BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Kajian Pustaka pada penelitian ini membahas mengenai penelitian terdahulu yang sebelumnya sudah pernah dilakukan. Pada tabel 2.1 merupakan pembahasan mengenai penelitian terdahulu yang mendukung penelitian saat ini untuk pembangunan sistem informasi rekam medis pada Klinik Mutiara Sehat.

***Tabel* 2.1 Pembahasan penelitian terdahulu**

Judul	**Uraian**	**Kesimpulan**
"Analisis dan Perancangan Sistem Informasi Rekam Medis Poli Gigi (Studi Kasus : Puskesmas Sumbersari Kecamatan Saradan Kabupaten Madiun)" yang telah dilakukan oleh Dzurriyatul Iflahah dan telah diterbitkan oleh Fakultas Ilmu Komputer Universitas Brawijaya.	Penelitian ini membahas mengenai analisis dan perancangan sistem informasi rekam medis dengan menggunakan pendekatan *Object Oriented Analysis and Design (OOAD).* Dalam proses analisis dan desainnya menggunakan bahasa pemodelan *Unified Modeling Language.*	Proses bisnis usulan didapatkan dari penambahan dan perubahan aktivitas proses bisnis. Analisis dan perancangan sistem inforamasi dalam penelitian ini terdiri dari kelas analisis dari *use case,* pemetaan kelas analisis ke mekanisme analisis, identifikasi elemen perancangan, perancangan *use case* berupa *sequence diagram,* perancangan kelas berupa *class diagram,* perancangan basis data dan perancangan antarmuka sistem yang ada disesuaikan dangan hasil analisis persyaratan yang dilakukan sebelumnya.

***Tabel* 2.1 Pembahasan penelitian terdahulu(lanjutan)**

Judul	**Uraian**	**Kesimpulan**
"Integrasi Antar Sistem Informasi yang Heterogen Menggunakan Metode Web Service SOAP" yang dilakukan oleh Muhamad fatkhur rahim dan diterbitkan oleh Fakultas Ilmu Komputer Universitas Dian.	Penelitian ini membahas integrasi antar dua sistem yaitu sistem Dinas Kependudukan dan Sistem Rumah Sakit dalam menggunakan data kependudukan untuk diintegrasikan dengan pasien dalam sistem informasi rumah sakit. Proses integrasi data penduduk antara sistem kependudukan dan sistem Rumah Sakit dibuat service untuk membagi data penduduk ke sistem lain dengan mengggunakan *web service* SOAP dengan komunikasi data menggunakan XML sehingga sistem Rumah Sakit cukup mengakses *service* tersebut untuk mendapatkan data penduduk.	Kesimpulan dari penelitian ini adalah dengan memanfaatkan metode web service SOAP, maka integrasi data diantara sistem yang heterogen tidak perlu mengubah sistem yang sudah ada meskipun diantara sistem memiliki konflik pada penggunaan DBMS yang berbeda, penamaan tabel yang berbeda dan struktur atribut yang berbeda.
"*Analysis of implementation of web services in e-learning* PDITT DIKTI" yang dilakukan oleh Nurul Hidayat dan Ahmad Ashari serta diterbitkan oleh *International Journal of Advance Eesearch in Engineering & Technology (IJARCET)*	Penelitian ini bertujuan untuk menganalisis *web service* dalam mengakses data PDIKT DIKTI yang memungkinkan antara platform aplikasi yang berbeda. Dalam penelitian ini, dilakukan analisis dan perancangan untuk mengimplementasikan *web service* pada DIKTI E-Learning, sehingga konten E-Learning yang diakses sesuai dengan *method* yang disediakan. Dalam implementasi pembangunan *web service* peneliti menggunakan *toolkit nuSOAP*.	Berdasarkan anailisi implementasi *web service* e-learning dapat disimpulkan bahwa *web service* dapat diaplikasikan pada *e-learning* sehingga memudahkan pengguna dalam pengembangan aplikasi untuk mengakses data sesuai dengan fungsi atau *method* yang disediakan di *e-Learning* PDITT DIKTI.

2.2 Klinik Mutiara Sehat

Klinik Mutiara Sehat merupakan klinik penyedia layanan fasilitas kesehatan tahap pertama yang dimana telah memiliki izin penyelenggaraan klinik rawat jalan sejak tanggal 18 maret 2013. Klinik yang bertempatan di Perum. PTP 23 Blik I/27 Karanglo, Malang ini memiliki 2 jenis poli yaitu poli umum dan poli gigi. Pasien yang datang sebagian besar merupakan pasien BPJS Kesehatan dan BPJS Ketenagakerjaan yang dimana mencapai hingga 80% perhari dibandingkan dengan pasien umum yang hanya 20%. Pada gambar 2.1 merupakan logo dari Klinik Mutiara Sehat.

Gambar 2.1 Logo Klinik Mutiara Sehat

Klinik Mutiara Sehat memiliki konsep layanan dengan motto service with Love yang senantias mengutamakan layanan dengan rasa kasih sayang layaknya terhadap keluarga sendiri sehingga tercipta suasana nyaman bagi pasien. Lokasi klinik yang terletak di daerah perumahan menjadikan sebuah karakteristik sendiri bagi klinik yang dimana susasan jauh dari kebisingan serta akses yang mudah di jangkau.

Visi Klinik Mutiara Sehat

Menjadi Klinik kesehatan yang memberikan pelayanan terbaik.

Misi Klinik Mutiara Sehat

- Memberikan pelayanan yang baik dan bermutu didasari kasih dan pengabdian kepada sesama.
- Menciptakan susana kerja yang dilandasi oleh rasa kekeluargaan antar tenaga medis dengan pasien
- Menumbuhkan kesadaran budaya hidup sehat pada semua orang

Falsafah

Pelayanan kesehatan diselenggarakan dengan berlandaskan etika dan profesionalisme

Motto

Service with Love

2.3 Sistem informasi

Sistem informasi merupakan media untuk menyajikan informasi sedemikian rupa sehingga bermanfaat bagi penerimanya. Tujuan dari sistem informasi yaitu untuk menyajikan informasi dalam perencanaanm memulai, pengorganisasian dan operasional sebuah perusahaan yang melayani sinergi organisasi dalam proses mengendalikan pengembilan keputuasan (Kertahadi & Astuti, 2007).

Fungsi pada Sistem informasi yaitu : untuk meningkatkan aksesbilitas data yang tersedia dapat terdistribusi dengan efektif dan efisien langsung kepada pengguna (*user*). memperbaiki produktivitas aplikasi pengembangan sistem beserta pemeliharaan sistem. Menjamin adanya kualitas serta keterampilan untuk memanfaatkan sistem informasi secara kritis dan mengembangkan proses perancanaan yang efektif.

Didalam sistem informasi terdapt beberapa komponen yaitu: (1) Komponen input adalah data yang dimasukan kedalam Sistem informasi (2) kompinen model adalah gabungan dari logika, prosedur dan moedel matematika untuk memproses data yang ada pada basis data. (3) Komponen Output adalag hasil keluaran dari sistem itu sendiri berupa informasi dan dokumentasi yang berguna bagi perusahaan. Sistem informasi bisa dikelompokan menjadi empat bagian yaitu sistem informasi manajemen, sistem pendukung keputusan, sistem informasi eksekutif dan sistem pemerosesan transaksi.

2.4 Proses Bisnis

Proses bisnis adalah suatu kumpulan aktivitas atau pekerjaan terstruktur yang saling terkait untuk menyelesaikan suatu masalah tertentu atau yang menghasilkan produk atau layanan demi meraih tujuan tertentu. Menurut Weske (2012) bahwa proses bisnis merupakan serangkaian aktivitas yang dibentuk yang saling bekerjasama dalam sebuah organisasi dan lingkungan teknis. Kegiatan ini bersama-sama mewujudkan suatu tujuan bisnis. Setiap proses bisnis diberlakukan oleh satu organisasi, tetapi mungkin berinteraksi dengan proses bisnis yang dilakukan oleh organisasi lain. Terdapat tiga jenis proses bisnis, antara lain proses manajemen, proses operasional dan proses pendukung. Proses manajemen, yakni proses yang mengendalikan operasional dari sebuah system, contohnya semisal Manajemen Strategis. Proses operasional, yakni proses yang meliputi bisnis inti dan menciptakan aliran nilai utama, contohnya semisal proses pembelian, manufaktur, pengiklanan dan pemasaran, dan penjualan. Proses pendukung, yang mendukung proses inti, contohnya semisal akunting, rekruitmen, pusat bantuan.

2.5 *Business Process Model Notation* (BPMN)

BPMN (*Business Process Modeling Notation*) yaitu suatu metodelogi yang dikembangkan oleh *Bussines Process Modeling Intiative* sebagai standar baru pada pemodelan proses bisni, dan juga sebagai alat desain pada sistem yang kompleks. Dikembangkan oleh *process Management Initiative (BPMNI).* BPMN memiliki tujuan untuk menyediakan notasi yang dapat dipahami oleh pengguna bisnis,

mulai dari analis bisnis yang membuat rancangan proses, lalu orang yang menjalankan proses atau yang mengimplementasikan teknologi, serta orang bisnis yang mengelola dan mengontrol proses tersebut (Silver, 2012).

BPMN memiliki empat kategori elemen sebagai penunjang dalam pembuatan BPMN (Object Management Group, 2011). Empat kategori elemen tersebut adalah :

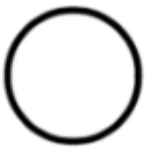
2.5.1 Flow Object

Flow Object dibagi menjadi 3, yaitu *event, activity,* dan *gateway*. Berikut penjelasannya :

1. *Event*

Event merupakan sesuatu yang terjadi dan memiliki dampak dalam proses bisnis. Suatu *event* dapat berasal dari internal dan eksternal suatu proses. *Event* dibagi menjadi tiga yaitu *start event, intermediate event,* dan *end event*. Setiap proses selalu memiliki sebuah *start event* untuk menunjukan awal dari proses bisnis. Tabel 2.2 merupakan penjelasan dari tipe *event*.

Tabel* 2.2 Tipe *Event

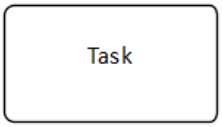
No	Tipe	Deskripsi	Simbol
1	*Start*	Mendeskripsikan dimana suatu proses dimulai	
2	*Intermediate*	Mendeskripsikan dimana suatu terjadi diantara awal dan akhir proses. Akan mempengaruhi alur dari proses, tapi tidak akan memulai atau memberhentikan proses.	
3	*End*	Mendeskripsikan suatu proses berakhir.	

Sumber : Object Management Group (2011)

2. *Activity*

Activity merupakan tugas yang dilakukan adlam sebuah proses bisnis. *Activity* ditunjukan dengan kotak dengan ujung bulat dengan nama yang menjelaskan aktivitas yang dilakukan. Terdapat dua macam *activity* yaitu *task* dan *sub process*. Tabel 2.3 Merupakan penjelasan dari tipe *activity.*

Tabel* 2.3 Tipe *Activity

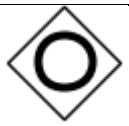
No	Tipe	Deskripsi	Simbol
1	*Task*	Merupakan aktivitas yang dilakukan pada alur proses	
2	*Sub Process*	Merupakan sebuah aktivitas majemuk yang dimasukkan dalam proses. Aktivitas majemuk tersebut dapat dijelaskan dengan lebih detail.	

Sumber : Object Management Group (2011)

3. *Gateway*

Gateway bertanggung jawab mengontrol bagaimana alur dari sebuah proses bisnis. Tabel 2.4 merupakan penjelasan dari tipe *gateway.*

Tabel* 2.4 Tipe *Gateway

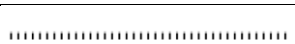
No	Tipe	Deskripsi	Simbol
1	*Exclusive*	Sebagai *divergence* : digunakan untuk membuat jalur alternatif dalam sebuah proses, tapi hanya satu yang dipilih. Sebagai *convergence* : digunakan untuk mengabungkan jalur alternatif.	
2	*Parallel*	Mendeskripsikan proses yang dijalankan secara bersamaan	
3	*Inclusive*	Mendeskripsikan sebuah proses yang dipecah menjadi beberapa jalur.	
4	*Complex*	Mendeksripsikan alur yang kompleks pada sebuah proses bisnis	

Sumber : Object Management Group (2011)

2.5.2 Connections

Connections adalah elemen yang menghubungkan *flow objects*. Tabel 2.5 merupakan penjelasan dari tipe *connections*.

Tabel* 2.5 Tipe *Connections

No	Tipe	Deskripsi	Simbol
1	*Sequence Flow*	Menunjukan urutan aktivitas yang dilakukan pada sebuah proses	
2	*Association*	Menunjukan hubungan antara data, teks, artifak lain, dan *flow object* pada sebuah proses	
3	*Message Flow*	Menunjukan alur pesan antara dua partisipan yang mampu mengirim dan menerima pesan	

Sumber : Object Management Group (2011)

2.5.3 Swimlanes

Swimlanes merupakan wadah grafis yang membagi suatu set aktivitas dengan aktivitas lain. Tabel 2.6 meupakan penjelasan tipe *swimlanes*.

Tabel* 2.6 Tipe *Swimlanes

No	Tipe	Deskripsi	Simbol
1	*Pool*	Merupakan wadah yang berisi satu proses dan *sequence flow* yang menghubngkan aktivitas	
2	*Lane*	Digunakan untuk mempresentasikan tanggungjawab aktivitas pada sebuah proses.	

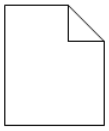
Sumber : Object Management Group (2011)

2.5.4 Artifacts

Artifacts mempresentasikan sebuah objek diluar sebuah proses. *Artifact* dapat mempresentasikan data atau catatan yang menjelaskan sebuah proses atau dapat

digunakan untuk mengelola tugas atau proses. Tabel 2.7 meupakan penjelasan tipe *artifacts*.

Tabel* 2.7 Tipe *Artifacts

No	Tipe	Deskripsi	Simbol
1	*Data Object*	Mempresentasikan informasi yang mengalir pada sebuah proses seperti dokumen bisnis, surat, *email* dan lain-lain.	
2	*Data Store*	Tempat dimana proses dapat membaca atau menulis data	
3	*Annotation*	Menunjukan informasi tambahan kepada pembaca sebuah diagram BPMN.	
4	*Group*	Memungkinkan untuk mengelompokkan elemen secara informal	
2	*Data Store*	Tempat dimana proses dapat membaca atau menulis data	

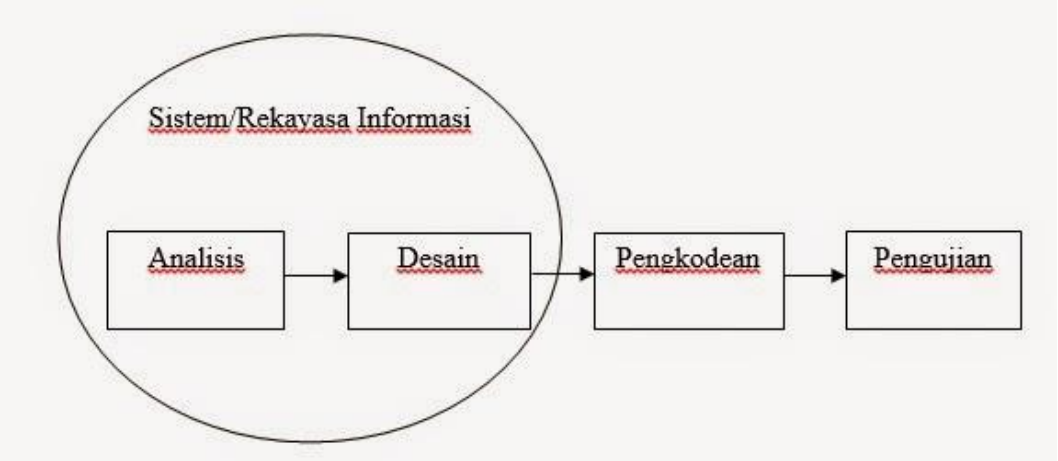
Sumber : Object Management Group (2011)

2.6 *Software Development Life Cycles* (SDLC)

Menurut turban (2000) *System Development Life Cyce* (SDLC) adalah metode pengembangan sistem tradisional yang digunakan sebagian besar organisasi saat ini. SDLC merupakan kerangka kerja (*framework*) yang terstruktur dan berisi proses-proses yang skuensial bagaimana mengembangkan sistem informasi. Didalam *framework* SDLC terdapat beberapa metode, diantaranya adalah metode *waterfall*, pada sub-bab berikut ini akan dijelaskan mengenai metode *waterfall* dan bagaimana tahapan-tahapan dalam metode *waterfall.*

2.6.1 Waterfall Model

Menurut Rosa A.S Model SDLC *Waterfall (*air terjun*)* sering juga disebut sekuensial linier atau alur hidup klasik. Model *waterfall* menyediakan pendekatan alur hidup yang terurut atau sekuensial dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung(*support*) (Rosa A.S, 2013). Model *waterfall* sangat cocok digunakan untuk pengembangan perangkat lunak atau sistem informasi dengan spesifikasi yang tidak beruba-ubah



Gambar 2.2 Ilustrasi model *waterfall*

Sumber : Rosa A.S(2013)

Pada gambar 2.1 merupakan *waterfall model* dimana terdapat beberapa tahapan-tahapan yang harus dilakukan secara berurutan. Berikut ini adalah penjelasan dari tahapan-tahapan tersebut:

1. Analisis Kebutuhan perangkat lunak
   Proses pengumpulan kebutuhan perangkat lunak akan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat diperoleh pemahaman perangkat lunak apa yang akan dibutuhkan.
2. Desain
   Desain perangkat lunak adalah langkah yang dokus pembuatan perogram perangkat lunak antra lain struktur data, arsitektur perangkat lunakm representasi antarmuka, dan prosedur pengodean. Pada tahap ini kebutuhan perangkat perangkat lunak yang diperoleh dari proses analisis kebutuhan kebutuhan perangkat lunak ke representasi desain agara dapat diimplemntasika menjadi program pada tahap selanjutnya yaitu tahapan pengkodean program.
3. Pembutan kode program
   Desain harus ditranslasikan kedalam program perangkat lunak yang dimana hasil dari tahapan ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahapan desain.
4. Pengujian
   Pengujuan digunakan untuk memastikan bahwa fungsional dan lojik sudah teruji agar meminimalisir kesalahan dan memastikan keluaran yang dihasilkan sesuai dengan yang inginkan.

2.7 Kebutuhan Fungsional dan Non-Fungsional

Kebutuhan sistem merupakan deskripsi dari apa saja yang harus sistem lakukan. Kebutuhan tersebut merefleksikan kebutuhan konsumen dari sistem yang menyediakan tujuan khusus. Proses mencari, menganalisis, mendokumentasikan dan memerikas ketersediaan layanan yang disebut rekayasa

kebutuhan (*requiretment engineering*). Menurut Sommerville (2011 hal.83) mengenai kebutuhan fungsional dan non – fungsionnal adalah

1. Kebutuhan Fungsional

   Merupakan pernyataan dari layanan sistem harus disediakan, bagaimana sistem harus bereaksi terhadap input tertentu, dan bagaimana sistem harus berperilaku dalam situasi tertentu. Dalam beberapa kasus, kebutuhan fungsional dapat juga secara eksplisit menyatakan apa yang sistem tidak harus melakukan.

2. Kebutuhan Non – fungsional

   Kebutuhan non-fungsional merupakan kendala pada layanan atau fungsi yang ditawarkan oleh sistem. Termasuk didalamnya kendala waktu, kendala proses pembangunan, dan kendala yang pada standarisasi. kebutuhan non-fungsional sering berlaku untuk sistem secara keseluruhan, bukan fitur sistem individual atau jasa.

2.8 *Object Oriented Analysis and Design* (OOAD)

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahapan untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi object (Rosa A.S., 2016). Sedangkan Desain berorientasi objek atau *Object Oriented Design* (OOD) adalah tahapan perantara untuk memetakan spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah dimplemetasikan dengan pemerograman berorientasi objek(Rosa A.S.,2016). OOA dan OOD memiliki batasan yang samar, sehingga banyak yang menyebut dengan *Object Oriented Analysis and Design* (OOAD). Pemodelan berorientasi object dapat dituangkan dengan banyak model dokumentasi perangkat lunak namun yang paling digunakan adalah UML (*Unified Modeling Language*).

2.8.1 UML (*Unified Modeling Languege)*

UML merupakan salah satu standar bahasa yang digunakan dalam mendefenisikan *requiretment* membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. Tujuan dari UML adalah untuk menggambarkan sebuah mekanisme pemodelan yang mudah dipahami dalam pembangunan suatu sistem. UML dikelompokan menjadi 3 kategori pembagian yaitu (1) *Behavior Diagram* yang meliputi *Use case diagram*, Activity diagram, *State machin diagram*; (2) *Structure Diagram* yaitu meliputi *Sequence diagram, Communication diagram, Timing Diagram dan Interaction diagram*; (3) *Structure diagram* yaitu meliputi *Class diagram, Object diagram, Component diagram, Composite diagram, Package diagram, dan Deployement diagram (*Rosa A.S*)*. Namun dalam pembahasan ini hanya akan menjelaskan diagram diagram yang digunakan dalam mendukung penelitian pengembangan sistem informasi rekam medis rumah sakit.

2.8.1.1 Use case diagram

Use case digunakan untuk menerjemahkan fungsi dan fitur perangkat lunak dari prespektif pengguna. Menggambarkan bagaimana pengguna berinteraksi dengan sistem dengan mendefinisikan langka-langkah diperlukan untuk mencapai tujuan tertentu. Berikut ini pada tabel 2.8 merupakan penjelasan dari simbol *use case diagram.*

Tabel* 2.8 Penjelasan simbol *use case diagram

No	Gambar	Nama	Keterangan
1		*Actor*	Orang, proses, atau sistem lain yang memiliki interaksi dengan sistem informasi yang akan dibangun.
2		*Use case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antara unit / aktor. Biasanya mengunakan kata kerja di awal frase nama *use case*
3		*Association*	Merupakan komunikasi antara *use case* dan aktor yang berpartisipasi dengan *use case*.
4	<<extend>>	*Extend*	Relasi *use case* tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan
5		*Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus). Hubungan ini berada antara 2 buah *use case*, fungsi salah satu *use case* lebih umum dari fungsi lainnya.
6	<<include>>	*Include*	Relasi *use case* tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan memerlukan *use case* tersebut untuk menjalankan fungsi / sebagai prasyarat jalannya *use case* tersebut.
7	<<invokes>>	*Invokes*	Merupakan sosiasi yang memanggil *use case* yang dituju dan merupakan superset dari *include* dan *extend*
8	<<precedes>>	*Precedes*	Merupakan asosiasi dimana *use case* yang dituju harus selesai terlebih dahulu sebelum *use case* yg merujuk

2.8.1.2 Use case Scenario

Model yang menggambarkan persyaratan fungsional suatu sistem atau klarifikasi lainnya dalam use case. Model use case menyajikan sebuah sistem dalam hal penggunaannya. Bila terjemahkan secara formal, model use case menjelaskan semua kemungkinan cara menggunakan sistem (Bittner dan Spence, 2003). Template yang digunkan oleh penulis berdasarkan oleh yang di buat oleh Leudke(2009) pada Gambar 2.3.

Use Case Template	'Create Order' Use Case
Use Case Name	*Create order*
Use Case Description	*Create order is the ability to request the purchase of a product*
Actor	*Order Creator*
Pre-conditions	• *Order Creator has been identified*
Basic Flow	1. *Order Creator selects 'order product' action* 2. *System requests customer/product identification information* 3. *Order Creator provides customer/product identification information* 4. *System requests mailing information* 5. *Order Creator provides mailing information* 6. *System verifies mailing information* 7. *System requests order be submitted* 8. *Order Creator submits order* 9. *System submits product order for processing* 10. *System confirms product order*
Post-conditions	• *Product order has been created*
Alternate Flows	• *Product is not in stock* • *Product has been discontinued* • *A customer's initial order is over $200*

USER REQUIREMENT

FUNCTIONAL REQUIREMENTS

Gambar 2.3 Template Use Case Scenario

Sumber : Betty Leudke (2009)

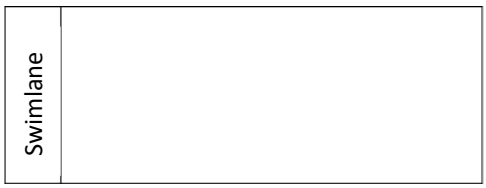
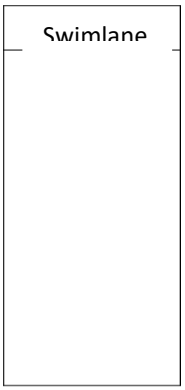
2.8.1.3 Activity diagram

Activity diagram merupakan sebuah diagram visual yang menggambarkan alur sebuah aktivitas seperti sistem, bisnis, alur kerja, atau proses yang lain. Activity diagram berfokus pada aktivitas yang dilakukan dan siapa yang bertanggung jawab pada proses tersebut(Booch, et, al., 2007). Activity diagram juga banyak digunakan untuk mendefinisikan hal-hal berikut (Rosa A.S):

- Rancangan proses bisnis yang dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
- Urutan atau pengelompokan tampilan dari sistem dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
- Rancangan pengujian diama setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
- Rancangan menu yang ditampilkan pada perangkat lunak

Berikut merupakan simbol-simbol yang digunakan dalam *activity diagram* yang dijelaskan pada tabel 2.9.

Tabel* 2.9 Penjelasan simbol *activity diagram

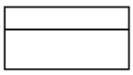
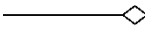
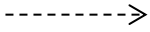
Simbol	**Deskripsi**
	Status awal aktivitas sistem, sebuah activity diagram memiliki sebuah status awal
	Aktivitas, Aktivitas yang dilakukan sistem atau aktor yang bertanggung jawab pada aktivitas tersebut, aktivitas biasanya diawali dengan kata kerja
	Percabangan, asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
	Penggabungan/join, asosiasi penggabungan diaman lebih dari satu aktivitas digabungkan menjadi satu
	Status akhir yang dilakukan sistem, sebuah *activity diagram* memiliki sebuah status akhir
 Atau 	Swimlane, memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber : Rosa A dan Shalahuddin (2016)

2.8.1.4 Class diagram

Class diagram digunakan untuk menunjukan keberadaan dari sebuah *class* dan hubungan antar *class*. Suatu class diagram merepresentasikan struktur dari sebuah *class* yang ada didalam sistem(Booch, et al.,2007). Elemen yang paling penting dalam sebuah *class diagram* adalah hubungan antar *class*. Tabel 2.10 menjelaskan simbol dari *class diagram*.

Tabel* 2.10 Penjelasan simbol *class diagram

No	Gambar	Nama	Keterangan
1		*Association*	Relasi antar kelas dengan makna umum
2		*Class*	Kelas pada struktur sistem
3		*Generalization*	Hubungan antar kelas dengan makna generalisasi dan spesialisasi (umum-khusus)
4		*Aggregation*	Relasi antar kelas dengan makana semua-bagian (whole-part)
5		*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas

Sumber : Rosa A dan shakahuddin (2016)

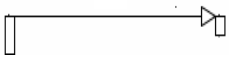
2.8.1.5 Sequence diagram

Sequence diagram menggambarkan urutan interaksi aktor dan objek didalam sebuah sistem dan interaksi antar objek. Kelebihan *sequence diagram* adalah menunjukan urutan fungsi dari sebuah objek ke objek lain. *Sequence diagram* membantu dalam representasi detail dari sebuag *use case* dan digunakan untuk menunjukan interaksi pada suatu skenario dalam perangkat lunak(Sommerville, 2011). Berikut pada tabel 2.11 merupakan penjelasan mengenai simbol-simbol yang digunakan dalam *sequence diagram.*

Tabel* 2.11 Penjelasan simbol *sequence diagram

No	Gambar	Nama	Keterangan
1		*Actor*	Orang, proses, atau sistem lain yang memiliki interaksi dengan sistem informasi yang akan dibangun.
2		*LifeLine*	Merupakan kehidupan suatu objek

Tabel* 2.11 Penjelasan simbol *sequence diagram(*lanjutan*)

No	Gambar	Nama	Keterangan
3		Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah tahapan yang dilakukan didalamnya.
4		*Message*	Spesifikasi dari komunikasi antar objek yang berisi informasi tentang aktifitas yang terjadi. Arah panah mengarah pada objek yang memiliki operasi / metode.

Sumber : Rosa A dan Shalahuddin (2016)

2.9 Pemrograman berorientasi obyek

Pemrograman dengan menggunakan metodelogi berorientasi obyek adalah siatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan obyek yang berisi data dan operasi yang diberlakukan terhadapnya(Rosa A.S., 2016). Pemerograman berorientasi obyek merupakan suatu cara bagaimana sistem informasi atau perangkat lunak dibangun melalui pengdekatan objek dengan cara sistematis. Keuntungan menggunakan metodelogi berorientasi objek menurut Rosa A.S (2016 hal 101) adalah :

- Meningkatkan produktivitas dengan menggunakan ulang objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut
- Kecepatan pengembangan karena sistem yang dibangun dengan baik dan benar pada saat anaisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat penkodean
- Kemudahan pemeliharaan karena dengan model objekm pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah
- Adanya konsistensi karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisism perancangan maupun pengkodean
- Meningkatkan kualitas perangkat lunak karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya.

Didalam menggunakan pemrograman berorientasi obyek terdapat sebuah teknologi *bast practice* yang digunakan untuk menyelesaikan permasalahan permrograman yang bisa disebut dengan *Design Pattern*(Pola Desain). Menurut Rosa A.S. (2016, hal 242) *design pattern* dibagi menjadi beberapa kelompok yaitu sebagai berikut:

1) Pola yang berbasis pada Proses Pembuatan

Pola disusun dengan menitikberatkan komponen-komponen atau objek-objek apa saja yang perlu untuk dibuat.

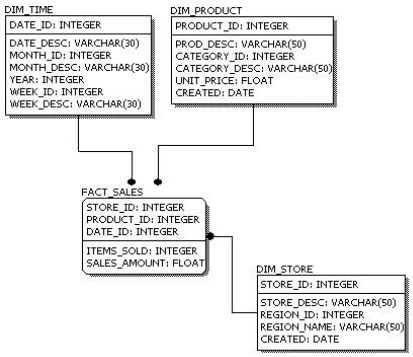
2) Pola yang berbasis pada Struktur (*Structural Pattern*)
   Pola yang disusn dengan menitik beratakan pada struktur spesifikasi komponen atau kesesuaian dengan objek dengan tindakan atau kemempuan yang diharapkan pada pemerograman berorientasi objek.
3) Pola yang berbasis pada kebiasaan(*Bahavior patterns*)
   Pola yang disusun dengan menitik beratkan pada spesifikasi kebiasaan sistem aplikasi yang akan dibuat apakah sesuai dengan kemampuan aplikasi yang diharapkan
4) Pola yang berbasis pada Desain Arsitektur (*Arcitectural Design Patterns*)
   Pola yang menitik beratkan pada hirarki architektur atau pemisahan blok-blok kerja. *Model View Controller* adalah salah satu contoh *arcitectural design patterns. Codeigneter* merupakan salah satu *framework* aplikasi hasil implementasi dari *design pattern* yang digunakan untuk memudahkan *programmer* dalam membuat sebuah aplikasi serta perubahan(*costumize*) terhadap aplikasinya.

2.10 Physical Data Model

Physical data model merepresentasikan bagaimana model akan dibangun di *database*. Sebuah *physical data model* menampilkan struktur tabel yang terdiri dari nama kolum, tipe data kolum, *contraints* kolum, *primary key, foreign key,* dan relasi antara tabel (1keydata, 2017). Terdapat beberapa tahap dalam pembuatan *physical data model,* yakni

1. Merubah entitas menjadi tabel.
2. Merubah relasi menjadi *foreign key.*
3. Merubah atribut menjadi kolum.
4. Memodifikasi *physical data model* sesuai dengan kebutuhan.

Pada Gambar 2.6 dapat dilihat contoh *physical data model.*



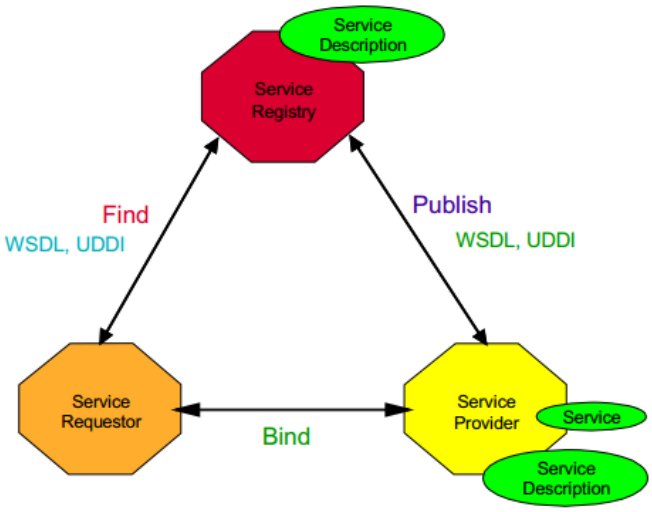
Gambar 2.4 *Physical data model*

Sumber : 1keydata(2017)

2.11 Web Service

Web Service merupakan suatu komponen software yang merupakan *self-containing*, aplikasi modular yang dapat dipublikasikan, dialokasikan, dan dilaksanakan pada web. *Web service* adalah teknologi yang mengubah kemampunan internet dengan menambahkan kemampuan *transactional* web, yaitu kemampuan untuk saling berkomunikasi oleh *program-to-program*. Berikut ini merupakan karakteristik *web service* (Yasin, 2012):

1. Merupakan *application logic* yang dapat diakses dan dipublikasikan menggunakan standar Internet (TCP/ IP, HTTP, web).
2. Dideskripsikan dalam format XML.
3. Diidentifikasikan dengan *Universal Resources Identifier* (URI).
4. Bersifat *Loosely coupled, self-contained*, modular dan terbuka (*nonpwprietary*)
5. Digunakan untuk mendukung interoperabilitas interaksi *machine-to-machine* melalui jaringan internet/intranet.



Gambar 2.5 Permodelan *web service*

Sumber : *Web Service Conceptual Architecture* (2011)

Pada gambar 2.2 merupakan permodelan *web service* dimana memiliki 3 peranan dalam berinteraksi yaitu:

1. *Service provider* (penyedia layanan)
   berfungsi untuk menyediakan layanan atau service dan mengolah sebuah *registry* agar layanan-layanan tersebut dapat tersedia.
2. *Service registry* (daftar layanan)
   berfungsi sebagai lokasi utama yang mendeskripsikan semua layanan atau *service* yang telah didatarkan.
3. *Service requestor* (pemohon layanan)
   berfungsi sebagai pemohon layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut.

Selain itu, web service memiliki 3 operasi yang terlibat didalamnya yaitu:

1. *Publish,* berfungsi untuk menerbitkan layanan ke dalam *registry.*
2. *Find,* berfungsi untuk *service requestor* yang mencari yang menemukan layanan yang dibutuhkan.
3. *Bind, b*erfungsi untuk *service requestor* setelah menemukan layanan yang dicari, kemudian melakukan *binding* ke *service provider* untuk melakukan interaksi dan mengakses layanan yang disediakan oleh *service provider*.

Dalam pengembangan sebuah *web service* terdapat standar pertukaran pesan berbasis XML yang berjalan melalui jaringan internet yang dinamakan dengan SOAP (*Simple Object Access Protokol*). Dokumen pesan yang dikirim dalam sebuah pesan SOAP adalah sebuah dokumen XML yang memiliki elemen-elemen berikut:

- *Evelope*, yaitu elemen yang mengidentifikasi dokumen XML sebagai sebuah pesan SOAP.
- *Header*, yaitu elemen yang berisikan informasi *header* dari sebuah pesan SOAP.
- *Body,* yaitu elemen yang berisikan panggilan dan merespon informasi.
- *Fault,* yaitu elemen yang berisikan pesan kesalahan yang terjadi pada waktu proses pemanggilan atau pengiriman pesan SOAP.

NuSOAP adalah salah satu *toolkit open source* yang tersedia untuk bahasa pemerograman PHP dalam membuat pesan SOAP. Salah satu keunggulan NuSOAP adalah tidak perlu melakukan registrasi khusus ke sistem operasi atau server.

2.12 Pengujian

Kebutuhan perangkat lunak dapat dibagi menjadi dua, yaitu : kebutuhan fungsional dan non fungsional. Dimana kebutuhan fungsional berkaitan dengan fungsi sistem dan kebutuhan non fungsional mencakup akan reliabilitas, kapasitas dan performansi sistem.

Pengujian fungsional dapat dilakukan dengan menggunakan metode whitebox dan blackbox untuk keperluan fungsional sistem apakah sudah dapat berjalan dengan baik atau tidak. Sedangkan untuk pengujian non fungsional ada beberapa yang tidak dapat di uji dengan mudah seperti kapastitas yang tidak cocok, dan rentang waktu yang cukup panjang untuk menjamin bahwa perangkat lunak tidak akan bermasalah. (Gede, 2012)

2.12.1 Pengujian *white box*

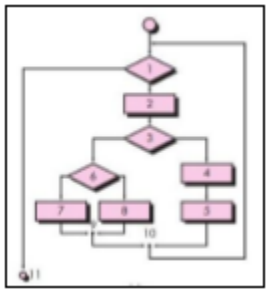
Metode *white box* testing merupakan merupakan teknik pengujian dimana pengujian dilakukan pada kode atau algoritme program. Tujuan dilakukan *white box testing* adalah untuk memastikan tidak ada kesalahan dalam kode atau algoritme program. Kelebihan dari *white box testing* adalah untuk mengetahui kesalahan dalam kode serta memberikan pertimbangan kepada pengembang dalam implementasi perangkat lunak (Agarwal, et al., 2010)

Salah satu metode dari *white box testing* adalah *basis path testing*. *Basis path testing* merupakan pengujian yang memiliki tujuan untuk mengetahui kompleksitas logika (*cyclomatic complexity*) dalam sebuah program (Pressman, 2010). *Cyclomatic complexity* didapatkan berdasarkan *flow graph* suatu kode atau

algoritme. *Cycloamtic complexity* digunakan sebagai dasar dalam membuat jalur eksekusi program. Jalur eksekusi program digunakan sebagai dasar dalam membangun kasus uji yang digunakan untuk menjalakan setiap pernyataan di dalam program dalam satu kali pengujian. *Basis path testing* memiliki empat elemen pengujian yaitu (Pressman, 2010) :

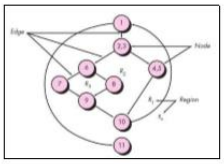
1. *Flow graph*

Flow Graph mendeskripsikan struktur logika sebuah program. Gambar 2.5 merupakan representasi alur program menggunakan *flow chart*. Gambar 2.6 merupakan representasi alur program menggunakan *flow graph*.



Gambar 2.6 *Flow chart Program*

Sumber Pressman (2012)



Gambar 2.7 *Flow grap program*

Sumber Pressman (2012)

Berdasarkan Gambar 2.5 dapat diketahui bahwa bentuk lingkaran merupakan sebuah node. Setiap node dapat mewakilli satu atau lebih prosedural. Setiap node yang mewakili suatu kondisi disebut predicate node. Tanda panah yang menghubungkan node dengan node disebut *edge*. *Edge* menggambarkan arah suatu algoritme program pada *flow chart*. Wilayah yang dibatasi oleh *edge* dan node disebut region. Ketika menghitung region, area diluar *flow graph* dihitung sebagai satu region

2. *Cyclomatic Complexity*

Cyclomatic Complexity merupakan suatu ukuran yang dapat mengetahui kompleksitas logika suatu program. Terdapat tiga cara dalam menghitung *cyclomatic complexity* V(G) :

a. V(G) = jumlah *region* dalam *flow graph*

b. V(G) = E (*Edge*) - N (*Node*) + 2

c. V(G) = P (*Predicate Node*) + 1

3. *Independent Path*

Independent Path merupakan semua jalur yang dilalui program untuk menghasilkan sutu hasil atau kondisi baru. Sebuah *independent path* setidaknya melewati satu *edge* dimana belum dilewati oleh *independen path* sebelumnya.

4. *Test Case*

Test Case merupakan merupakan data *input* yang digunakan untuk memeriksa alur logika atau kondisi pada masing-masing *independent path*. Berdasarkan (Software Engineering Institute, 1997) nilai *cyclomatic complexity* memiliki arti pada kode, algoritme, atau program. Tabel 2.12 merupakan penjelasan arti nilai *cyclomatic complexity.*

Tabel* 2.12 Arti Nilai *Cysclomatic Complexity

Cysclomatic Complexity	**Penjelasan**
1-10	Program mudah dipahami, mudah dilakukan implemntasi, mudah dilakukan perbaikan, mudah dilakukan pengujian dan resiko kesalahan program rendah
11-20	Program lebih kompleks, pengujian lebih sulit, dan resiko kesalahn sedang
21-50	Program sangat kompleks, pengujian memerlukan usaha karena terdapat banyak jalur eksekusi program, dan resiko kesalahan program tinggi
Lebih dari 50	Program sulit dilakukan implementasi, perbaikan, pengujian, dan rsiko kesalahan program sangat tinggi

2.12.2 Pengujian *Black Box*

Black box testing merupakan teknik pengujian yang hanya melibatkan observasi output dari input nilai tertentu dan tidak ada percobaan untuk analisis kode program. Black box testing berfokus pada fungsi sebuah perangkat lunak. Black box testing memungkinkan software engineer untuk membuat set kondisi input yang dapat dikerjakan oleh semua fungsi dalam perangkat lunak (Agarwal, et al., 2010). Metode pengujian *black box* diperlukan untuk memfokuskan pada keperluan fungsional dari sistem. Kriteria metode black box testing berupa input dan scenario yang dijalankan oleh *user* menghasilkan output yang diharapkan. Dalam hal ini perlu mekanisme uji yang berisi task atau langkah kerja yang dilakukan pengguna beserta datanya. Jika output yang dihasilkan sesuai atau benar maka uji fungsional berhasil, apabila output yang dihasilkan tidak sesuai maka uji fungsional gagal (Gede, 2012).

2.12.2.1 Validation Testing

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Item-item yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black Box*, karena tidak memerlukan untuk berkonsentrasi terhadap alur jalannya algoritme program dan lebih ditekankan untuk menemukan kinerja sistem dengan daftar kebutuhan (Indriati, 2010).