# BAB II

**LANDASAN TEORI**

## 2.1 Penelitian Terdahulu

Penelitian sebelumnya telah menjadi salah satu rujukan bagi penulis untuk melakukan penelitian, sehingga penulis dapat memperkaya teori-teori yang digunakan untuk pengujian penelitian. Dari penelitian sebelumnya, penulis tidak menemukan penelitian dengan nama yang sama dengan penelitian penulis. Akan tetapi penulis telah mengajukan beberapa penelitian sebagai referensi untuk memperkaya bahan penelitian dalam penelitian penulis. Berikut ini adalah penelitian sebelumnya dalam beberapa format jurnal terkait dengan penelitian yang dilakukan oleh penulis.

Penelitian pertama karya dari (Novari, 2013) menghasilkan sebuah sistem informasi pengajuan barang, Sistem tersebut dapat digunakan untuk mengatur pengajuan komoditas dan memantau sejauh mana proses pengajuan komoditas yang dilakukan oleh unit di AMIK AKMI Baturaja. Sistem terdiri dari beberapa menu, antara lain menu utama, menu transaksi, menu laporan dan menu keluar. Pada menu data utama terdapat submenu antara lain data unit, data proyek, data pegawai dan data supervisor. Ada beberapa submenu pada menu pemrosesan, antara lain submenu submission, submenu approval dan submenu delivery. Menu laporan berisi beberapa laporan yang diperlukan dalam sistem pengiriman komoditas.

Penelitian kedua karya dari (Azdy and Rini, 2018) menghasilkan sebuah aplikasi berbasis web yang dapat digunakan oleh semua pengguna jasa universitas untuk melakukan pengaduan. Pengembangan aplikasi menerapkan metode rekayasa perangkat lunak yang digunakan adalah *eXtreme Programming.* Hasil penelitian menunjukkan bahwa penerapan *eXtreme Programming* dalam pengembangan aplikasi mengarah pada perkembangan yang pesat dengan jumlah anggota tim yang paling sedikit. Pengembangan dapat diselesaikan dengan cepat, karena pengembangan aplikasi terlebih dahulu membangun komponen yang paling sederhana, komponen verifikasi akun dan kemudian komponen lainnya. Hasil pengujian menggunakan metode *black box* dapat berjalan normal dan memenuhi *acceptance criteria* pada *user story.*

Penelitian ketiga karya dari (Anwar and Irawan, 2017) menjelaskan bahwa penelitian ini membahas bahwa komputer dan internet merupakan peralatan yang diciptakan untuk mempermudah pekerjaan manusia. PT. Andalan Chrisdeco membutuhkan sebuah sistem informasi untuk mendukung pekerjaan dalam proses pengajuan bagian, sehingga pekerjaan menjadi lebih efisien. Sistem yang ada PT. Mid-stream in-pillars Chrisdeco hampir seluruhnya terkomputerisasi, namun proses pengajuan pengadaan suku cadang masih dilakukan secara manual dan belum bisa diakses melalui Internet. Oleh karena itu, diperlukan sistem informasi untuk memproses permintaan pembelian suku cadang.

Penelitian terakhir karya dari (Susanto, Rusdianto and Kurniawan, 2018) menghasilkan sistem pengelolaan pengajuan kegiatan mahasiswa memiliki 11 peserta dan 23 persyaratan fungsional, yang dapat membantu pengelolaan proses pengusulan kegiatan mahasiswa. Berdasarkan hasil perancangan, terdapat total 44 kategori yang terbagi dalam komponen model, view, dan controller yang menggambarkan setiap operasi tiap kategori dalam bentuk algoritma, 14 tabel pada hasil perancangan database, dan interface yang digunakan sebagai referensi pada tahap implementasi. Hasil dari implementasi sistem tersebut telah menghasilkan fungsi-fungsi yang dapat membantu mengelola saran kegiatan siswa.

## 2.2 Sistem Informasi

Menurut O'Brien (2006), sistem informasi adalah kombinasi terorganisir dari personel, perangkat keras, perangkat lunak, jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah, dan mendistribusikan informasi dalam suatu organisasi. Informasi adalah proses mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk tujuan tertentu. Pada saat yang sama, menurut Loudon (2005), sistem informasi secara teknis dapat didefinisikan sebagai komponen yang saling berhubungan yang mengumpulkan (atau mengekstrak), memproses, menyimpan dan mendistribusikan informasi untuk membantu pengambilan keputusan dan pengendalian dalam organisasi. Kelompok penulis ini percaya bahwa sistem informasi dapat didefinisikan sebagai seperangkat sumber daya yang saling terkait yang dirancang untuk mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi dalam suatu organisasi. (Mahendra, Tresno and Yanto, 2018)

Menurut O’Brien (2006), ada tiga peran utama teknologi dan sistem informasi dalam perusahaan bisnis saat ini, yaitu :

1. Memberikan dukungan untuk proses dan operasi bisnis.
2. Mendukung pengambilan keputusan.
3. Mendukung berbagai strategi keunggulan kompetitif.

## 2.3 Agile Development Methods

Konsep *agile development methods* diciptakan oleh Kent Beck dan 16 rekannya, yang menemukan bahwa pengembangan perangkat lunak tangkas adalah metode membuat perangkat lunak dengan menjalankan perangkat lunak dan membantu orang lain membuatnya. Teknologi *agile development methods* atau metodologi tangkas adalah seperangkat metode pengembangan perangkat lunak berdasarkan pengembangan berulang, di mana persyaratan dan solusi dikembangkan melalui kolaborasi antara tim yang terorganisir (Pressman, 2010). Meskipun Sommerville (2011) percaya bahwa metode pengembangan tambahan, dengan fokus pada pengembangan cepat, rilis perangkat lunak tambahan, mengurangi overhead proses dan kode kualitas dan proses pengembangan, secara langsung menarik pelanggan. (Mahendra, Tresno and Yanto, 2018)

Ada beberapa model pengembangan perangkat lunak yang termasuk agile software development methods, yaitu:

### 2.3.1 Extreme Programming

Metode XP dikembangkan oleh Kent Beck, seorang ahli pengembangan perangkat lunak yang berkolaborasi dengan Chrysler yang membuat proyek C3 (*Chrysler Comperhensive Compensation*). Ron Jeffries sengaja menggunakan metode yang berbeda untuk menyelesaikan proyek ini, sehingga metode ini disebut metode XP. Kent Beck mengubah desain proyek agar lebih efisien, mudah beradaptasi, dan fleksibel. (Rahmi, Sari and Suhatman, 2016). XP memiliki 4 nilai dasar yang menjadi inti pokok metode XP , yaitu :

1. *Communication* (Komunikasi)

XP berfokus pada komunikasi yang baik antara anggota tim. Anggota tim perlu membangun hubungan dan berbagi pengetahuan dan keterampilan mereka dalam pengembangan perangkat lunak. Pemrogram sebagian besar adalah diri yang cukup besar yang harus ditekan dan terbuka untuk bekerja sama dengan pemrogram lain. Saat menulis kode program.

1. *Simplicity* (Kesederhanaan)

Menyederhanakan hal-hal adalah salah satu nilai inti XP. Gunakan metode singkat, buat desain tetap sederhana, hapus fitur yang tidak perlu dan proses penyederhanaan lainnya, ini akan selalu menjadi nilai inti dari semua aspek XP.

1. *Feedback* (Umpan Balik)

Selalu memberikan umpan balik kepada anggota tim dan pihak lain yang terlibat dalam pengembangan perangkat lunak. Selalu ungkapkan pemikiran dan diskusikan kesalahan selama proses pengembangan. Selalu dengarkan pendapat rekan-rekan yang lain, karena dari *feedback* inilah sering mengerti bagian mana yang salah, atau bisa diperbaiki lagi melalui *software* yang dikembangkan.

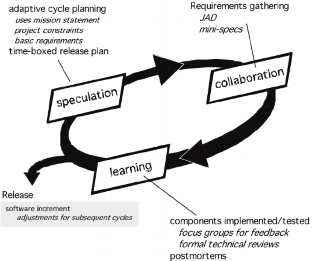
1. *Courage* (Keberanian)

Anggota tim dan penanggung jawab untuk pengembangan perangkat lunak harus selalu percaya pada tanggung jawab mereka dan melakukannya dengan itikad baik. Bahkan di bawah tekanan dari lingkungan (seperti dari pelanggan atau pemilik bisnis), integritas ini harus selalu dijaga. Perasaan saling percaya. XP mencoba membangun dan mengkomunikasikan rasa saling percaya ini dengan berbagai cara.

### 2.3.2 Adaptive Software Development

*Adaptive Software Development (ASD)* merupakan salah satu model dari beberapa model, yang merupakan pengembangan dari *Agile Methods*. Oleh karena itu, ada sebagian orang menyebutnya sebagai *Agile Software Development* (Sagala, 2014).

Jim Highsmith menyarankan ASD sebagai metode untuk membangun sistem dan perangkat lunak yang kompleks. Konsep ASD adalah kolaborasi otonom antara orang dan tim. Perencanaan siklus responsif menggunakan informasi latar belakang seperti pesanan pelanggan, batasan proyek, dan persyaratan dasar untuk menentukan paket perangkat lunak. *Incremental* (produk perangkat lunak dikirimkan secara teratur). ASD menekankan pengorganisasian diri tim, kerja sama antara orang-orang dan pembelajaran sepanjang hayat, baik itu individu atau kelompok. ASD menggunakan alat yang disebut *timeframes*, yang terdiri dari operasi yang menentukan jumlah waktu tertentu untuk menyelesaikan berbagai tugas. Ketika waktu yang ditentukan telah berlalu, sistem akan terus menjalankan tugas berikutnya, berharap berhasil menyelesaikan tugas-tugas utama tertentu sebelum akhir total waktu tugas. Terdapat tiga tahapan pada model ASD seperti gambar 2.1, yaitu: *Speculation, Collaboration, dan Learning*.



**Gambar 2.1** *Adaptive Software Development* (Sagala, 2014)

Sistem kerja yang terdapat dalam metode *adaptive software development* adalah *Collaboration* dan *Learning*, yang berarti :

1) *Collaboration* adalah orang-orang proaktif bekerja sama, saling melengkapi, membantu, pekerja keras, terlatih di bidangnya, dan melaporkan masalah untuk menemukan solusi yang efektif.

2) *Learning* : Tim pengembang biasanya berpikir bahwa mereka tahu segalanya tentang proyek, tetapi ini tidak selalu terjadi. Oleh karena itu, proses ini memungkinkan mereka untuk mempelajari lebih lanjut tentang proyek dalam tiga cara:

a) Focus group: Pelanggan dan pengguna memberikan informasi tentang perangkat lunak.

b) Tinjauan teknis formal: Tinjauan dilakukan oleh seluruh tim ASD.

c) Otopsi: Tim ASD menganalisis kinerja dan proses.

Tahapan pengembangan perangkat lunak adaptif, yaitu:

1) Inisialisasi proyek: Menentukan maksud dan tujuan proyek.

2) Menentukan jadwal proyek (perkiraan durasi proyek).

3) Tentukan jumlah siklus atau interval waktu yang optimal.

4) Tulislah pernyataan tujuan untuk setiap siklus.

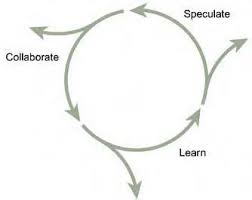
5) Menentukan komponen utama setiap siklus.

6) Buat daftar tugas proyek.

7) Tinjau siklus keberhasilan.

8) Rencanakan siklus berikutnya.

Praktek dari ASD digerakkan olehkepercayaan pada adaptasi yang terus-menerus disesuaikan untuk menerima perubahan yang terus-menerus sebagai norma. Dalam ASD, siklus hidup *plan-design-build* yang statis diganti dengan siklus hidup *Speculate-Collaborate-Learn* yang dinamis yang tercantum pada gambar 2.2. Siklus hidup ini didedikasikan untuk pembelajaran yang terus-menerus, re-evaluasi, mengamati ketidakpastian masa depan, and kolaborasi yang kuat diantara developer, manajemen, dan customer.



**Gambar 2.2** Siklus hidup *Speculate-Collaborate- Learn* (Sagala, 2014).

Keterangan Gambar 2.2:

1. *Speculate* : Inisiasi dan Perencanaan (Initiation and Planning).
2. *Collaborate* : Pengembangan Fitur Bersama.
3. *Learn* : Review Kualitas (Quality Review).

Ada empat kategori umum dalam pembelajaran pada akhir setiap pengembangan iterasi, yakni : Kualitas hasil dari sudut pandang *customer*, Kualitas hasil dari sudut pandang teknis, Menggunakan fungsi dari anggota *delivery team* dan *practices team*, dan status dari proyek (Sagala, 2014).

### 2.3.3 Dynamic Systems Development Method

DSDM merupakan salah satu metode Agile untuk pengembangan perangkat lunak. DSDM merupakan suatu kerangka kerja yang awalnya didasarkan pada Rapid Application Development (RAD) dan mengutamakan keterlibatan pengguna secara berkesinambungan dengan pendekatan pengembangan secara berulang dan bertambah, tanggap terhadap perubahan, untuk membangun sistem perangkat lunak yang memenuhi kebutuhan bisnis tepat waktu dan juga tepat anggaran .

DSDM adalah metode pengembangan perangkat lunak yang fleksibel. DSDM adalah kerangka kerja yang awalnya didasarkan pada *Rapid Application Development* (RAD), yang memprioritaskan interaksi pengguna yang berkelanjutan dan mengadopsi metode *iteratif* dan *inkremental* untuk pengembangan berbasis perubahan untuk menciptakan sistem perangkat lunak yang memenuhi kebutuhan bisnis, tepat waktu dan sesuai anggaran.

9 prinsip inti DSDM merupakan keunggulan DSDM, yaitu:

1) Partisipasi pengguna adalah kunci utama penyampaian proyek yang efektif dan efisien. Pengguna dan pengembang bekerja sama untuk membuat keputusan dengan presisi dan akurasi.

2) Tim pengembangan proyek memiliki hak untuk membuat keputusan yang penting bagi kemajuan proyek tanpa menunggu persetujuan tingkat yang lebih tinggi.

3) Perhatikan berapa kali sistem muncul, dengan asumsi bahwa lebih baik mendapatkan sesuatu yang "cukup baik" lebih awal daripada mendapatkan keseluruhan yang "sempurna" untuk memasuki babak atau tahap berikutnya.

4) Keselarasan dengan tujuan bisnis adalah kriteria utama untuk menerima hasil Membangun sistem yang memenuhi semua kebutuhan bisnis yang mungkin tidak sepenting berfokus pada fungsi-fungsi utama.

5) Pentingnya pengembangan berulang dan bertahap berdasarkan masukan pengguna untuk membuat keputusan bisnis yang efektif.

6) Semua perubahan dalam perkembangan bersifat reversibel.

7) Persyaratan dan persyaratan harus ditentukan sebelum proyek dimulai.

8) Pengujian berjalan melalui seluruh siklus hidup proyek. Dalam hal ini, pengujian bukanlah aktivitas pengembangan yang terpisah. Umpan balik dari pengembang dan pengguna sangat penting untuk memastikan bahwa proyek dijalankan dari perspektif bisnis dan teknis.

9) Kerja sama yang efektif antara semua pemangku kepentingan sangat penting.

DSDM meliputi 3 tahapan utama dan 5 sub tahapan, yaitu:

1) Sebelum proyek diluncurkan, menentukan calon proyek, pembiayaan proyek dan penjaminan proyek. Memecahkan masalah ini pada tahap ini dapat menghindari masalah di tahap selanjutnya.

2) Siklus hidup proyek sebagai dasar DSDM meliputi 5 sub-langkah, yaitu:

a. Studi kelayakan.

b. Studi bisnis.

c. Perulangan model fungsional.

d. Perulangan perancangan dan pembuatan.

e. Penerapan.

3) Setelah proyek berakhir, pastikan pengoperasian sistem yang efektif dan efisien. Hal ini dilakukan melalui *service, improvement,* dan *maintenance* sesuai prinsip DSDM. Pemeliharaan dapat dilihat sebagai upaya untuk meningkatkan pengembangan berdasarkan sifat DSDM (Rusdiana, 2018).

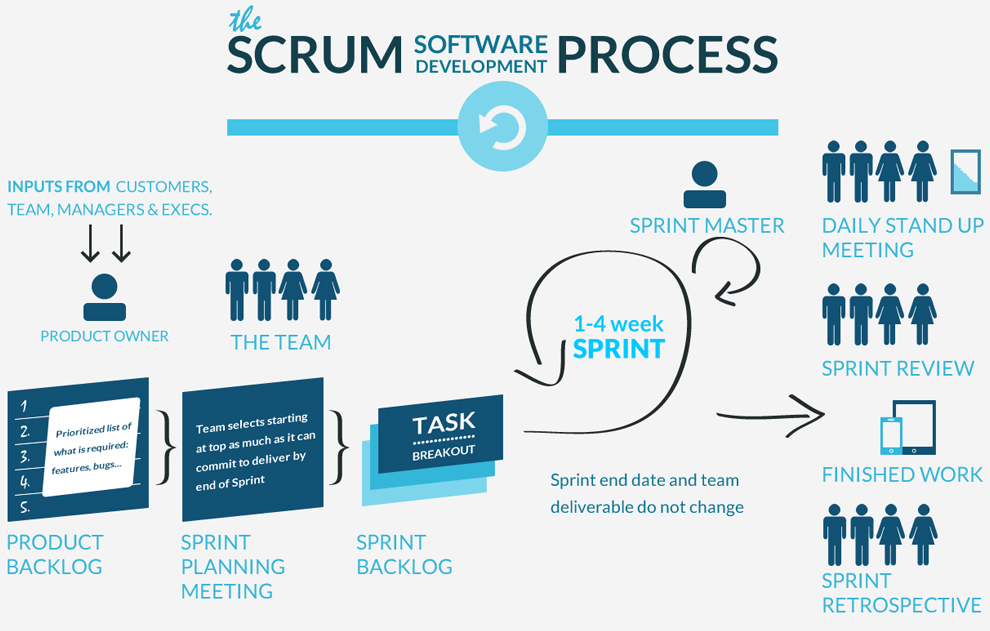
### 2.3.4 Model Scrum

1. Sejarah Scrum

Scrum pertama kali diperkenalkan dalam sebuah artikel berjudul *“New Game for New Product Development”* yang diterbitkan oleh Takeuchi dan Nonaka pada tahun 1986 oleh Harvard Business Review (HBR). Dalam artikel ini, Takeuchi dan Nanaka menyelidiki beberapa perusahaan Jepang, antara lain: Fuji Xerox, Canon, 3M, dan Honda. Survei ini dilakukan untuk memahami bagaimana perusahaan-perusahaan ini berhasil menciptakan produk baru dengan kualitas terbaik. Takeuchi dan Nanaka menemukan bahwa perusahaan-perusahaan ini menggunakan metode yang sama untuk mengembangkan produk mereka. Pertama gunakan metode scrum. Pengembangan perangkat lunak dimulai pada tahun 1993 oleh Jeff Sutherland dari Easel Corporation. Metodologi Scrum pertama kali dikembangkan pada tahun 1995 dengan judul *"Scrum Development Process"* dan diserahkan kepada tim manajemen aset. (Faza, 2018).

1. Apakah Scrum ?

Scrum secara garis besar dapat di ilustrasikan sebagai berikut:



**Gambar 2.3** Scrum *Process* (Faza, 2018)

Scrum building block disebut Sprint. Sprint adalah sebuah kotak-waktu (yang biasanya mempunyai durasi 1 hingga 4 minggu) dimana tim pengembang fokus dalam mencapai target yang jelas. Setiap Sprint selalu berakhir dengan diikuti Sprint Review, dimana hasil yang sudah dibuat dipresentasikan dan didemontrasikan didalam sebuah rapat tim.

1. Istilah – istilah dalam Scrum
   1. *Product Backlog*

Pemilik proyek mengkompilasi semua pertanyaan dan persyaratan sistem, seperti persyaratan sistem *fungsional* atau *non-fungsional* yang diperlukan. Setelah menetapkan tujuan, semua permintaan dan persyaratan dipecah menjadi bagian-bagian kecil, dan setiap bagian kecil harus memiliki nilai. Mereka layak untuk dikembangkan.

Pemilik proyek juga menentukan rentang prioritas untuk menangani detail kecil ini. Bagaimana dan bagaimana detail kecil ini diterapkan dan disampaikan? Pertanyaan-pertanyaan ini membentuk daftar lengkap berdasarkan pasar yang terus berubah dan kebutuhan pelanggan.

* 1. *Backlog Refinement*

Agar sprint dapat berjalan dengan lancar, tim *scrum* harus menjaga backlog perencanaan yang benar dan tepat. *Sprint* (14 minggu) telah selesai. Oleh karena itu, sangat bermanfaat bagi tim untuk menjadwalkan waktu tertentu setiap minggu sehingga dapat melakukan penyempurnaan penjadwalan pada waktu yang sama.

* 1. *Sprint*

*Sprint* adalah kerangka waktu yang mencakup periode kerja di mana *sprint* berfokus pada pengiriman produk berdasarkan item yang dipilih dari *product backlog*. Dalam *sprint*, waktu lari selalu diatur secara berurutan, dan *sprint* baru dimulai segera setelah *sprint* yang ada selesai..

* 1. *Daily Scrum*

Tim *scrum* harus mengadakan rapat hingga 15 menit setiap hari. Tujuannya adalah untuk menyelaraskan kemajuan, mengidentifikasi masalah dan memecahkan masalah yang sedang berlangsung di tempat kerja. Pertanyaan yang Sering Diajukan:

1) Sejak pertemuan terakhir, apa yang Anda lakukan?

2) Apa yang anda rencanakan sebelum pertemuan berikutnya?

3) Apakah ada masalah yang menghalangi anda menyelesaikan pekerjaan yang direncanakan sebelumnya?

Ikuti pertanyaan 1 dan 2 untuk melihat seberapa banyak kemajuan yang telah dicapai dan siapa yang melakukan apa setiap hari. Pertanyaan ketiga digunakan untuk mencari solusi dari permasalahan yang muncul. Dari masalah teknis hingga masalah non-teknis dihitung sebagai kegagalan.

* 1. *Sprint Review*

Setiap *sprint* selalu diakhiri dengan demonstrasi, yang telah dikembangkan untuk memastikan bahwa fungsi-fungsi ini bekerja dengan baik. Tinjauan *sprint* harus dilakukan oleh tim kecil sebelum produk ditampilkan dan disajikan kepada tim besar (pemilik produk, pelanggan, dan manajer).

* 1. *Sprint Retrspective*

*Sprint Retrspective*, tim scrum merefleksikan bagaimana tugas dieksekusi di sprint sebelumnya ?

Diharapkan akan ada tindakan korektif dalam tinjauan *sprint* untuk membuat *sprint* berikutnya lebih baik, dan perbaikan ini harus diterapkan di *sprint* berikutnya.

### 2.3.5 Agile Modelling

*Agile modeling* (AM) adalah sebuah pendekatan untuk menggambarkan kebutuhan dari sebuah sistem perangkat lunak yang akan dibangun secara komprehensif. AM *is chaordic, in that it blends the “chaos” of simple modeling practices and blends it with the order inherent in software modeling artifacts* (Auliasari, 2010)*.*

Banyak teknik telah dikembangkan dalam *agile sofware engineering*. Setiap metode memiliki atribut dalam setiap proses pengembangan perangkat lunak. Sejumlah penelitian telah dilakukan untuk mengoptimalkan penerapan teknologi ini dalam pengembangan perangkat lunak. Penelitian ini termasuk mengidentifikasi setiap proses dalam berbagai teknik pengembangan perangkat lunak tangkas yang digunakan. Sukses dalam pengembangan perangkat lunak. Metode *agile sofware engineering* lainnya, XP meminimalkan penciptaan artefak dalam proses pengembangan perangkat lunak, tetapi metode ini mencoba untuk mendekatkan proses pengembangan perangkat lunak kepada pengguna potensial melalui diskusi langsung (Auliasari, 2010).

Dari penelitian-penelitian sebelumnya, tiap metode memiliki prosedur yang baku tanpa meninggalkan prinsip *agile software engineering. Agile modeling* (AM) hadir sebagai suatu pendekatan *software engineering* yang fleksibel disesuaikan dengan metode *software engineering* apapun yang ingin dipakai pada saat pengerjaan pengembangan perangkat lunak. AM bukanlah sebuah metode yang baku namun memiliki langkah kerja yang jelas.

Menurut penelitian sebelumnya, setiap metode memiliki prosedur standar dan tidak menyimpang dari prinsip pengembangan *agile software engineering*. *Agile modeling* (AM) adalah metode pengembangan *agile software engineering*, berlaku untuk metode pengembangan perangkat lunak apa pun yang ingin Anda gunakan dalam pengembangan perangkat lunak. Metode standar, tetapi dengan langkah kerja yang jelas.

Dalam hal ini, AM adalah solusi dari masalah dengan menggunakan metode tradisional. Ini digunakan untuk memodelkan persyaratan perangkat lunak yang harus dibuat secara efektif sehingga dapat diidentifikasi tanpa mengganggu implementasi proses pengembangan perangkat lunak.

Langkah kerja pada *agile modeling*, yaitu:

* + 1. *Collecting artifact*
    2. *Apply right artifact*
    3. *Create several models in paralel*
    4. *Iterate to another artifact*
    5. *Model in small increments*

Pada langkah kerja *collecting artifact* dan *apply right artifact* ada beberapa hal yang diperhatikan, yaitu:

1. *Understanding business environment*
2. *Identify initial scope*
3. *Work closely with stakeholder*

Dalam fase *modeling* pada *agile modeling*, pemodelan yang dibuat untuk menggambarkan kebutuhan sistem, meliputi beberapa hal :

1. Pemodelan kebutuhan calon pengguna.
2. Pemodelan analisa sistem.
3. Pemodelan arsitektur sistem.
4. Pemodelan desain sistem.

Kelebihan dari pengaplikasian *agile modeling* dalam mengembangkan perangkat lunak adalah AM mampu mengoptimalkan dan mengintegrasikan model-model dalam metode *software engineering* yang digunakan agar sesuai dengan tujuan pembangunan perangkat lunak.

Kekurangan dari *agile modeling* adalah AM sangat tergantung terhadap eksistensi model pada *software engineering* yang lain dan tidak ada standar yang baku dalam AM sehingga menimbulkan beda persepsi dalam menggunakan pendekatan ini, sehingga tidak adanya komunikasi yang seragam antara *software developer* yang satu dengan *software developer* yang lain yang sama-sama menggunakan pendekatan AM.

## 2.4 Codeigniter (CI)

CodeIgniter bersifat *open source*, digunakan untuk membangun aplikasi PHP dinamis, yaitu kerangka kerja aplikasi *web*. Tujuan utama kerangka kerja ini adalah untuk membantu pengguna mulai menggunakan aplikasi lebih cepat daripada menulis semua kode dari awal. CodeIgniter menggunakan metode pengembangan *Model View Controller (MVC)* (Noval Aditya, Febriliyan and Radityo Prasestianto, 2013). Ada tiga komponen yang membangun suatu MVC, yaitu :

1) *Model* adalah kode aplikasi yang ditugaskan ke *database* lain, misalnya dalam bentuk file teks atau file xml.

2) *View* adalah kode yang terkait dengan tampilan pengguna. Dalam tampilan, tidak berisi kode atau logika pemrosesan data. Hanya berisi *variabel* data yang akan ditampilkan.

3) *Controller* adalah bagian dari kode yang menghubungkan model dan tampilan. Pada *controller* ini terdapat *class* dan *function* yang menangani *request* data dari *view* ke *model*.

## 2.5 UML (Unified Modeling Languange)

UML adalah bahasa pemodelan, bukan metode. Biasanya, sebuah metode setidaknya terdiri dari prinsip, bahasa pemodelan, dan proses. Bahasa pemodelan (terutama grafik) adalah nama metode yang digunakan untuk desain cepat.

Menurut (Rosa & Salahuddin, 2013) UML adalah salah satu standar bahasa *visual* yang banyak digunakan di industri untuk mendefinisikan persyaratan, melakukan analisis dan desain, dan menggambarkan arsitektur dalam pemrograman berorientasi objek. .UML hanya digunakan untuk pemodelan, sehingga penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML terutama digunakan untuk metodologi berorientasi objek.

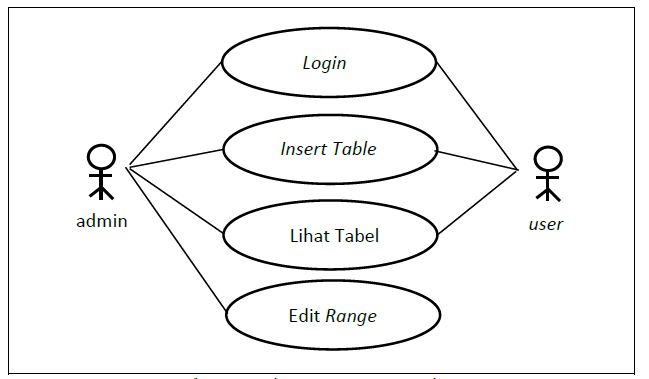
### 2.5.1 Use Case

*Use case* merupakan model perilaku dari sistem informasi yang akan dibuat. *Use case* menggambarkan interaksi antara sistem dan partisipan, termasuk satu atau lebih. *Use case* biasanya digunakan untuk mengetahui fungsi atau tugas apa saja yang termasuk dalam suatu sistem informasi dan siapa yang berhak menggunakan fungsi tersebut (Rosa & Shalahudin, 2013).

**Tabel 2.1** Notasi *Use Case* Diagram

|  |  |
| --- | --- |
| **Symbol** | **Keterangan** |
|  | **Aktor:** berfungsi pemakai sistem, dapat berupa manusia atau sistem *terotomatisasi* lain. Aktor seseorang yang berkomunikasi dengan sistem, yaitu siapa dan apa yang terdapat didalam sistem. Aktor menjelaskan peran bukan pemakai individu dari sistem. Aktor mempunyai nama yang menyatakan peran aktor. |
|  | ***Use Case*:** merupakan cara spesifik penggunaan didalam sistem oleh seorang aktor. Use Case melihatkan interaksi antara aktor yang ada dengan sistem. Use Case mengemukakan suatu kerja yang tampak. |
|  | ***Assosiation*:** hubungan antara aktor dan *use case* |
|  | ***Include*:** perilaku *use case* merupakan bagian dari *use case* lain |
|  | ***Extend*:** perilaku *use case* memperluas *use case* yang lain |

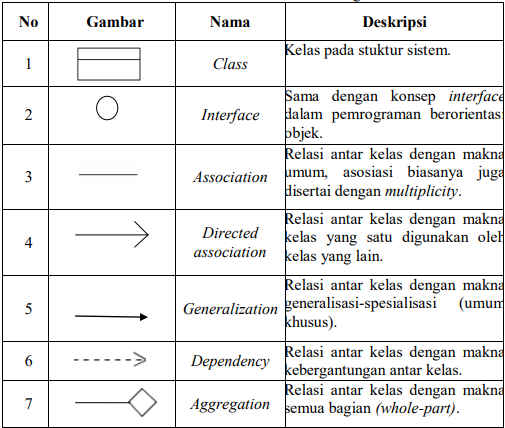
Sumber : (Rosa & Salahuddin, 2013)

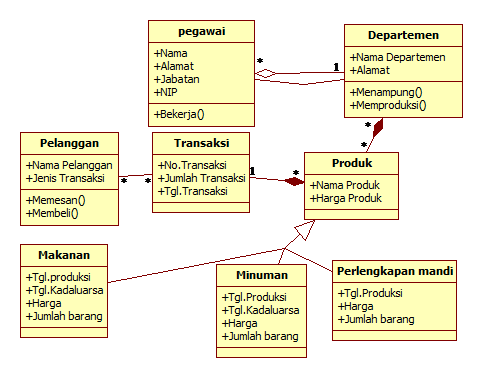


**Gambar 2.4** Use Case Diagram (Rosa & Salahuddin, 2013)

### 2.5.2 Class Diagram

*Class diagram* atau diagram kelas menggambarkan struktur sistem sesuai dengan kelas-kelas yang ditetapkan untuk membangun sistem. Rosa dan Shalahuddin (2013:141). Berikut adalah simbol-simbol yang ada pada diagram kelas:

**Tabel 2.2** Notasi *Class* Diagram  
  
Sumber : (Rosa & Salahuddin, 2013)



**Gambar 2.5** *Class* Diagram Penjualan (Rosa & Salahuddin, 2013)

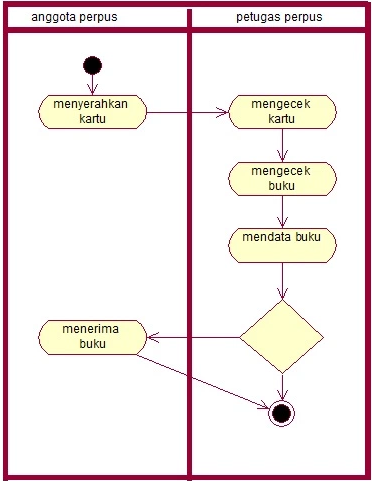
### 2.5.3 Activity Diagram

Diagram aktivitas yang menggambarkan alur kerja atau aktivitas sistem atau proses bisnis atau menu dalam perangkat lunak. Apa yang benar-benar perlu diperhatikan adalah bahwa diagram aktivitas menggambarkan tindakan sistem, bukan apa yang dilakukan peserta, tetapi tindakan yang dilakukan sistem (Rosa & Shalahudin, 2013).

**Tabel 2.3** Notasi *Activity* Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| No | Simbol | Nama | Deskripsi |
| 1 |  | Status Awal | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal. |
| 2 |  | Aktivitas | Aktivitas yang diklakukan sistem biasanya diawali dengan kata kerja. |
| 3 |  | Join | Asosiasi penggabungan dimana jika ada pilihan aktivitas lebih dari satu. |
| 4 |  | Decision | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| 5 |  | Status Akhir | Status akhir yang dilakukan sebuah sistem, sebuah diagram aktivitas memiliki sebuah status akhir |

Sumber : (Rosa & Shalahudin, 2013)



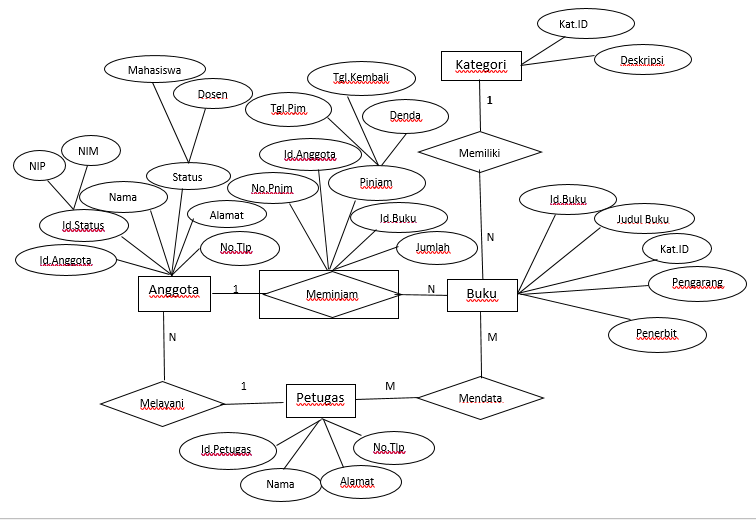
**Gambar 2.6** *Activity* DiagramPerpustakaan (Rosa & Shalahudin, 2013)

## 2.6 Entity Relationship Diagram (ERD)

Menurut (Sutanta, 2011) " *Entity Relationship Diagram* (ERD) adalah model data berorientasi objek". ERD digunakan untuk menjelaskan secara logis hubungan antara data dalam database kepada pengguna. Menggunakan ERD relatif mudah dipahami, bahkan untuk pemula. Modelkan sistem basis data yang sedang dikembangkan dengan bantuan perancang atau analis sistem. Model desainer akan membantu Anda menganalisis dan mendesain database karena dapat menunjukkan jumlah data yang dibutuhkan dan hubungan antara data yang dikandungnya.

**Tabel 2.4** Notasi *Entity Relationship*  Diagram

  
Sumber : (Rosa & Salahuddin, 2013)

  
**Gambar 2.7** *Entity Relationship* DiagramPerpustakaan (Rosa & Salahuddin, 2013)

## 2.7 Conceptual Data Model (CDM)

(Ramadhani, 2010-2011). CDM adalah model yang didasarkan pada asumsi bahwa dunia nyata terdiri dari banyak objek dasar, yang disebut entitas, dan hubungan antara entitas tersebut. CDM biasanya dinyatakan sebagai *entity relationship diagram* (ERD). (Khori, 2018). Adapun manfaat penggunaan CDM dalam perancangan database:

a. Memberikan gambaran yang lengkap dari struktur basis data yaitu arti, hubungan, dan batasan-batasan

b. Alat komunikasi antar pemakai basis data, designer, dan analis.

## 2.8 Physical Data Model (PDM)

(Ramadhani, 2010-2011). PDM adalah model yang menggunakan serangkaian tabel untuk menggambarkan data dan hubungan antara data tersebut. Setiap tabel memiliki banyak kolom, dan setiap kolom memiliki nama yang unik. (Khori, 2018).