

BAB II TINJAUAN PUSTAKA

2.1 Pengertian Sistem Informasi

Sistem dapat didefinisikan dengan mengumpulkan, memproses, menyimpan, menganalisis, menyebarkan informasi untuk tujuan tertentu. Seperti sistem lainnya, sebuah sistem informasi terdiri atas input (data, instruksi) dan output (laporan, kalkulasi) (Hakim dan Anshori, 2019).

2.2 Lembaga Pemasyarakatan

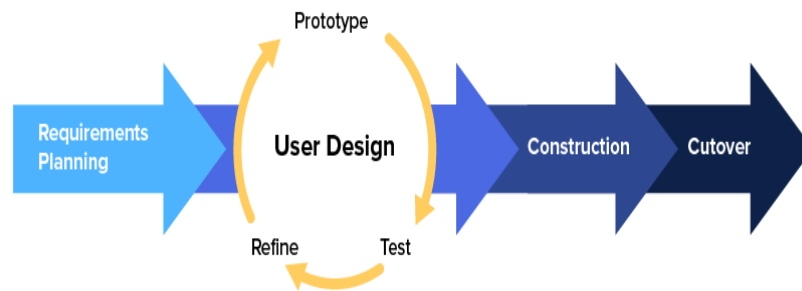
Lembaga Pemasyarakatan (Lapas) adalah tempat untuk melaksanakan pembinaan Narapidana dan Anak Didik Pemasyarakatan. Proses Pembinaan harus memperhatikan pemenuhan hak-hak Narapidana. Salah satu hak narapidana sebagaimana diatur dalam Undang-undang nomor 12 tahun 1995 tentang Pemasyarakatan pasal 14 mengatakan bahwa narapidana berhak menerima kunjungan keluarga, penasihat hukum, atau orang tertentu lainnya.

2.3 *Rapid Application Development (RAD)*

Menurut Rosa and Shalahuddin (2019), metode yang berfokus pada pengembangan aplikasi secara cepat, melalui pengulangan dan *feedback* berulang-ulang. RAD diajukan oleh IBM pada tahun 1980 sampai 1990-an, ketika permintaan terhadap aplikasi semakin meningkat. Dengan banyaknya *demand*, orang-orang di dunia IT harus mencari solusi untuk memenuhi permintaan tersebut. Metode ini merupakan semacam cikal bakal *agile project management*, karena bisa mengikuti *pace* bisnis yang terus berkembang dan juga kebutuhan pasar yang terus meningkat. Pengembangan software pada umumnya seperti waterfall model membutuhkan perencanaan yang terbilang cukup kaku. Klien atau pelanggan seakan ‘dipaksa’ untuk menyetujui banyak hal di awal, tetapi mereka tidak bisa melihat proses pembuatannya.

Keuntungan utama menjalankan *rapid application development* adalah jangka waktu pengembangan lebih cepat. Hal ini dikarenakan *feedback* dari pelanggan cepat didapatkan dan semua perubahan yang dilakukan akan sesuai hasil

tersebut. Akan tetapi, salah satu kekurangan RAD adalah kamu membutuhkan tim berisikan *developer* yang benar-benar memiliki *skill* tinggi dan juga metode ini hanya bisa digunakan untuk proyek yang bisa termodulasi.



Gambar 1.1 *Rapid Application Development (RAD)*
Sumber: (Rosa and Shalahuddin, 2019)

1. Kelebihan Model RAD

Kelebihan metodologi RAD menurut Marakas (2006):

- a. Penghematan waktu dalam keseluruhan fase proyek dapat dicapai.
- b. RAD mengurangi seluruh kebutuhan yang berkaitan dengan biaya proyek dan sumberdaya manusia.
- c. RAD sangat membantu pengembangan aplikasi yang berfokus pada waktu penyelesaian proyek.
- d. Perubahan desain sistem dapat lebih berpengaruh dengan cepat dibandingkan dengan pendekatan SDLC tradisional.
- e. Sudut pandang user disajikan dalam sistem akhir baik melalui fungsi-fungsi sistem atau antarmuka pengguna.
- f. RAD menciptakan rasa kepemilikan yang kuat di antara seluruh pemangku kebijakan proyek.

2. Kelemahan Model RAD

Kelemahan pada pengembangan tersebut dapat dilihat berdasarkan kesesuaian pengembangan yang dilakukan, berikut adalah kelemahan metode pengembang sistem RAD :

- a. Dengan metode RAD, penganalisis berusaha mempercepat proyek dengan terburu-buru.

- b. Kelemahan yang berkaitan dengan waktu dan perhatian terhadap detail. Aplikasi dapat diselesaikan secara lebih cepat, tetapi tidak mampu mengarahkan penekanan terhadap permasalahan-permasalahan perusahaan yang seharusnya diarahkan.
- c. RAD menyulitkan *programmer* yang tidak berpengalaman menggunakan perangkat ini di mana *programmer* dan *analyst* dituntut untuk menguasai kemampuan-kemampuan baru sementara pada saat yang sama mereka harus bekerja mengembangkan sistem

2.3.1 Tahapan Penelitian

Tahapan dalam penelitian sebagai langkah-langkah penelitian yang harus dikerjakan, berikut adalah tahapan penelitian Rapid Application Development.

1. Tahap *Requirements Project*

RAD dimulai dengan menentukan kebutuhan sebuah proyek (*project requirements*). Pada tahap ini, tim perlu menentukan kebutuhan yang ingin dipenuhi dari sebuah proyek. Kebutuhan ini tidak perlu spesifik. Tapi, sifatnya benar-benar umum dan jumlahnya bisa banyak. Baru dari situ, tim akan menentukan mana kebutuhan yang perlu diprioritaskan. Setelah mendapatkan kebutuhan yang jelas, barulah tim menentukan hal-hal yang lebih detail. Misalkan seperti tujuan, timeline, dan budget yang diperlukan.

2. Tahap Membuat *Prototype*

Hal yang selanjutnya dilakukan adalah membuat prototype. Developer secepat mungkin akan membuat prototype dari aplikasi yang diinginkan. Lengkap dengan fitur dan fungsi yang berbeda-beda. Tujuannya, sekadar untuk mengecek apakah *prototype* yang dibuat sudah sesuai dengan kebutuhan klien. Meski begitu, tahap ini bisa saja dilakukan berulang-ulang. Kadang juga melibatkan user untuk testing dan memberikan feedback. Proses ini memungkinkan tim mempelajari error yang mungkin muncul ke depannya. Ini berguna untuk mengurangi error dan debugging. Lewat tahapan ini, tim developer memiliki modal untuk membuat aplikasi yang mudah dipakai, stabil, tidak sering error, dan desainnya yang bagus.

3. Proses Pengembangan dan Pengumpulan *Feedback*

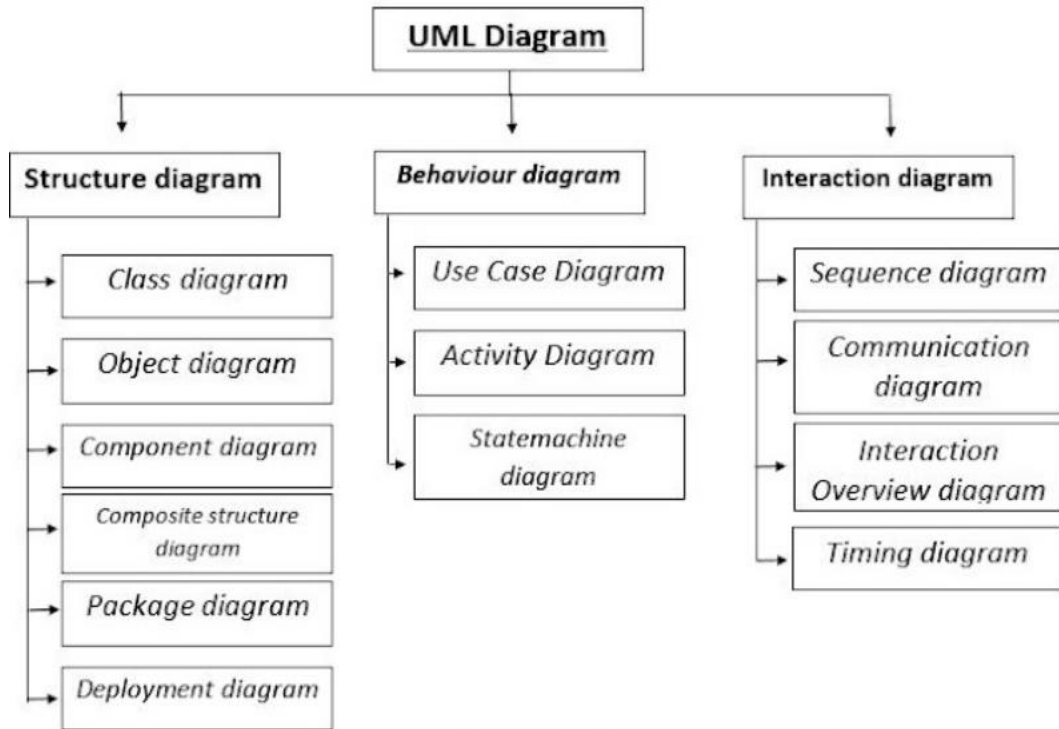
Setelah tahu aplikasi seperti apa yang ingin dibuat, developer mengubah *prototype* ke bentuk aplikasi versi beta sampai dengan final. Jadi, bisa dibilang tahap RAD inilah yang cukup intens. Developer terus-menerus melakukan coding aplikasi, melakukan testing sistem, dan integrasi dengan bagian-bagian lainnya. Karena itulah, developer menggunakan tools dan *framework* yang mendukung RAD agar cepat. Apalagi proses ini terus diulang sambil terus mempertimbangkan feedback dari klien. Baik itu soal fitur, fungsi, interface, sampai keseluruhan aspek dari produk yang dibuat. Jika prosesnya berjalan lancar, developer akan melanjutkan ke langkah berikutnya. Yaitu, finalisasi produk atau implementasi. Kalau pun tidak, proses ini kemungkinan akan terus diulang.

4. Tahap *Implementation* (implementasi).

merupakan tahap pengujian terhadap aplikasi yang dikembangkan. Tahap ini programmer mengembangkan desain menjadi suatu program kemudian dilakukan proses pengujian untuk memeriksa kesalahan sebelum diaplikasikan.

2.4 *Unified Modelling Language (UML)*

Alat pengembang sistem merupakan konsep desain yang digunakan untuk menggambarkan sistem dengan menggunakan diagram. Penyesuaian alat yang digunakan harus sesuai dengan metode pengembangan yang dilakukan salah satunya adalah penerapan *Unified Modelling Language*. Menurut (Rosa dan Shalahuddin, 2019), *Unified Modelling Language* adalah bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. Berikut ini merupakan penjelasan tentang masing-masing diagram yang ada pada *Unified Modelling Language*.




Gambar 1.2 Bagan UML

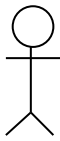

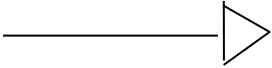
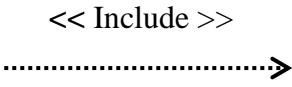
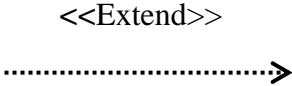
Sumber : (Rosa dan Shalahuddin, 2019)

2.4.1 Use Case Diagram

Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Rosa dan Salahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Use Case Diagram* dapat dilihat pada Tabel 2.1.

Tabel 1.1 Simbol *Use Case Diagram*


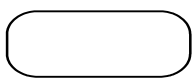
No	Simbol	Deskripsi
1.		<i>Use case</i> : Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal <i>frase</i> nama <i>use case</i> .

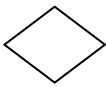

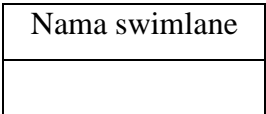

No	Simbol	Deskripsi
2.		Aktor: seseorang/sesuatu yang berinteraksi dengan yang akan dibuat. diluar sistem informasi. Biasanya dinyatakan menggunakan kata benda
3.		Asosiasi (<i>association</i>): merupakan komunikasi antara aktor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan aktor.
4.		Generalisasi (<i>generalization</i>): merupakan hubungan (umum – khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum
5.		Include berarti <i>use case</i> yang ditambahkan akan dipanggil saat <i>use case</i> tambahan dijalankan.
6.		Ekstensi (<i>extend</i>) merupakan <i>use case</i> tambahan ke sebuah <i>use case</i> yang ditambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu.

2.4.2 Activity Diagram

Activity diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa dan Salahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *activity diagram* dapat dilihat pada Tabel 2.2.

Tabel 1.2 Simbol *Activity Diagram*

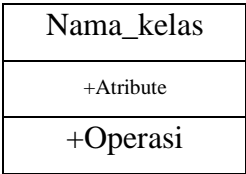


No.	Simbol	Keterangan
1.		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.

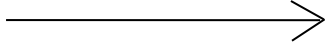
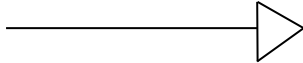
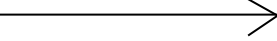
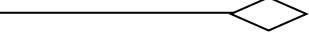
No.	Simbol	Keterangan
3.		Percabangan (<i>Decision</i>) merupakan asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.		Penggabungan (<i>Join</i>) merupakan asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.		Swimlane Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas.
6.		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

2.4.3 Class Diagram

Class diagram mengembangkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Rosa dan Salahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Class Diagram* dapat dilihat pada Tabel 2.3.

Tabel 1.3 Simbol *Class Diagram*

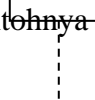


No.	Simbol	Deskripsi
1.		Kelas pada struktur sistem.
2.	Antar Muka/Interface  Nama_Interface	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3.	Asosiasi / Association 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>symbol</i>

No.	Simbol	Deskripsi
4.	Asosiasi Berarah / <i>Digunakan Association</i> 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>symbol</i> .
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Ketergantungan / dependency 	Relasi antar kelas dengan makna ketergantungan antar kelas.
7.	Agregasi / <i>aggregation</i> 	Relasi antar kelas dengan makna semua bagian (<i>whole-part</i>)

2.4.4 Sequence Diagram

Diagram *Sequence* menjelaskan bagaimana suatu operasi itu dilakukan; message (pesan) apa yang dikirim dan kapan pelaksanaannya (Rosa A.S. dan Shalahuddin, 2019). Berikut simbol *sequence diagram* pada Tabel 2.4.

Tabel 1.4 Simbol *Sequence Diagram*

No.	Simbol	Deskripsi
1.	<i>Object lifeline</i> 	Menggambarkan panjang kehidupan suatu objek selama scenario sedang di buat contohnya
2.	<i>Activation</i> 	Dimana proses sedang dilakukan oleh <i>object</i> atau <i>class</i> untuk memenuhi pesan atau perintah
3.	<i>Message</i> 	Sebuah anak panah yang mengindikasikan pesan diantara objek. Dan objek dapat mengirimkan pesan ke dirinya sendiri

2.5 *CodeIgniter*

CodeIgniter adalah *Framework* untuk bahasa pemrograman PHP, yang dibuat Rick Ellis pada tahun 2006. *CodeIgniter* memiliki banyak fitur yang membantu para pengembang PHP untuk dapat membuat aplikasi secara mudah dan cepat serta memiliki sifat yang fleksibel dapat dikembangkan dalam perangkat *web*, *dekstop* maupun *mobile* (Raharjo, 2018). *Codeigniter* adalah sebuah aplikasi gratis yang berupa kerangka kerja untuk membangun website menggunakan bahasa pemrograman PHP (Heru, 2018).

2.6 PHP (*Hypertext Preprocessor*)

PHP (*Hypertext Preprocessor*) merupakan bahasa pemrograman untuk membuat website atau situs dinamis dan mengenai rangkaian bahasa pemrograman antara *client side scripting* dan *server side scripting* (Heru, 2018).

PHP merupakan bahasa pemrograman yang digunakan secara luas untuk penanganan, pembuatan dan pengembangan sebuah situs web dan biasanya bersamaan dengan HTML (Oetomo and Maharginono, 2020).

Sehingga PHP merupakan bahasa pemrograman yang digunakan oleh pengembang untuk membuat sistem *website* dengan kumpulan bahasa HTML dan *script* lainnya.

2.7 Pengujian *Black Box Testing*

Black box testing yaitu pengujian perangkat lunak dari segi pendefinisian fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan (Rosa dan Salahuddin, 2019).

Pengujian yang dilakukan dengan membuat kasus yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji dilakukan harus dibuat dengan benar dan salah, seperti proses *login* “Jika user memasukan *username* dan *password* yang benar maka dapat *login* ?”.

2.8 Penelitian Terdahulu

Penelitian terdahulu bertujuan untuk mendapatkan bahan perbandingan dan acuan. Selain itu, untuk menghindari anggapan kesamaan dengan penelitian ini. Maka dalam kajian pustaka ini peneliti mencantumkan hasil-hasil penelitian terdahulu sebagai berikut:

Tabel 1.5 Penelitian Terdahulu

Nama	Judul	Masalah	Metode	Hasil Analisis
(Hayka, Priyanto and Nasution, 2017)	Sistem Informasi Administrasi Pelayanan Narapidana pada Lapas Kelas IIA di Kota Pontianak	Lembaga Pemasarakatan Klas II A Kota Pontianak saat ini proses pendataan dan rekapitulasi data narapidana dilakukan secara manual. Pendataan data narapidana dilakukan dengan cara mencatat data narapidana ke dalam buku besar dan pelepasan bersyarat dicatat ke dalam buku pembinaan. Kemudian	Waterfall	Aplikasi sistem informasi administrasi pelayanan narapidana dapat mempertemukan antara keluarga yang bersangkutan dengan narapidana dalam bentuk menerima informasi tentang narapidana yang diinputkan oleh admin di lapas.
(Prasetyo, Herlinda and Surajoyo, 2021)	Perancangan Aplikasi Pencatatan Pengunjung Narapidana pada Lembaga Pemasarakatan Terbuka Kelas II B Jakarta	Sistem pencatatan pengunjung yang berjalan di Lembaga Pemasarakatan Terbuka Kelas II B Jakarta masih bersifat manual dengan memakai formulir cetakan kertas sehingga data beresiko untuk hilang dan rusak, sehingga pencarian data pengunjung sulit dilakukan	Waterfall	peneliti dapat menyimpulkan bahwa pencatatan berbasis komputer dapat menghemat penggunaan kertas karena data disimpan secara digital, resiko kerusakan atau kehilangan datanya lebih sedikit, pemrosesan dan pencarian data lebih cepat, data lebih akurat dibanding dengan ditulis tangan.

