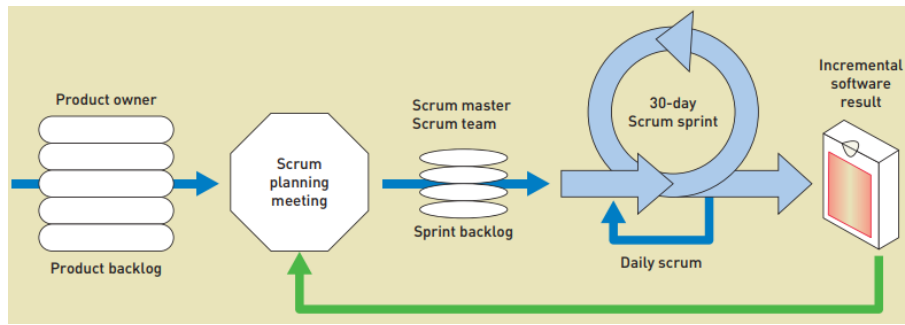
2.4 Scrum

Filosofi *Scrum* didasarkan pada prinsip-prinsip pengembangan *Agile*. *Agile Development* merupakan serangkaian pedoman yang digunakan untuk mengembangkan sistem informasi dalam lingkungan yang tidak diketahui, cepat berubah, dan dapat digunakan bersama metodologi pengembangan sistem

apapun. Biasanya, digunakan untuk melengkapi pendekatan adaptif bagi SDLC dan metodologi yang mendukungnya. Tetapi penekanannya adalah pada mengambil pendekatan adaptif dan membuatnya gesit dalam semua kegiatan dan tugas pengembangan (Satzinger, Jackson, Burd, 2016).

Scrum cukup responsif terhadap perubahan dan dinamis dimana pengguna mungkin tidak tahu persis apa yang dibutuhkan dan mungkin akan sering terjadi perubahan prioritas. Banyaknya perubahan mungkin menghambat penyelesaian proyek. *Scrum* sangat unggul dalam situasi seperti ini. Proyek dikembangkan bertahap dengan memasukkan rincian tugas ke dalam daftar dan menyelesaikannya. Daftar tugas disebut dengan *product backlog* dimana di dalamnya terdapat usulan, fitur serta *platform*. Pengerjaan tugas ditentukan oleh prioritas dan dikerjakan dalam satu waktu sesuai dengan kebutuhan proyek saat ini (Satzinger, Jackson, Burd, 2016).

Tiga elemen utama dalam organisasi proyek *Scrum* yaitu *Product Owner*, *Scrum Master* serta anggota tim *Scrum*. *Product Owner* adalah penanggung jawab bisa juga sebagai klien. Dalam *Agile Development*, pengguna dan klien sering dilibatkan dalam proyek. Begitu juga di *Scrum*, *Product Owner* mengelola daftar *backlog*, menentukan prioritas serta permintaan apapun harus melalui persetujuannya terlebih dahulu. Untuk menyelesaikan tugas, perlu fasilitator yang membantu tim menyelesaikan setiap pekerjaan, posisi ini diisi oleh *Scrum Master* yang sebanding dengan manajer proyek. Namun karena tim mengatur tugas yang dikerjakan sendiri, fokus *Scrum Master* yaitu pada komunikasi serta pelaporan kemajuan proyek, tanpa menetapkan jadwal ataupun memberikan tugas ke bawahan langsung. Berikutnya posisi tim *Scrum* terdiri dari sekelompok kecil pengembang, yang saling bekerjasama. Bila menerima proyek yang besar, maka tim ini akan dibagi menjadi tim yang lebih kecil untuk menyelesaikan tugas. Tim juga yang akan menentukan apa yang ingin dicapai pada periode waktu tertentu. Kemudian secara mandiri membagi tugas dan menyelesaikannya (Satzinger, Jackson, Burd, 2016).



Gambar 2.3 *Scrum software development process*

(Satzinger, Jackson, Burd, 2016)

Proyek dibagi menjadi beberapa periode yang disebut dengan *Sprint*. Sedangkan tujuan per periode yang ingin dicapai dinamakan sebagai *Deliverable*. Pada awal *Sprint*, tim akan berkumpul untuk membuat rencana dan menentukan tujuan utama. Tim memutuskan tugas apa saja yang akan dikerjakan berikut dengan prioritasnya. Bila tujuan dan tugas telah disetujui, maka akan dimasukkan ke daftar *backlog*. Ketika proyek dimulai, setiap anggota tim akan memilih tugas harian mereka dari daftar *backlog* dan mengerjakannya. Untuk memantau perkembangan, *Scrum Master* beserta anggota tim berkumpul selama kurang lebih 15 setiap hari. Tujuannya adalah melaporkan masalah. Bila menemui kendala dalam pekerjaan maka tim akan berkolaborasi dalam menyelesaikannya. Pada tiap akhir *Sprint*, *Deliverable* akan dirilis. Berikutnya tim akan mengadakan pertemuan *Scrum Review* yang akan membahas kemajuan yang telah dilakukan beserta perubahan apa yang akan dikerjakan pada *Sprint* berikutnya (Satzinger, Jackson, Burd, 2016).




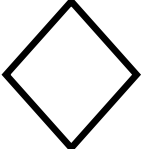


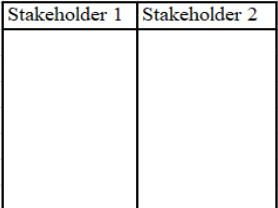
2.5 *Unified Modeling Language (UML)*

Unified Modeling Language (UML) adalah kumpulan standar konstruksi model dan notasi yang didefinisikan oleh *Object Management Group (OMG)*, organisasi standar untuk pengembangan sistem. Dengan menggunakan *UML*, analis dan pengguna akhir dapat menggambarkan dan memahami berbagai diagram spesifik yang digunakan dalam proyek pengembangan sistem (Satzinger, Jackson, Burd, 2016).

2.5.1 Activity Diagram

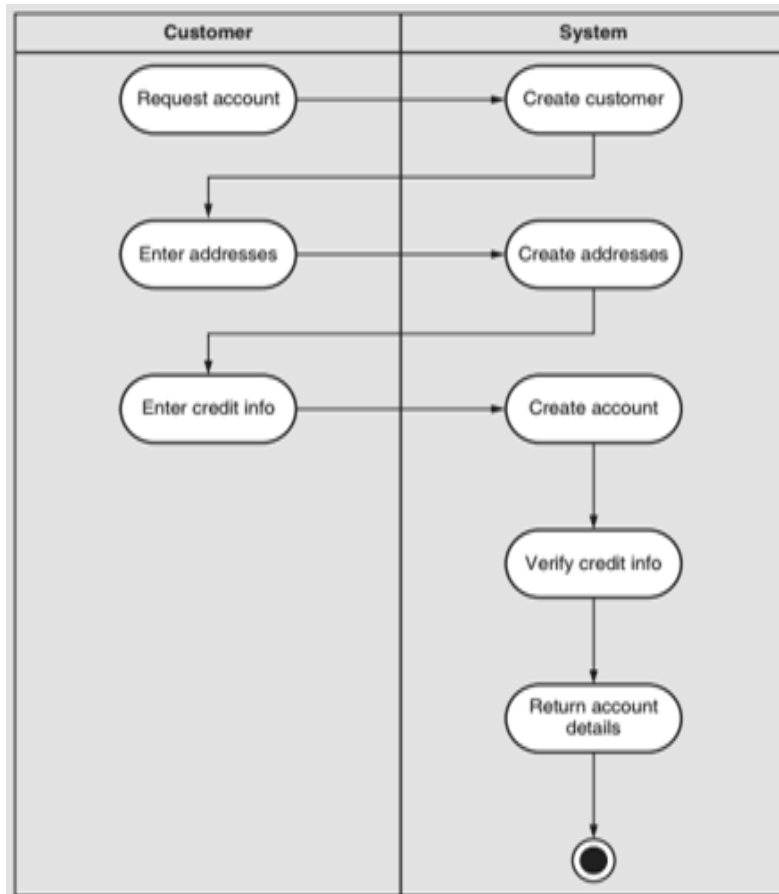
Activity diagram merupakan diagram alur kerja yang menggambarkan alur aktivitas pengguna (atau sistem) yang melakukan setiap aktivitas, dan aliran berurutan dari sebuah aktivitas tersebut. Adapun simbol yang digunakan dalam *activity diagram* adalah sebagai berikut (Satzinger, Jackson, Burd, 2016):

Tabel 2.1 Simbol *Activity Diagram* (Satzinger, Jackson, Burd, 2016)

No	Simbol	Deskripsi
1		<i>Start activity (pseudo)</i> , menunjukkan dimulainya suatu <i>workflow</i> pada sebuah <i>activity diagram</i> .
2		<i>Activity</i> , menunjukkan sebuah pekerjaan / tugas dalam <i>workflow</i> pada sebuah <i>activity diagram</i> .
3		<i>Transition arrow</i> , menunjukkan kegiatan berikutnya dalam <i>workflow</i> pada sebuah <i>activity diagram</i> .
4		<i>Decision activity</i> , menunjukkan titik keputusan untuk mengikuti satu jalur atau jalur yang lainnya.
5		<i>Ending activity</i> menunjukkan dimulainya suatu <i>workflow</i> pada sebuah <i>activity diagram</i> .
6		<i>Synchronization bar (split)</i> , <i>Synchronization bar (join)</i> menunjukkan sebagai pembagi jalur dan penggabungan jalur dalam sebuah <i>activity diagram</i> .
7		<i>Swimlane heading</i> , kolom diagram aktivitas berisi semua aktivitas untuk satu agen atau unit organisasi

Pada Gambar 2.4 terdapat contoh *activity diagram* yang menjelaskan aliran aktivitas untuk kasus penggunaan akun pelanggan. Aktivitas dimulai dari

pelanggan meminta akun, sistem merespon dengan membuat akun kemudian pelanggan memasukkan alamat, sistem merespon dengan membuat alamat, pelanggan memasukkan info lain yang berkaitan, kemudian sistem merespon dengan membuat akun, verifikasi info dan kembali pada rincian akun pelanggan.




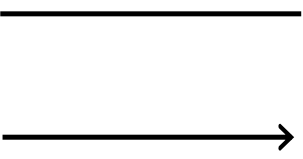



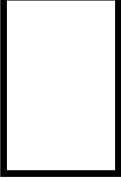
Gambar 2.4 Contoh *Activity Diagram* (Satzinger, Jackson, Burd, 2016)

2.5.2 Use Case Diagram

Use case diagram adalah model UML yang digunakan untuk menggambarkan *use case* dan hubungannya dengan pengguna secara fungsionalitas dari sebuah sistem. *Use case* menggambarkan tentang kebutuhan sistem dari sudut pandang *user*, fokus pada *automated process* (terkomputerisasi), dan menggambarkan hubungan antara *use case* dan *actor*.

Adapun simbol yang digunakan dalam *use case diagram* adalah sebagai berikut (Satzinger, Jackson, Burd, 2016) :

Tabel 2.2 Simbol *Use Case Diagram* (Satzinger, Jackson, Burd, 2016)

No.	Simbol	Deskripsi
1	 Actor	<i>Actor</i> , menunjukkan sebagai orang / sistem / perangkat lain yang menggunakan sistem, menggambarkan peran atau tugas bukan posisi / jabatan.
2		<i>Associations</i> garis tanpa panah menunjukkan partisipasi <i>actor</i> dalam sebuah <i>use case diagram</i> . <i>Associations</i> panah terbuka mengindikasikan <i>actor</i> berinteraksi secara pasif terhadap sistem.
3		<i>Use case</i> , menunjukkan kegiatan yang dilakukan oleh <i>actor</i> dalam sebuah <i>use case diagram</i> .
4	<p><<extend>></p> 	<i>Extend</i> menunjukkan bahwa <i>use case</i> target memperluas perilaku <i>use case</i> sumber pada suatu titik yang diberikan.
5	<p><<include>></p> 	<i>Include</i> menunjukkan bahwa <i>use case</i> dapat memanggil <i>use case</i> lain.
6		<i>Automation boundary</i> menunjukkan batasan sistem antara aplikasi dan pengguna namun masih tetap dalam bagian total sistem.

Pada Gambar 2.5 terdapat contoh *use case diagram* yang menjelaskan mengenai proses pembuatan dan perubahan akun dimana terdapat empat aktor yang terlibat, yaitu *customer*, *customer service representative*, *store sales representative* dan *management* yang diizinkan untuk mengakses sistem secara langsung. Namun, untuk *create / update* hanya dapat dilakukan oleh *customer*

service representative, store sales representative, sedangkan untuk penyesuaian akun hanya dapat diproses oleh management.



Gambar 2.5 Contoh *Use Case Diagram* (Satzinger, Jackson, Burd, 2016)

2.5.3 Use Case Description

Use case description merupakan model tekstual yang menjelaskan detail pemrosesan dari suatu *use case*, disebut juga sebagai rincian penggunaan diagram *use case* yang memberikan gambaran singkat terhadap pengembangan sistem. Berdasarkan kebutuhan pada saat menganalisa, *use case description* dapat dibagi menjadi dua yaitu *brief use case description* dan *fully developed use case description* (Satzinger, Jackson, Burd, 2016).

a. Brief Use Case Description

Brief use case description digunakan untuk kasus yang sangat sederhana, seperti memberikan gambaran singkat tentang pengembangan sistem untuk aplikasi kecil agar mudah dipahami (Satzinger, Jackson, Burd, 2016).

Pada Tabel 2.3 terdapat contoh *brief use case description* mengenai penambahan komentar produk atau mengirim pesan. Terdapat tiga *use case* dengan masing-masing *brief description* yang saling berkaitan yang diawali dengan membuat akun pelanggan, mencari akun pelanggan yang telah dibuat, dan menyesuaikan proses yang terjadi pada akun pelanggan.

Tabel 2.3 Contoh *Brief Use Case Description* (Satzinger, Jackson, Burd, 2016)

Use case	Brief use case description
<i>Create customer account</i>	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
<i>Look up customer</i>	User/actor enters customer account number, and the system retrieves and displays customer and account data.
<i>Process account adjustment</i>	User/actor enters order number, and the system retrieves customer and order data; actor enters adjustment amount, and the system creates a transaction record for the adjustment.

b. Fully Developed Use Case Description

Fully developed use case description adalah metode formal untuk menggambarkan *use case* dengan penjelasan pada tingkat yang lebih detail. Adapun rincian dari *fully developed use case description* adalah sebagai berikut (Satzinger, Jackson, Burd, 2016) :

1. *Use case name* menggambarkan *use case* yang akan dijelaskan.
2. *Scenario* menjelaskan tentang *scenario use case*.
3. *Triggering event*, keadaan yang menyebabkan terjadinya *use case*.
4. *Brief description* ringkasan singkat dari *use case* yang akan dijelaskan.
5. *Actors*, sebagai pengguna sistem yang terkait dengan alur *use case*.
6. *Related use case*, *use case* lain terkait dengan *use case* utama
7. *Stakeholders*, merupakan pihak-pihak yang memiliki kepentingan dengan *use case* yang akan dijelaskan.

8. *Preconditions*, merupakan kondisi awal yang harus ada seperti informasi yang harus tersedia atau kondisi *actor* sebelum *use case*.
9. *Postconditions*, kondisi yang terjadi setelah *use case* dijalankan.
10. *Flow of activities*, menjelaskan rincian urutan aktivitas yang terlaksana dari *use case* yang akan dijelaskan dan menjelaskan interaksi aktivitas yang dilakukan oleh *actor* dengan sistem.
11. *Exception condition*, menjelaskan aktivitas khusus yang terjadi apabila kondisi terpenuhi pada saat *use case* dijalankan.

Pada Tabel 2.4 terdapat contoh dari *fully developed use case description* untuk pembuatan akun pelanggan yang dilakukan secara *online* karena pelanggan baru menginginkan proses pembuatan akun dilakukan dengan memasukkan informasi umum, alamat dan kartu debit / kredit yang digunakan melalui *website*.

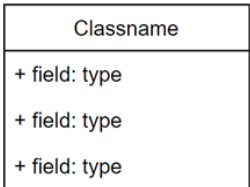



Tabel 2.4 *Fully developed use case description* (Satzinger, Jackson, Burd, 2016)

Use case name:	Create customer account.	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
Actors:	Customer.	
Related use cases:	Might be invoked by the <i>Check out shopping cart</i> use case.	
Stakeholders:	Accounting, Marketing, Sales.	
Preconditions:	Customer account subsystem must be available. Credit/debit authorization services must be available.	
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
Flow of activities:	Actor	System
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for credit/debit card.
	3. Customer enters credit/debit card information.	3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

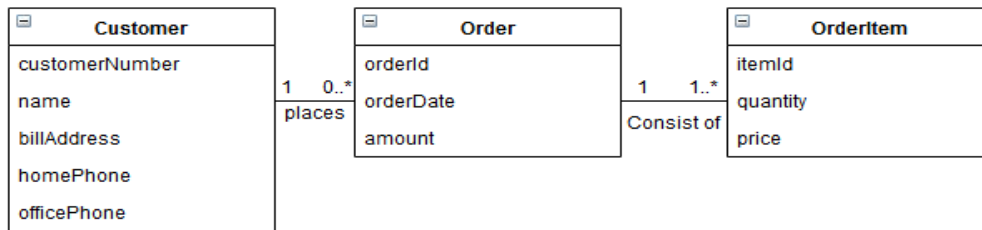
2.5.4 Domain Model Class Diagram

Domain Model Class Diagram merupakan salah satu jenis diagram UML yang terdiri dari suatu *class* berisi objek-objek yang memiliki karakteristik sama yang berperan untuk membantu pengembang mendapatkan struktur sistem dan menghasilkan rancangan sistem. Adapun simbol yang digunakan dalam *domain model class diagram* adalah sebagai berikut (Satzinger, Jackson, Burd, 2016) :

Tabel 2.5 Simbol *Domain Model Class Diagram* (Satzinger, Jackson, Burd, 2016)

No	Simbol	Deskripsi
1		<i>Class name</i> menunjukkan nama kelas, nama kelas dimulai dengan huruf kapital. Bagian bawahnya menunjukkan daftar atribut kelas. Nama atribut dimulai dengan huruf kecil.
2		<i>Zero or one (optional), zero or more (optional).</i>
3		<i>One and only one (mandatory), zero or more alternate (optional).</i>
4		<i>One and only one alternate (mandatory), one or more (mandatory).</i>

Pada Gambar 2.6 terdapat contoh sederhana dari *domain model class diagram* yang terkait dengan proses pemesanan barang. Proses tersebut memiliki tiga kelas, yaitu *Customer*, *Order*, dan *Order Item*. Setiap *Customer* dapat melakukan banyak pesanan. Setiap pesanan ditempati oleh satu pelanggan dimana pesanan tersebut memiliki satu atau banyak barang.



Gambar 2.6 Contoh *Domain Model Class Diagram*

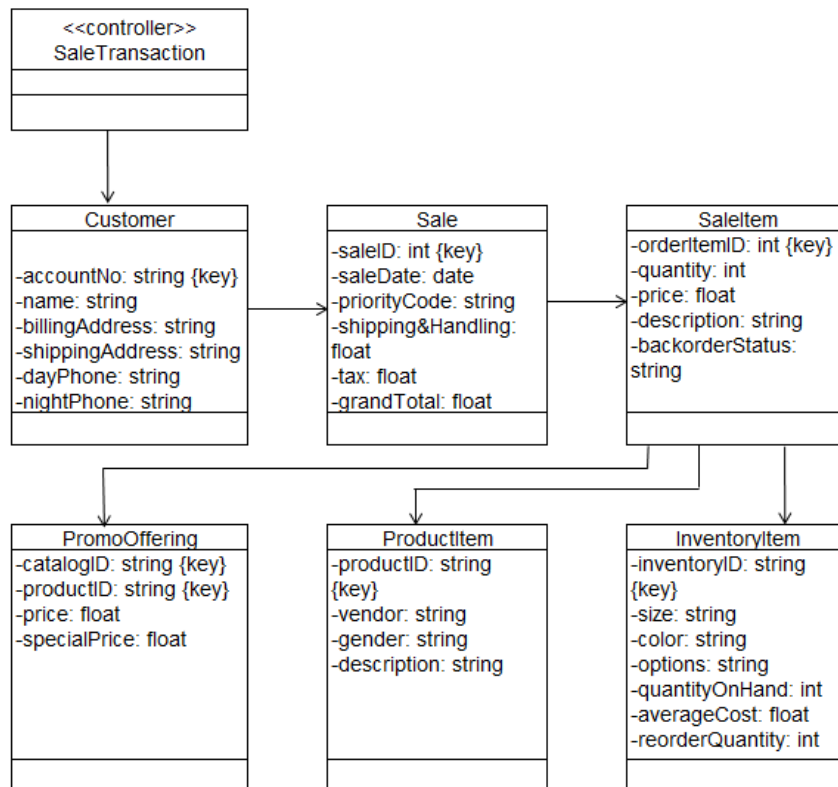
(Satzinger, Jackson, Burd, 2016)

Domain model class diagram memiliki pengembangan lanjutan, diantaranya adalah sebagai berikut :

a. *First-Cut Domain Model Class Diagram*

First-cut domain class diagram digunakan untuk mengidentifikasi kelas dengan menguraikan atribut setiap kelas sehingga dapat diketahui kelas apa saja yang membutuhkan navigasi ke kelas lain berdasarkan informasi yang sudah tersedia dan mengidentifikasi kelas mana saja yang dapat melaksanakan *use case* (Satzinger, Jackson, Burd, 2016).

Pada Gambar 2.7 terdapat contoh *First-cut domain model class diagram* yang menjelaskan mengenai proses penjualan ponsel. Terdapat satu kelas desain tambahan yaitu *SaleHandler* sebagai kelas pengontrol dengan fungsi sebagai kelas utilitas untuk membantu pemrosesan *use case*.



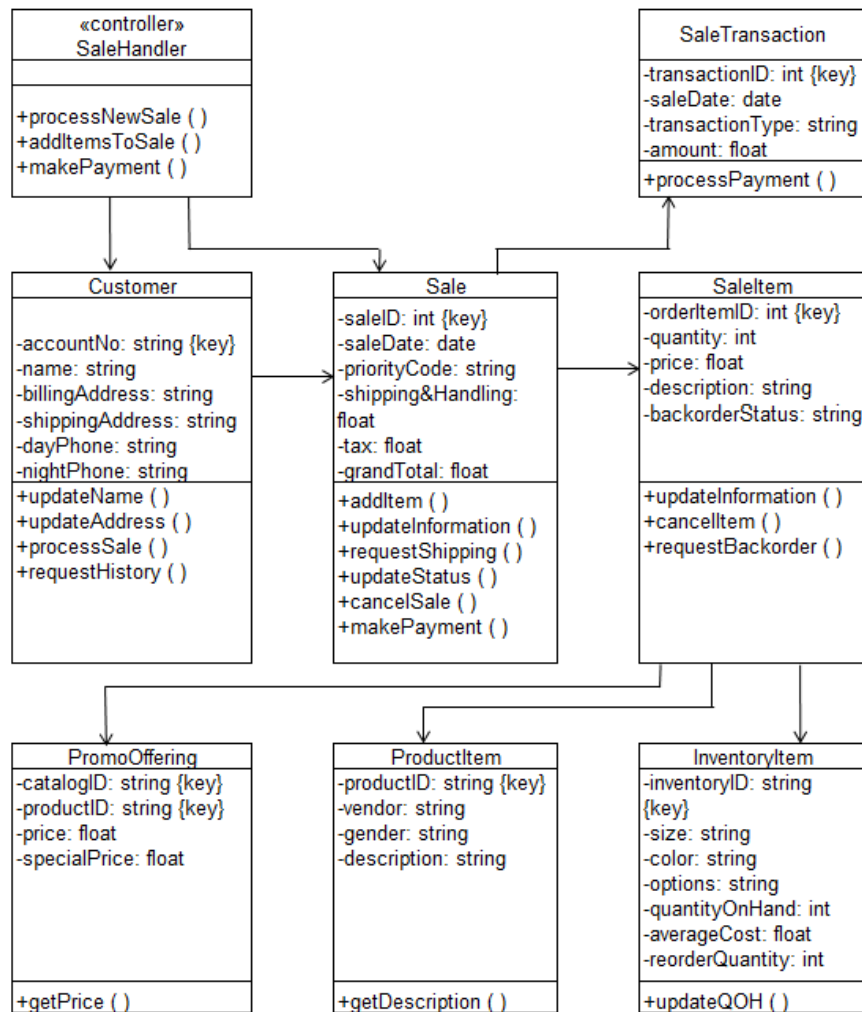
Gambar 2.7 Contoh *First-Cut Domain Model Class Diagram*

(Satzinger, Jackson, Burd, 2016).

b. *Update Design Class Diagram*

Updated design class diagram adalah *design class diagram* yang telah diperbarui metode dan visibilitasnya dalam setiap *use case*. Sehingga diagram ini menjadi pusat semua informasi setiap kelas dalam sebuah sistem (Satzinger, Jackson, Burd, 2016).

Pada Gambar 2.8 terdapat contoh dari *Update design class diagram create ponsel use case* yang menunjukkan bahwa dengan adanya *update method* dan visibilitas memunculkan penambahan kelas baru.





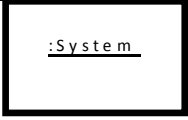

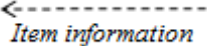
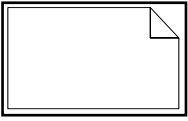

Gambar 2.8 Contoh Update Design Class Diagram

(Satzinger, Jackson, Burd, 2016).

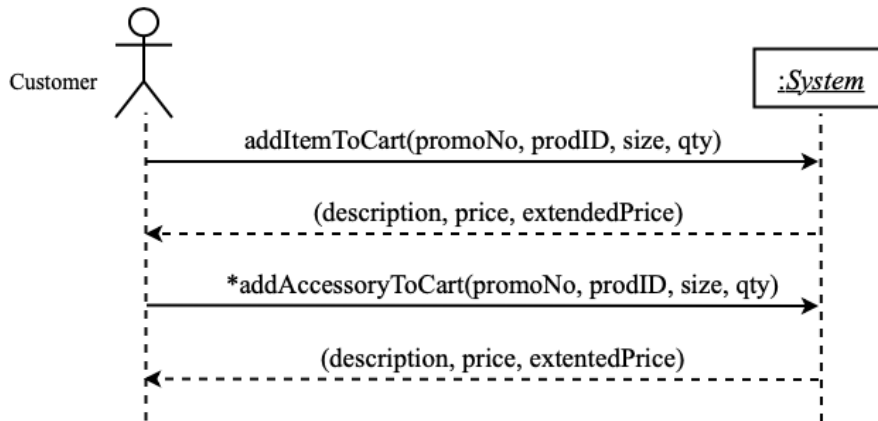
2.5.5 System Sequence Diagram

System sequence diagram (SSD) merupakan diagram yang menjelaskan urutan aliran informasi yang masuk dan keluar dari sebuah sistem. System sequence diagram (SSD) dapat mendokumentasikan input, output dan mengidentifikasi interaksi antara actor dan sistem. SSD termasuk bagian dari diagram interaksi, yaitu diagram yang menunjukkan interaksi antara objek. Adapun simbol yang digunakan dalam system sequence diagram adalah sebagai berikut (Satzinger, Jackson, Burd, 2016) :

Tabel 2.6 *System Sequence Diagram* (Satzinger, Jackson, Burd, 2016)

No	Simbol	Deskripsi
1	 Actor	<i>Actor</i> , menunjukkan sebagai orang yang berinteraksi dengan sistem.
2	 <i>An input message</i>	<i>An input message</i> , informasi atau pesan masuk yang diterima oleh sistem. Pesan diberi label untuk menjelaskan tujuan serta input data yang dikirim. Nama pesan harus mengikuti sintaks kata benda untuk membuat tujuannya jelas.
3	 :System	: <i>System</i> , adalah objek yang mewakili keseluruhan sistem otomatis.
4		<i>Lifeline</i> atau <i>object lifeline</i> , menunjukkan berlalunya waktu untuk objek serta "urutan" pesan, dari atas ke bawah.
5	 <i>Item information</i>	<i>A returned value</i> , menunjukkan informasi atau pesan yang dibawa dikembalikan mengikuti pesan awal. Format labelnya juga berbeda. Hanya data yang dikirim pada respons yang dicatat.
6		Catatan <i>opsional</i> untuk menjelaskan sesuatu dalam diagram.
7		<i>Activation</i> , sebagai indikasi bahwa sebuah obyek akan melakukan sebuah aksi.

Pada Gambar 2.9 terdapat contoh dari *system sequence diagram* (SSD) penggunaan isi keranjang belanja dari seorang *customer* yang sedang melakukan pemilihan produk kemudian memasukkan belanjaan ke dalam keranjang belanja pada sebuah sistem.



Gambar 2.9 Contoh *System Sequence Diagram*

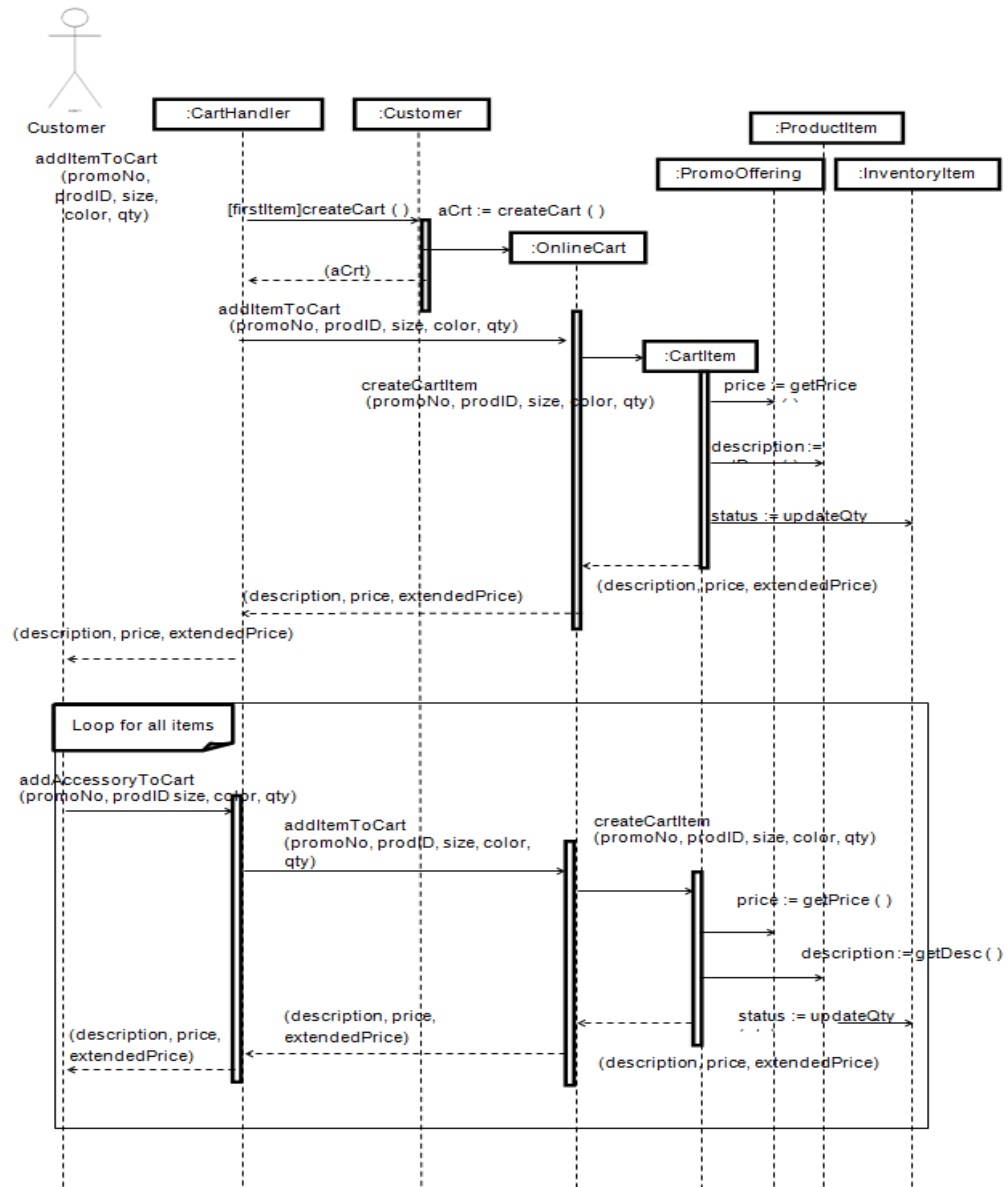
(Satzinger, Jackson, Burd, 2016)

System sequence diagram memiliki pengembangan lanjutan (*developing multilayer design*), di antaranya adalah sebagai berikut :

a. *First-Cut Sequence Diagram*

Sequence diagram yang rinci, menggunakan semua elemen yang ada dalam *sequence diagram*. *Firs-cut sequence diagram*, digunakan untuk mengidentifikasi masalah yang ada dalam pesan dan *class domain*. Dalam proses identifikasi ini, objek harus menentukan asal dan tujuan objek serta dapat mengirimkan suatu pesan. Pesan tersebut harus mencerminkan permintaan yang dikirim (Satzinger, Jackson, Burd, 2016).

Pada Gambar 2.10 menunjukkan bahwa pesan-pesan yang ada dalam *First-cut sequence diagram* menjelaskan tentang penggunaan keranjang belanja dimana pesan-pesan yang ada hanyalah duplikat dari pesan-pesan sebelumnya dengan perubahan-perubahan kecil yang menyertainya.



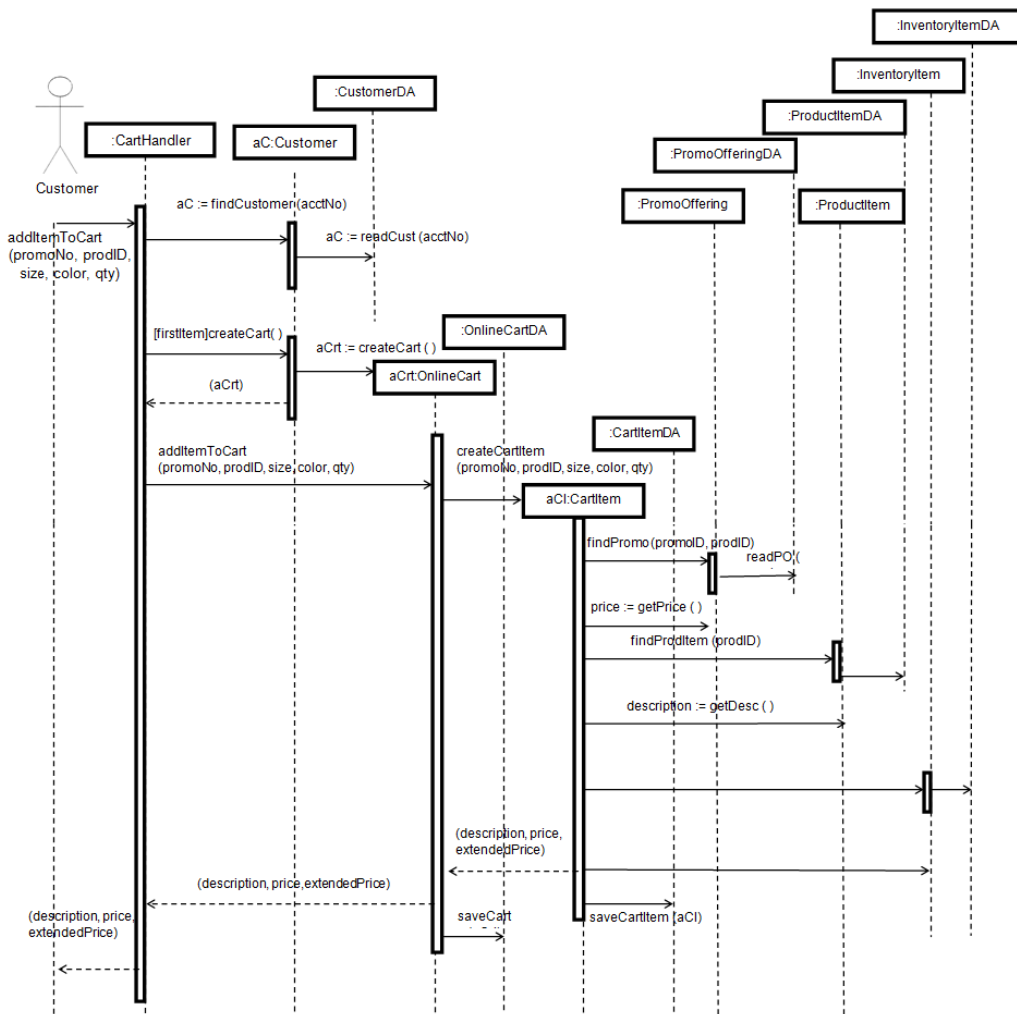
Gambar 2.10 Contoh *First-Cut Sequence Diagram*

(Satzinger, Jackson, Burd, 2016)

b. Data Access Layer

Data access layer merupakan salah satu bagian dari *three layer design* yang berhubungan dengan *database*. *Data access layer* diperlukan pada saat proses sistem yang dikembangkan cukup kompleks. *Data access layer* memiliki peran untuk mengelola data yang disimpan pada satu atau lebih *database* (Satzinger, Jackson, Burd, 2016).

Pada Gambar 2.11 terdapat contoh dari *data access layer* yang menunjukkan bahwa dalam diagram *sequence* mencakup *class domain* dan kelas *data access*. Untuk memahami *data access layer* dapat dimulai dengan melihat pesan yang ada dalam *sequence diagram*.



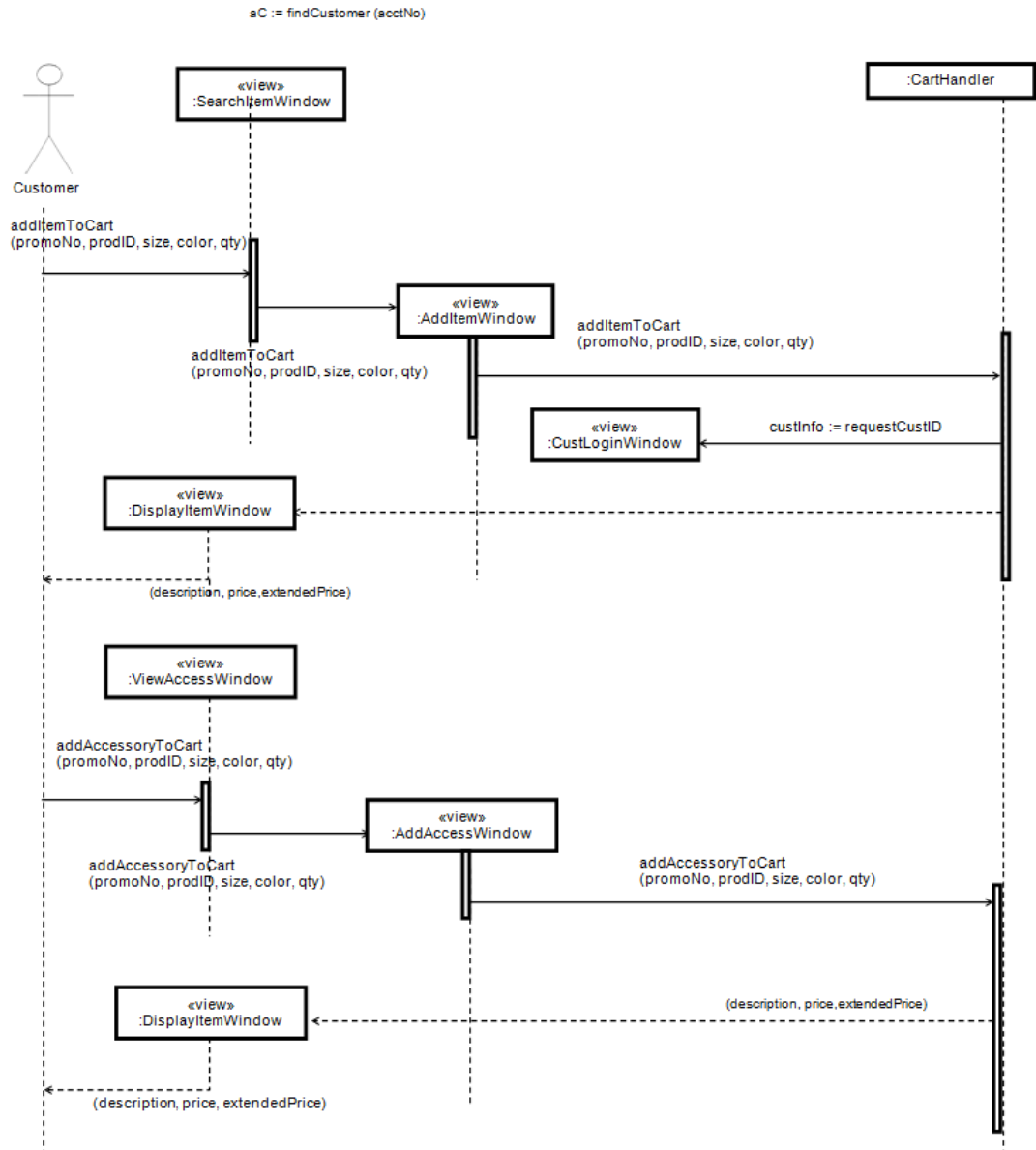
Gambar 2.11 Contoh *Data Access Layer*

(Satzinger, Jackson, Burd, 2016)

c. View Layer

View Layer merupakan langkah akhir dari *developing multilayer design* yang berhubungan dengan *user interface* yang lebih kompleks. *View layer* bertugas untuk menerima *input user* dan menampilkan hasil dari proses sistem (Satzinger, Jackson, Burd, 2016).

Pada Gambar 2.12 terdapat contoh *view layer* yang menunjukkan bahwa ada dua *input* untuk *view layer* yaitu komponen *user interface* dan *sequence diagram* yang sudah teridentifikasi kelas akses datanya.



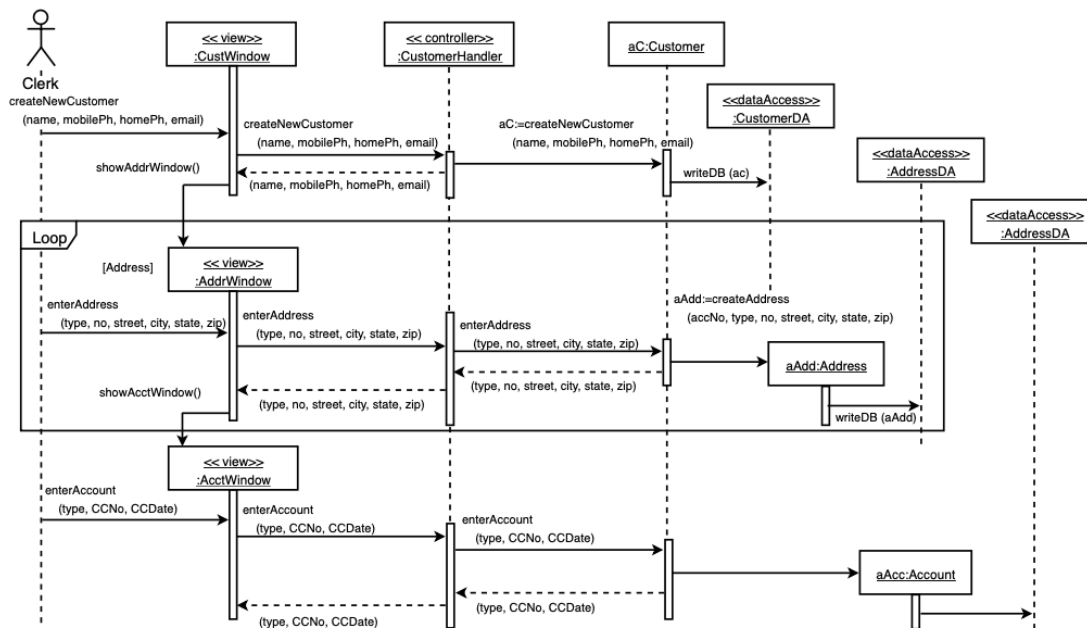
Gambar 2.12 Contoh *View Layer*

(Satzinger, Jackson, Burd, 2016)

d. *Multi Layer*

Multi Layer memiliki tujuan utama untuk mengidentifikasi kolaborasi kelas dan apakah kelas tersebut harus mengirim pesan antara satu sama lain (Satzinger, Jackson, Burd, 2016). *Multi Layer* menyediakan sebuah dasar yang sempurna untuk membuat program dari *use case* yang ada. Melalui proses desain yang *detail*, perancang dapat berpikir melalui kompleksitas dari setiap *use case* tanpa adanya komplikasi dalam membuat program.

Pada Gambar 2.13 terdapat contoh *multi layer* yang terdiri dari *data access layer* dan *view layer* untuk memastikan bahwa *user interface* yang dikembangkan konsisten dengan desain aplikasi dimana semua ada pada *system sequence diagram*.



Gambar 2.13 Contoh *Multi Layer*

(Satzinger, Jackson, Burd, 2016)

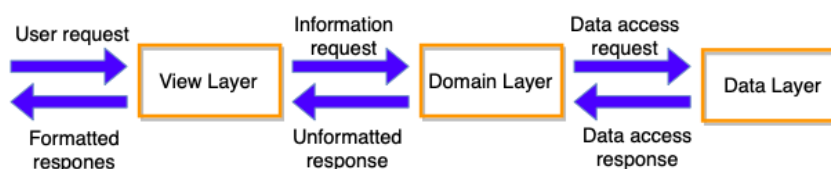
e. *Three-Layer Client-Server Architecture*

Three-Layer Client-Server Architecture merupakan salah satu metode desain perangkat lunak yang efektif untuk memisahkan rutinitas antarmuka pengguna dari rutinitas logika bisnis dan memisahkan rutinitas logika bisnis dari rutinitas

akses database. Metode perancangan lunak aplikasi ini disebut *Three-Layer Client-Server Architecture* (Satzinger, Jackson, Burd, 2016).

Pada Gambar 2.14 *Three-Layer Client-Server Architecture* membagi aplikasi lunak menjadi tiga lapisan, yaitu :

- *User Interface* atau *View Layer*, yang menerima *input* dan format pengguna dan menampilkan hasil pemrosesan.
- *Business Logic* atau *Domain Layer*, yang mengimplementasikan aturan dan prosedur pemrosesan bisnis.
- *Data Layer*, yang mengelola data yang disimpan dan biasanya dalam satu atau lebih *database*.



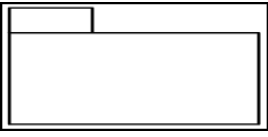


Gambar 2.14 Contoh *Three-Layer Client-Server Architecture*

(Satzinger, Jackson, Burd, 2016)

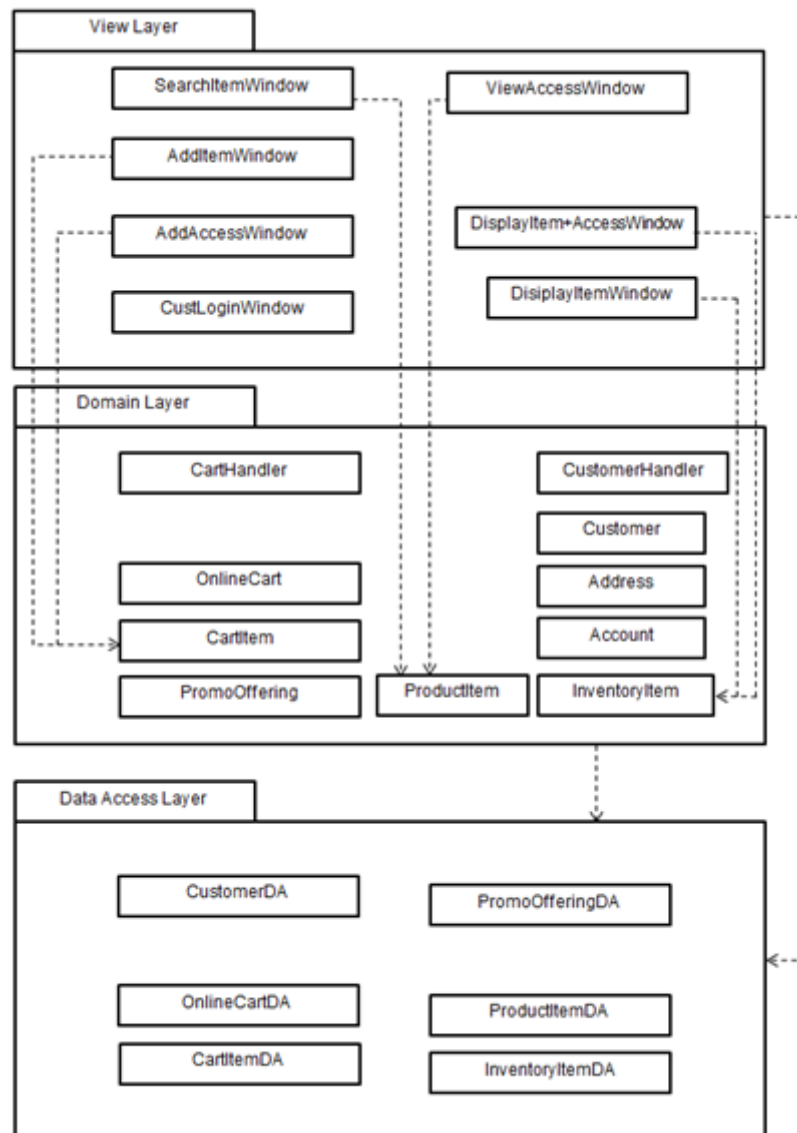
2.5.6 Package Diagram

Package diagram di UML merupakan diagram tingkat tinggi yang memungkinkan desainer untuk mengaitkan kelas-kelas grup terkait. *Package diagram* dapat digunakan untuk mengelompokkan semua jenis elemen desain di UML. Umumnya, *package diagram* menghubungkan kelas atau komponen sistem lainnya seperti *node* jaringan. Adapun simbol yang digunakan dalam *package diagram* adalah sebagai berikut (Satzinger, Jackson, Burd, 2016) :

Tabel 2.7 Simbol *Package Diagram* (Satzinger, Jackson, Burd, 2016)

No	Simbol	Deskripsi
1		<i>Package</i> , kelompok kelas yang memiliki persamaan dan saling berkaitan.
2		<i>Class</i> , terdiri dari elemen-elemen yang ditempatkan di dalam paket yang sesuai.
3		<i>Dependency Relationship</i> , menunjukkan hubungan antara <i>package</i> , <i>class</i> atau <i>use case</i> yang saling bergantung, apabila perubahan dalam item independen maka harus dilakukan perubahan pada item dependen.

Pada Gambar 2.15 terdapat contoh *package diagram* RMO (*Ridgeline Mountain Outfitters*) yang menunjukkan bahwa perbedaan atau persamaan yang terhimpun dalam sebuah lapisan berdasarkan lingkungan pemrosesan, dapat saling terhubung.



Gambar 2.15 Contoh *Package Diagram* (Satzinger, Jackson, Burd, 2016)

2.5.7 Persistent Object

Persistent Object adalah kelas entitas yang objeknya ada setelah sistem atau program berhenti. Nilai datanya harus disimpan oleh sistem bahkan ketika aplikasi tidak dijalankan. Objek yang tersedia pada sistem untuk diingat dan digunakan dari waktu (Satzinger, Jackson, Burd, 2016).

Pada Gambar 2.16 terdapat contoh *persistent object create phone sell* yang menunjukkan bahwa masing-masing *persistent class* akan memiliki akses kelas data untuk membaca dan menulis di *database*. Dalam contoh ini, *persistent objectnya* adalah *SaleHandler*, *Customer*, *Sale*, *SaleItem*, dan *PromoOffering*.

CatalogID	ProductID	Price	SpecialPrice
23	1244	\$15.00	\$12.00
23	1245	\$15.00	\$12.00
23	1246	\$15.00	\$13.00
23	1247	\$15.00	\$13.00
23	1248	\$14.00	\$11.20
23	1249	\$14.00	\$11.20
23	1252	\$21.00	\$16.80
23	1253	\$21.00	\$16.40
23	1254	\$24.00	\$19.20
23	1257	\$19.00	\$15.20

Gambar 2.16 Contoh Diagram *Persistent Object*

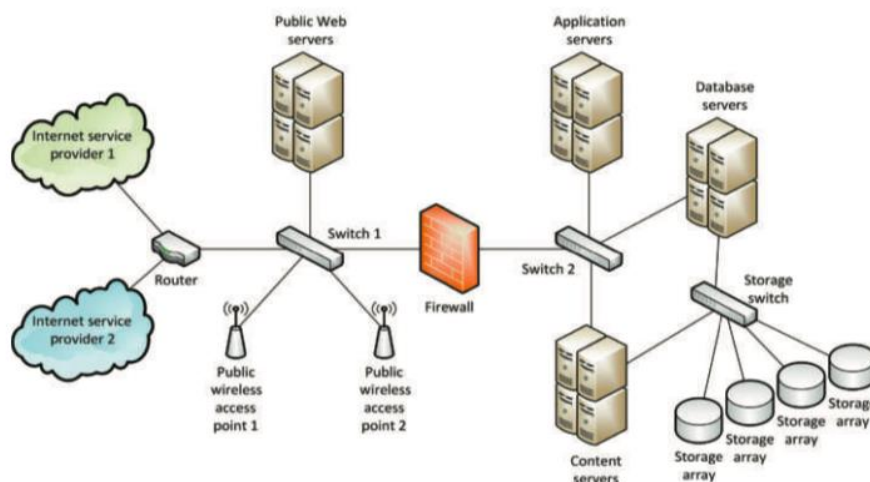
(Satzinger, Jackson, Burd, 2016)

2.6 *Environment Design*

2.6.1 *Network Diagram*

Network Diagram menunjukkan bagaimana lokasi dan komponen perangkat keras saling terhubung dengan perangkat jaringan dan kabel. Terdapat berbagai jenis diagram jaringan dimana masing-masing menekankan aspek yang berbeda seperti *network*, *connected hardware resources*, and *users* (Satzinger, Jackson, Burd, 2016).

Pada Gambar 2.17 menunjukkan *Network Diagram* lain berfokus lebih khusus pada koneksi jaringan dan perangkat keras jaringan. Diagram jenis ini biasanya digunakan untuk menggambarkan koneksi jaringan gedung atau di dalam ruang server.



Gambar 2.17 Contoh *Network Diagram*

(Satzinger, Jackson, Burd, 2016)

2.7 Website

World Wide Web adalah jaringan yang terus berkembang pesat dan sudah sangat jauh melampaui konsepsinya. Awal 1990-an, *web* diciptakan untuk memecahkan masalah tertentu. Ketika eksperimen canggih di CERN (*European Laboratory for Particle Physics* yang sekarang dikenal sebagai operator *Large Hadron Collider*) memproduksi jumlah data yang sangat banyak. Akhirnya untuk mendistribusikan, data dibagikan melalui *internet* untuk disampaikan kepada ilmuwan yang berpartisipasi yang tersebar di seluruh dunia (Nixon, 2014).

Hypertext Transfer Protocol (HTTP) adalah standar komunikasi yang mengatur permintaan dan respon antara *browser* yang berjalan di komputer pengguna dan *server web*. Pekerjaan *server* adalah menerima permintaan dari klien dan berusaha memberikan respon, biasanya dengan menyajikan halaman *web* yang diminta. Antara klien dan *server* mungkin ada beberapa perangkat lain seperti *router*, *proxy*, *gateway*, dan sebagainya. Mereka melayani peran yang berbeda dalam memastikan permintaan tersebut dan tanggapan dikirimkan dengan benar antara klien dan *server*. Biasanya, mereka menggunakan *internet* untuk mengirim informasi ini (Nixon, 2014).

2.7.1 HTML

HyperText Markup Language (HTML) didefinisikan sebagai bahasa *markup*. Bahasa *markup* adalah suatu cara untuk memberikan keterangan di sebuah dokumen dengan cara membuat teks yang ditambahkan tersebut dapat diketahui. Bahasa *markup* seperti HTML, XML, dan XHTML mengizinkan penggunaannya untuk mengontrol bagaimana teks dan elemennya diletakkan dan ditampilkan ke pengguna. Sehingga, *markup* menjadi sebuah cara untuk mengindikasikan informasi mengenai konten yang tampil berbeda dari konten aslinya (Nixon, 2014).

Informasi dalam konten di HTML diimplementasikan menggunakan *tag* yang ditandai dengan "< >", contoh : `<button>` yang diartikan sebagai tombol. Dalam menjelaskan informasi dalam sebuah konten, banyak bahasa *markup* yang dapat membuat tampilan informasi berbeda sesuai keinginan *programmer* untuk dapat ditampilkan ke pengguna. Tampilan tersebut bisa berupa pembuatan tulisan yang menjadi lebih tebal, maupun format huruf yang berbeda (Nixon, 2014).

2.7.2 CSS

Cascading Style Sheets (CSS) merupakan standar W3C untuk menggambar tampilan dari elemen-elemen yang ada di HTML. Salah satu cara paling umum dalam menjelaskan fungsi CSS adalah untuk mempresentasikan dokumen HTML. Dengan CSS, pengguna dapat mengatur huruf, warna, ukuran, garis, latar Gambar, hingga posisi suatu elemen di dalam sebuah halaman. CSS dapat ditambahkan langsung ke semua elemen HTML menggunakan atribut *style* di dalam elemen `<head>` di HTML. Bisa juga dengan membuat *file* terpisah yang berisikan CSS tersebut, namun masih mempunyai hubungan dengan halaman HTML tersebut (Nixon, 2014).

Dengan menggunakan CSS, kita dapat menerapkan gaya ke halaman *web* untuk membuatnya terlihat persis seperti yang kita inginkan. Ini berfungsi karena CSS terhubung ke DOM (*Document Object Model*). Integrasi antara CSS dan DOM, dapat mempercepat dan memudahkan kita dalam menata ulang elemen-elemen yang ditampilkan. Misalnya, jika kita tidak menyukai tampilan asli dari *tag heading* `<h1>`, `<h2>`, dan lainnya, kita dapat menetapkan gaya baru untuk mengganti pengaturan huruf maupun ukuran teks yang digunakan,

selain itu kita juga bisa menambahkan properti teks dengan huruf tebal ataupun huruf miring (Nixon, 2014).

2.7.3 *Javascript*

JavaScript dibuat untuk mengaktifkan akses *scripting* ke semua elemen dokumen HTML. Dengan kata lain, ini menyediakan sarana untuk interaksi pengguna yang dinamis seperti memeriksa validitas alamat *e-mail* dalam *input* formulir atau menampilkan pesan *pop-up*. Namun cukup rumit untuk menggunakannya karena *browser* didesain secara berbeda sehingga mengimplementasikan hasilnya pun antara satu *browser* dengan lainnya akan berbeda. Begitupun bila dengan versi *browser* yang lebih lama, belum tentu kompatibel dengan *script* yang dibuat (Nixon, 2014).

Javascript juga digunakan sebagai AJAX (*Asynchronous JavaScript and XML*), dimana fungsinya bisa mengakses *server web* di latar belakang. AJAX adalah proses utama di balik apa yang sekarang dikenal sebagai Web 2.0. Halaman *web* sudah mulai menyerupai program mandiri karena tidak harus selalu dimuat ulang secara keseluruhan tapi cukup bagian tertentu saja (Nixon, 2014).

2.7.4 PHP

Hypertext Preprocessor (PHP) adalah bahasa pemrograman yang digunakan untuk membuat *server* menghasilkan *output* secara dinamis. *Output* yang ditampilkan berpotensi berbeda setiap kali *browser* melakukan *request* halaman (Nixon, 2014).

Umumnya, dokumen PHP diakhiri dengan ekstensi *.php*. Ketika *server web* bertemu ekstensi ini dalam *file* yang diminta, secara otomatis meneruskannya ke prosesor PHP. Tentu saja, *server web* sangat dapat dikonfigurasi. Beberapa pengembang *web* memilih untuk memaksakan *file* yang diakhiri dengan *.htm* atau *.html* juga dapat diuraikan oleh prosesor PHP, biasanya karena mereka ingin menyembunyikan fakta bahwa mereka menggunakan bahasa pemrograman PHP (Nixon, 2014).

2.8 Database

Database adalah kumpulan catatan atau data terstruktur yang disimpan dalam sistem komputer dan diatur sedemikian rupa sehingga dapat dengan cepat dicari dan informasi dapat dengan cepat diambil. SQL di MySQL adalah singkatan dari *Structured Query Language*. Bahasa ini digunakan bebas berdasarkan bahasa Inggris dan juga digunakan dalam *database* lain seperti Oracle dan Microsoft SQL Server (Nixon, 2014).

2.8.1 DBMS

Database Management System (DBMS) adalah perangkat lunak yang berinteraksi dengan program aplikasi dan data pengguna. Umumnya, DBMS menyediakan fasilitas sebagai berikut (Connolly & Begg, 2014).

1. *Data Definition Language* (DDL)

Fasilitas yang memungkinkan pengguna untuk mendefinisikan basis data. DDL memungkinkan pengguna untuk menentukan tipe dan struktur data serta *constraint* pada data yang akan disimpan dalam *database*.

2. *Data Manipulation Language* (DML)

Fasilitas yang memungkinkan pengguna untuk memasukkan, memperbarui, menghapus, dan mengambil data dari *database*. DML memiliki repositori pusat untuk semua data dan deskripsi data sehingga DML bisa melakukan permintaan data melalui bahasa *query* yang umum dikenal sebagai *Structured Query Language* (SQL). Saat ini SQL merupakan bahasa standar formal untuk DBMS relasional.

3. Menyediakan akses terkontrol ke *database*. Misalnya, ini dapat memberikan :

- a. sistem keamanan, yang mencegah pengguna yang tidak sah mengakses *database*.
- b. sistem integritas, yang menjaga konsistensi data yang disimpan.
- c. sistem kontrol konkurensi, yang memungkinkan akses bersama dari *database*.
- d. sistem kontrol pemulihan, yang mengembalikan *database* ke konsistensi sebelumnya setelah terjadi kegagalan baik pada *hardware* maupun *software*.

- e. katalog yang dapat diakses pengguna, yang berisi deskripsi data dalam *database*.

2.8.2 MySQL

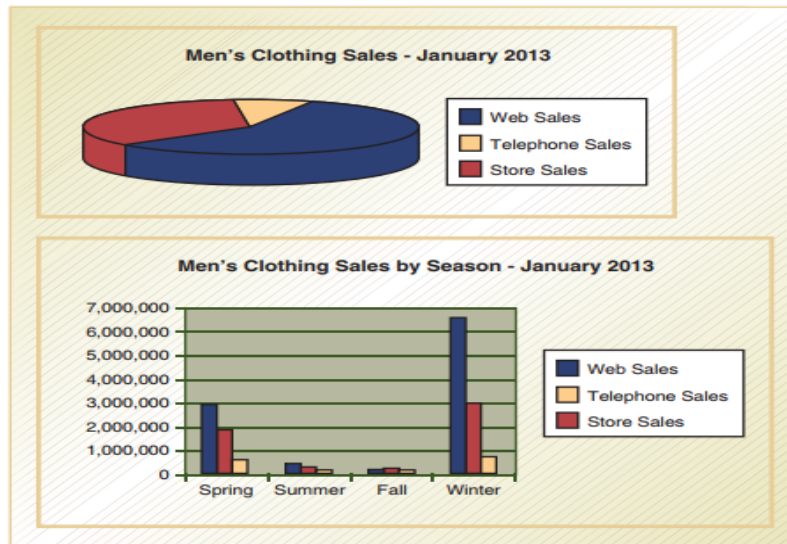
My Structured Query Language (MySQL) merupakan salah satu basis data yang paling populer dalam sistem manajemen berbasis *web*. Dikembangkan pada pertengahan 1990-an, kini sudah lebih dari 10 juta instalasi telah menjadikannya teknologi yang cukup mumpuni dan sangat matang. Salah satu alasan keberhasilannya adalah karena tersedia gratis. Selain itu cukup kuat dan cepat responnya, hampir tidak mengurangi sumber daya sistem dan *hardware* (Nixon, 2014).

Ketika memutuskan untuk membangun *web* dinamis dan bukan hanya halaman HTML statis, kita mungkin perlu menggunakan perangkat lunak *database* relasional yang mampu menjalankan query SQL. Biasanya DBMS *open-source* yang digunakan adalah MySQL. Walaupun terhitung gratis, MySQL cukup mumpuni untuk situs *web* dengan data yang besar serta lalu lintas data yang padat. Beberapa *website* ternama yang menggunakan MySQL seperti Facebook dan Flickr (Nixon, 2014).

2.9 System Interface

System interface atau antarmuka sistem adalah mekanisme *input* dan *output* sangat sedikit membutuhkan intervensi manusia. Mekanisme *input* ditangkap secara otomatis oleh perangkat khusus seperti *scanner*, pesan elektronik dari dan yang ditujukan ke sistem lain, atau transaksi yang diambil oleh sistem lain. Sedangkan *output* dianggap sebagai antarmuka sistem jika mereka mengirim pesan atau informasi ke sistem lain misalnya dalam bentuk notifikasi. *Output* bisa juga menghasilkan laporan, pernyataan, atau dokumen untuk agen eksternal atau aktor tanpa banyak intervensi manusia seperti contohnya *e-statement* kartu kredit yang dikirim ke alamat *e-mail* pemegang kartu (Satzinger, Jackson, Burd, 2016).

Berikut ini contoh *System Interface output* berupa laporan penjualan pakaian pria periode Januari 2013 berdasarkan kategori media penjualan :



Gambar 2.18 Contoh *System Interface* (Satzinger, Jackson, Burd, 2016).

2.10 *User Interface*

User Interface atau antarmuka pengguna adalah mekanisme *input* dan *output* yang secara langsung melibatkan pengguna sistem. Antarmuka pengguna dapat dilakukan oleh pengguna internal ataupun eksternal. Desainnya bervariasi sangat tergantung pada faktor-faktor seperti tujuan antarmuka, karakteristik pengguna, dan karakteristik perangkat antarmuka tertentu. Misalnya, meskipun semua antarmuka pengguna harus dirancang untuk kemudahan penggunaan maksimal, ada pertimbangan lain, seperti efisiensi operasional, mungkin penting bagi pengguna internal yang dapat dilatih menggunakan antarmuka spesifik yang dioptimalkan untuk perangkat perangkat keras tertentu misalnya *keyboard*, *mouse*, dan layar besar beresolusi tinggi (Satzinger, Jackson, Burd, 2016).

Sebaliknya, antarmuka pengguna lainnya akan sangat mungkin berbeda pada sistem terkait pelanggan, yang mengasumsikan ponsel sebagai perangkat *input* dan *output*. Dalam sebagian besar proyek pengembangan sistem, analisis memisahkan desain sistem antarmuka dari desain antarmuka pengguna karena masing-masing memerlukan keahlian dan teknologi tersendiri. Namun seperti halnya dengan desain komponen sistem apapun, diperlukan koordinasi yang besar (Satzinger, Jackson, Burd, 2016).

Adapun delapan aturan penting untuk merancang layar antarmuka yang interaktif dan fungsional yaitu (Satzinger, Jackson, Burd, 2016) :

1. *Affordance and Visibility*

Tampilan menu harus jelas serta dapat digunakan secara maksimal.

2. *Consistency*

Merancang tampilan yang konsisten dan antarmuka yang fungsional menjadi sangat penting. Pengaturan *form*, nama, serta menu, ukuran dan bentuk ikon-ikon serta alur dari sistem harus konsisten dan diketahui secara spesifik fungsinya sehingga bisa dipakai oleh pengguna.

3. *Shortcut*

Umumnya pengguna yang sudah sering menggunakan aplikasi lebih menginginkan kecepatan dalam mengakses informasi yang diinginkan.

4. *Feedback*

Umpan balik dibutuhkan untuk memberikan informasi kepada pengguna sesuai dengan aksi yang dilakukannya.

5. *Dialogs That Yield Closure*

Urutan tindakan sebaiknya diatur di dalam suatu kelompok bagian awal, tengah dan akhir. Umpan balik yang diberikan akan memberitahukan pengguna sistem bahwa tindakan yang dilakukan sudah benar dan dapat melanjutkan ke tindakan berikutnya.

6. *Error Handling*

Sistem dirancang untuk mencegah pengguna sistem agar tidak melakukan kesalahan fatal. Jika terjadi, maka sistem dapat langsung memberikan pencegahan kesalahan dengan cepat dan memberikan mekanisme yang simpel dan mudah dipahami oleh pengguna sistem.

7. *Easy Reversal of Actions*

Sistem dirancang untuk tidak menyulitkan pengguna. Pengguna sistem dibuat untuk tidak takut akan pilihan menu-menu baru karena adanya menu *undo* atau *back* dimana memungkinkan pengguna untuk melakukan tindakan kembali jika salah melakukan tindakan.

8. *Reduce Short-Term Memory Load*

Pengguna tidak disulitkan dengan menu-menu yang banyak di dalam sistem atau aplikasi sehingga dapat melakukan tindakan dengan memilih

menu yang simpel tanpa harus mengingat semua perintah atau fungsi menu-menu sistem.

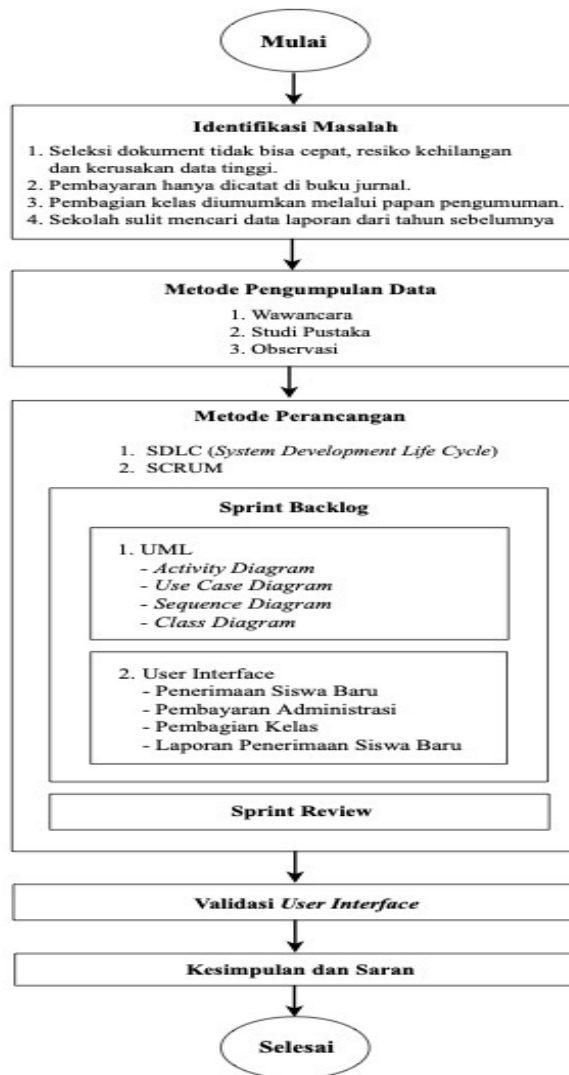
Berikut ini contoh *User Interface form* detail produk *RidgeLine Mountain Outfitters* (RMO) dengan tampilan seperti kontrol form *Microsoft Windows*.



Gambar 2.19 Contoh *User Interface* (Satzinger, Jackson, Burd, 2016).

2.11 Kerangka Pemikiran

Berikut ini merupakan tahapan penelitian dari perancangan sistem informasi Penerimaan Siswa Baru pada SMK Gapura Kasih :



Gambar 2.20 Kerangka Pemikiran

a. Identifikasi Masalah

Hal pertama kali yang dilakukan dalam melaksanakan penelitian ini adalah melakukan identifikasi masalah dari tempat studi kasus penelitian yang akan dilaksanakan, yaitu sekolah SMK Gapura Kasih. Identifikasi masalah yang ditemukan adalah sebagai berikut :

1. Seleksi dokumen menggunakan media kertas yang ditulis dengan tangan tidak bisa dilakukan dengan cepat serta memiliki resiko kerusakan dan kehilangan data tinggi.
2. Pembayaran ditulis di buku jurnal seringkali terjadi kesalahan pencatatan karena lupa atau salah catat tanggal pembayaran, jumlah pembayaran dan nama pembayar.

3. Pembagian kelas diumumkan melalui papan pengumuman sehingga mengharuskan siswa baru rutin datang ke sekolah.

4. Sekolah kesulitan mencari data laporan dari tahun sebelumnya.

b. Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan yaitu dengan melakukan wawancara dengan pemilik Yayasan Langgeng Wahana yang menaungi SMK Gapura Kasih untuk mengetahui proses bisnis yang berlangsung. Kemudian melakukan studi pustaka berkaitan dengan referensi yang digunakan untuk teori-teori penelitian. Berikutnya, observasi terkait kebutuhan yang diperlukan pada objek penelitian.

c. Metode Perancangan

Metode perancangan menggunakan SDLC (*System Development Lifecycle*) dengan pendekatan adaptif, yang bisa menyesuaikan perubahan kebutuhan dan prioritas rancangan sistem. Selain itu juga menggunakan *Scrum* untuk pembagian tugas dalam perancangan ini menjadi lebih spesifik. Di dalam *scrum* terdapat *sprint backlog* yang berisi UML menggambarkan perancangan sistem yang terdiri dari *Activity Diagram*, *Use Case Diagram*, *Domain Class Diagram* dan *Sequence Diagram*. Untuk menggambarkan interaksi *user* dengan rancangan sistem, dibuat *design user interface* penerimaan siswa baru, pembayaran administrasi, pembagian kelas, dan laporan penerimaan siswa baru. Kemudian, terdapat *sprint review* yang menjelaskan tahap evaluasi *sprint* biasanya dilaksanakan oleh tim *Scrum* dan *Stakeholder* pada akhir periode *Sprint*.

d. Validasi *User Interface*

User Interface berfokus untuk memberikan kemudahan kepada pengguna (*user*) ketika berinteraksi dengan rancangan sistem. Sehingga diperlukan validasi apakah *design user interface* yang dibuat, sudah memenuhi kebutuhan *user* atau belum.

e. Kesimpulan dan Saran

Hasil akhir dari penelitian ini adalah perancangan sistem informasi penerimaan siswa baru pada SMK Gapura Kasih, nantinya dapat menjadi acuan dan dapat dikembangkan oleh SMK Gapura Kasih ke dalam aplikasi berbasis *web*.