Perancangan dan Implementasi *Finite Automata* pada Simulasi *Vending Machine*

[1)] Jessica Christiani Irawan, [2)] M. A. Ineke Pakereng, [3)] Ramos Somya

[1)]Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Jl. Diponegoro 52-60, Salatiga 50771, Indonesia, jck-catrise@yahoo.co.id,
[2)] Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Jl. Diponegoro 52-60, Salatiga 50771, Indonesia, inekep200472@yahoo.com
[3)] Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Jl. Diponegoro 52-60, Salatiga 50771, Indonesia, ramos.somya@gmail.com

Abstract

Language and Automata Theory is one of information technology's component. This theory is the based of ideas and model of a computation system. One example of the Automata implementation is a vending machine. A vending machine can sell goods to customers without an operator to operate the machine. However, in Indonesia,the presence of vending machines are rarely seen. In this study, the author will try to learn about how a vending machine works, as well as to apply part of the Language and Automata Theory to design and create a simulation of a vending machine. As the result of the design and implementation of Finite Automata in vending machine simulation, it can be concluded that Finite Automata can be used for basic logic to make a vending machine simulation. Through this application simulation, user can get an experience in operating a vending machine and learn how to use the vending machine.

Keywords*: Language and Automata Theory, Finite Automata, Vending Machine, Simulation.*

Abstrak

Teori bahasa dan automata merupakan salah satu komponen ilmu informatika. Teori inilah yang mendasari ide dan model dari sebuah sistem komputasi. Salah satu contoh penerapan automata adalah pada *vending machine*. *Vending machine* dapat menjual barang-barang untuk konsumen tanpa adanya seorang operator. Namun, di Indonesia, keberadaan *vending machine* masih dapat dikatakan langka. Dalam penelitian ini, akan dipelajari cara kerja *vending machine*, sekaligus diterapkan bagian dari Teori Bahasa dan Automata untuk merancang dan membuat simulasinya. Berdasarkan hasil perancangan dan implementasi *Finite Automata* pada simulasi *vending machine*, dapat diambil kesimpulan bahwa *Finite Automata* dapat dijadikan sebagai logika dasar untuk membuat simulasi *vending machine*. Melalui aplikasi simulasi ini, diharapkan *user* dapat memperoleh pengalaman dalam mengoperasikan sebuah *vending machine* serta mengetahui cara menggunakan sebuah *vending machine*.

Kata Kunci : Teori Bahasa dan Automata, *Finite Automata, Vending Machine,* Simulasi.

1. Pendahuluan

Penggunaan automata pada perangkat lunak terutama pada pembuatan kompiler bahasa pemrograman. Bahasa pemprograman bertindak sebagai sarana komunikasi antara manusia dan permasalahannya dengan komputer, yang dipakai untuk membantu memperoleh pemecahan. Sebagai keluaran dari automata, bahasa memungkinkan penyampaian gagasan dan pemikiran manusia, sedangkan sebagai ilmu yang juga mempelajari mengenai mesin abstrak, automata dapat membaca *input* berupa string dari alphabet yang diberikan dari *input file*.

Salah satu contoh penerapan automata adalah pada *vending machine*. *Vending machine* dapat menjual barang-barang untuk konsumen tanpa adanya seorang operator. Sebuah *vending machine* biasanya menjual satu macam barang, misalnya makanan ringan, minuman, koran, dan lain sebagainya Konsumen yang ingin memperoleh barang yang ada di *vending machine* tersebut harus memasukkan uang sejumlah harga yang tertera ke lubang uang yang tersedia pada *vending machine* itu, kemudian menekan tombol barang yang dipilih, dan barang tersebut akan keluar.

Namun, di Indonesia, keberadaan *vending machine* masih dapat dikatakan langka. Oleh karena itu, muncullah sebuah keinginan untuk mencoba mempelajari cara kerja *vending*

machine, sekaligus menerapkan bagian dari Teori Bahasa dan Automata untuk merancang dan membuat simulasinya.

2. Kajian Pustaka

Contoh penerapan Teori Bahasa dan Automata adalah Penggunaan Algoritma *Greedy* Dalam Aplikasi *Vending Machine*. Pada penerapan ini ditekankan mengenai cara agar sebuah *vending machine* dapat memberikan uang kembalian secara optimal. Yang dimaksud optimal adalah uang kembalian mempunyai jumlah koin paling sedikit diantara semua kemungkinan uang kembalian. Algoritma *Greedy* bisa diterapkan sebagai salah satu cara untuk menyelesaikan jumlah uang kembalian tersebut [3].

Melalui penelitian ini, akan dirancang dan diimplementasikan *Finite Automata* pada simulasi *vending machine*.

Konsep Bahasa dan Automata

Teori Bahasa dan Automata termasuk pada komponen pertama dari dua komponen tersebut. Teori Bahasa dan Automata banyak diterapkan pada perancangan digital, pembuatan bahasa pemrograman, dan kompilator [4].

Automata merupakan suatu sistem yang terdiri atas sejumlah *state* berhingga, dimana setiap *state* menyatakan informasi mengenai *input* sebelumnya, dan dapat pula dianggap sebagai memori mesin [6].

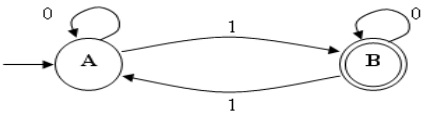
Input pada mesin automata dianggap sebagai bahasa yang harus dikenali oleh mesin. Selanjutnya mesin automata membuat keputusan yang mengindikasikan diterima atau tidaknya *input* tersebut, sehingga mesin automata dapat dipakai untuk menghasilkan suatu bahasa yang aturannya ditentukan oleh bahasa tersebut.

Finite Automata

Finite Automata (FA), yang bisa disebut juga sebagai *Finite State Automata* (FSA), automata dengan *state* berhingga, merupakan suatu model matematika dari suatu sistem yang menerima *input* dan menghasilkan *output* diskrit [5]. FA memiliki *state* yang banyaknya berhingga (terbatas), dan dapat berpindah-pindah dari satu *state* ke *state* lain. Perubahan *state* ini dinyatakan dengan fungsi transisi. *State* adalah kondisi atau keadaan atau kedudukan.

Sebuah *Finite Automata* M dinyatakan dengan lima tupel, yaitu (Q, Σ, δ, S, F), di mana:

Q = himpunan *state*, Σ = himpunan alfabet masukan, δ = fungsi transisi, S = *state* awal (*initial state)*, F = himpunan *state* akhir (*final state*)



Gambar 1 Contoh *State* Diagram *Finite Automata*

- Gambar lingkaran menyatakan *state*
- Label pada lingkaran adalah nama *state* tersebut
- Busur panah menyatakan transisi atau perpindahan *state*
- Gambar lingkaran yang didahului sebuah busur panah tanpa label menyatakan *state* awal
- Gambar lingkaran ganda menyatakan *final state*

Maka: Q = {A, B}, Σ = {0,1}, S = {A}, F = {B}

δ = fungsi transisi

δ (A, 0) = A, δ (A, 1) = B, δ (B, 0) = B, δ (B, 1) = A

Dari fungsi transisi tersebut, dapat dibuat tabel transisi seperti pada Tabel 1.

Tabel 1 Tabel Transisi Berdasarkan Gambar 1

δ	0	1
A	A	B
B	B	A

Contoh bila string yang masuk adalah 1011, maka string tersebut bergerak dari Start ke *state* A, kemudian membaca karakter ’1’ dan berpindah ke *state* B, yang merupakan *state* tujuan dari hasil pembacaan karakter ’1’. Kemudian string selanjutnya yang dibaca adalah ’0’. Karena *state* tujuan dari pembacaan karakter ’0’ adalah B sendiri, maka *state* tidak berpindah. Selanjutnya membaca karakter ’1’. Dari *state* B berpindah ke *state* A yang merupakan *state* tujuan setelah membaca karakter ’1’. Setelah itu, karakter ’1’ dibaca dan *state* berpindah ke *state* B. Pembacaan string berhenti karena karakter sudah habis. *State* terakhir yang ditempati adalah *state* B. Karena *state* B berada dalam himpunan *final state*, maka string ’1011’ diterima oleh *Finite Automata* tersebut.

Simulasi

Simulasi adalah tiruan dari proses dunia nyata atau sistem. Simulasi menyangkut pembangkitan proses serta pengamatan dari proses untuk menarik kesimpulan dari sistem yang diwakili. Simulasi mempelajari atau memprediksi sesuatu yang belum terjadi dengan cara meniru atau membuat model sistem yang dipelajari. [7]

Vending Machine

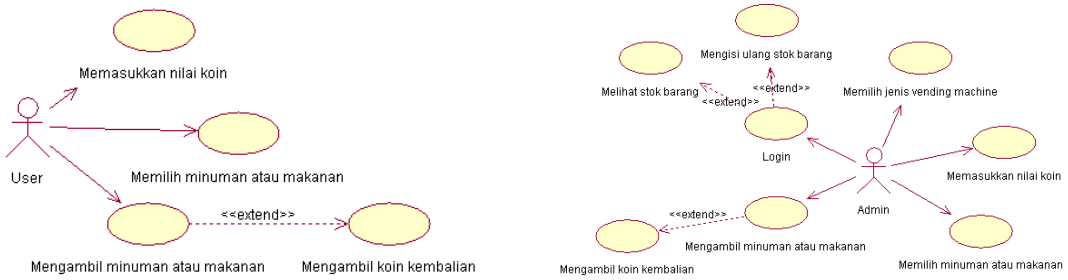
Sebuah *vending machine* memungkinkan seseorang untuk membeli barang yang terdapat di dalam mesin tersebut secara otomatis, tanpa adanya seorang penjaga atau kasir, yang menjadi perantara proses jual-beli tersebut. Proses penjualan ditangani sepenuhnya oleh mesin. Pembeli hanya tinggal memasukkan uang dan memilih barang yang akan dibeli.

Vending machine pertama yang ditemukan adalah hasil karya Hero dari Alexandria, seorang ahli mesin dan matematika pada abad pertama. Mesin tersebut dapat menerima koin, lalu mengeluarkan air suci dalam jumlah tertentu. Cara kerjanya masih dibilang sederhana. Ketika koin dimasukkan, koin tersebut jatuh ke dalam sebuah panci yang terhubung dengan tuas. Tuas tersebut kemudian membuka sebuah katup yang membuat sejumlah air keluar. Panci tersebut akan terus miring akibat berat koin, sampai akhirnya koin tersebut jatuh dan mengembalikan posisi tuas serta menutup katup air [8].

3. Perancangan Sistem

Sistem dirancang dengan menggunakan UML (*Unified Modelling Language*) yang terdiri dari *use case diagram, activity diagram, sequence diagram, class diagram* dan *deployment diagram.*

Sebuah *use case diagram* berfungsi untuk mendeskripsikan tindakan sistem dari sudut pandang *user*, sebagai deskripsi fungsional dari sebuah sistem dan proses utamanya, serta menjelaskan secara visual siapa yang menggunakan sistem dan bagaimana interaksinya. Aplikasi simulasi *vending machine* ini digunakan oleh *user* yang terdiri dari *user* dan *admin*. *Use case diagram* untuk *user* dapat dilihat pada Gambar 2.



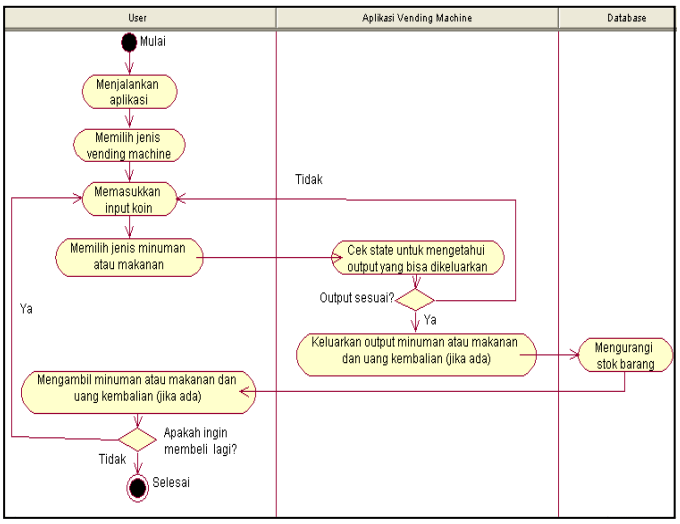
Gambar 2 (a) *Use Case Diagram User* **(b)** *Use Case Diagram Admin*

Setelah menjalankan aplikasi, *user* dapat memakai fasilitas yang dapat dilakukan terhadap aplikasi, seperti memilih jenis *vending machine*, memasukkan nilai koin untuk memulai simulasi, memilih jenis minuman atau makanan yang akan dibeli, mengambil minuman atau makanan yang keluar jika proses transaksi berhasil, dan juga mengambil kembalian jika ternyata nilai uang yang dimasukkan berlebih.

Use case diagram untuk *admin* dapat dilihat pada Gambar 2(b). Sebenarnya tugas utama *admin* berkaitan dengan stok barang yang ada pada *database*, yaitu mengawasi stok barang pada *vending machine*. Untuk memperoleh akses ke *database*, *admin* harus melakukan *login* terlebih dahulu. Setelah itu, barulah *admin* dapat melihat stok barang dan menggunakan fasilitas isi ulang stok barang (re-stok) bila stok barang di *vending machine* sudah habis. Dapat menjalankan simulasi *vending machine* merupakan fasilitas tambahan bagi *admin*.

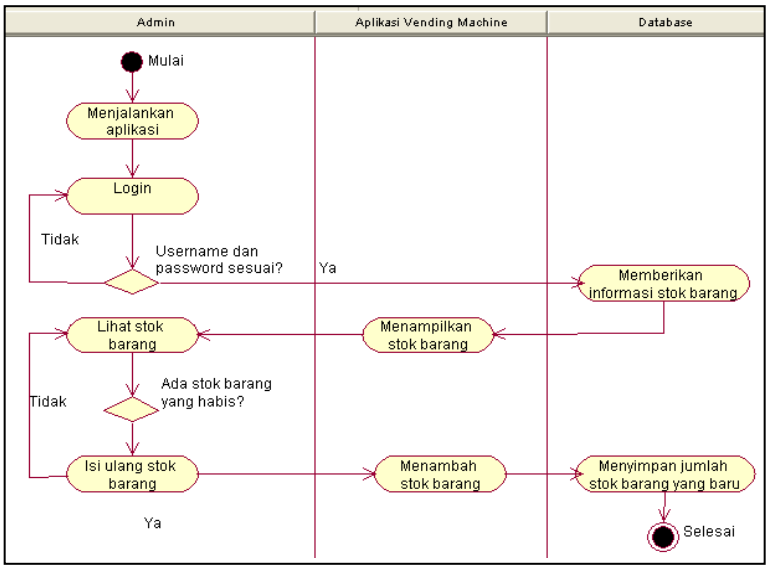
Activity diagram menggambarkan aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran dimulai, apa keputusan yang terjadi, dan bagaimana aktivitas tersebut berakhir.

Jika aplikasi simulasi ini digunakan oleh *user*, maka proses interaksi antara *user* dengan aplikasi simulasi *vending machine* dapat digambarkan seperti pada Gambar 3.



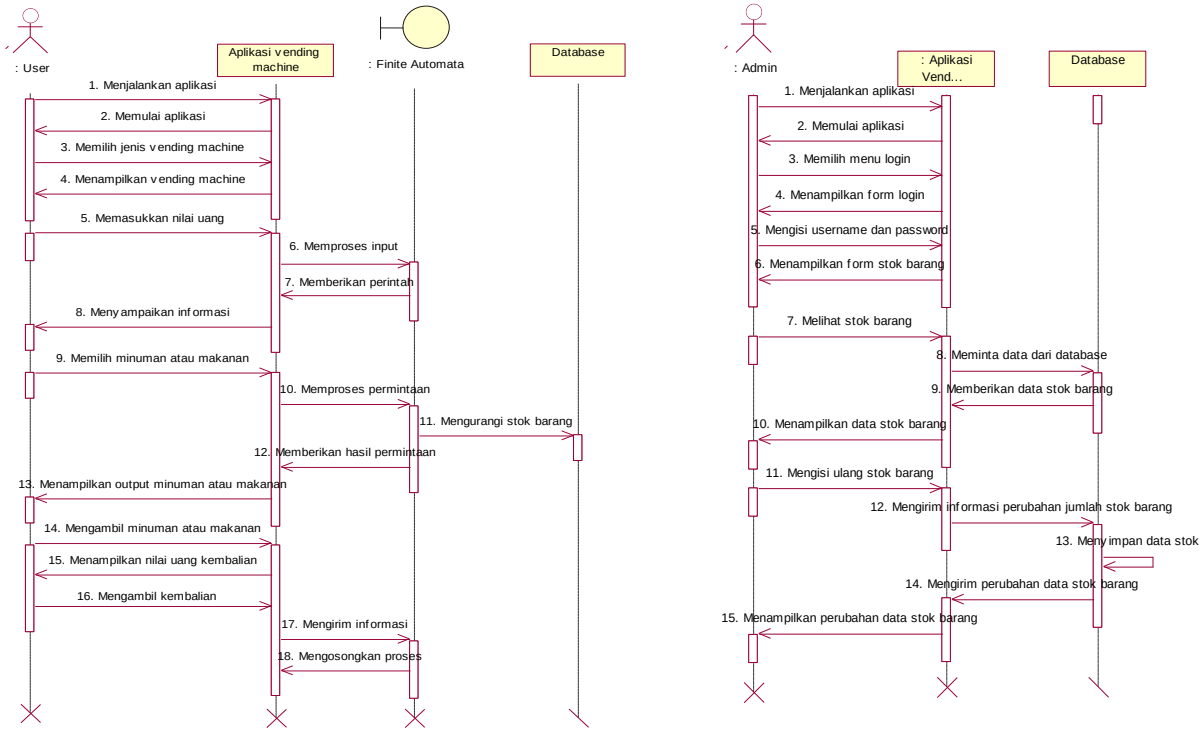
Gambar 3 *Activity Diagram* Interaksi *User* dengan Aplikasi *Vending Machine*

User memulai simulasi dengan menjalankan aplikasi *vending machine*. Kemudian *user* dapat memilih jenis *vending machine* yang akan dijalankan, yaitu *vending machine* minuman ringan atau makanan ringan. Setelah aplikasi *vending machine* muncul, *user* dapat memasukkan nilai koin pada aplikasi tersebut.



Gambar 4 *Activity Diagram Admin*

Gambar 4 menjelaskan *activity diagram* untuk tugas utama *admin*. *Admin* menjalankan aplikasi, lalu untuk mengakses *database*, *admin* harus melakukan *login* terlebih dahulu. Jika *username* dan *password* yang diisikan sesuai, maka *admin* dapat melihat stok barang yang ada di *database*. Jika ada barang yang stoknya habis, maka *admin* dapat mengisi ulang stok barang tersebut dan data jumlah barang di *database* akan diperbaharui.



Gambar 5 (a) *Sequence Diagram* Interaksi *User* dengan Aplikasi *Vending Machine*, (b) *Sequence Diagram Admin*

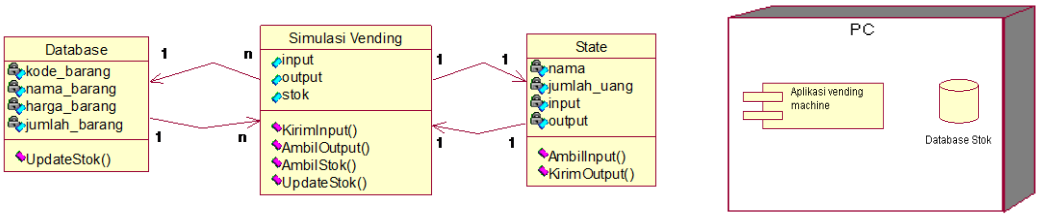
Gambar 5 (a) merupakan *sequence diagram user* yang berinteraksi dengan aplikasi simulasi *vending machine*. Pada diagram tersebut, *user* menjalankan aplikasi, memilih jenis *vending machine* (minuman ringan atau makanan ringan), lalu memberikan *input* kepada aplikasi berupa nilai uang koin. Aplikasi menerima *input* tersebut lalu memprosesnya dengan *Finite Automata*. Aplikasi memberikan informasi kepada *user* mengenai jumlah uang yang dimasukkan. Jika melihat jumlah uang sudah cukup, maka *user* memilih minuman atau makanan yang akan dibeli. Aplikasi kembali memproses permintaan tersebut dengan *Finite Automata*. *Output* minuman keluar. *User* mengambil minuman atau makanan tersebut berikut kembaliannya, dan aplikasi mengirim informasi bahwa proses sudah selesai. Proses simulasi dikosongkan dan dimulai dari awal lagi.

Pada Gambar 5(b) ditunjukkan *sequence diagram admin* yang berinteraksi dengan aplikasi simulasi *vending machine*. *Admin* menjalankan aplikasi, lalu melakukan *login*. Setelah itu, *admin* dapat melihat isi *database* stok barang pada *vending machine* dengan perantara aplikasi. Jika perlu mengisi ulang stok barang, maka *admin* dapat menggunakan fasilitas re-stok dan *database* akan menyimpan jumlah stok barang yang baru. Pembaharuan data ini kemudian akan ditampilkan kembali kepada *admin*.

Class diagram menggambarkan struktur dan deskripsi, *class*, *package*, serta hubungannya satu sama lain. *Class diagram* ditampilkan dalam beberapa kelas serta paket yang ada dalam sistem yang sedang dikembangkan. Gambar 6(a) merupakan *class diagram* sistem pada proses simulasi *vending machine*.

Dari Gambar 6(a) dapat diketahui bahwa *class* simulasi *vending machine* memiliki atribut *input*, *output*, dan stok. Atribut-atribut ini berfungsi untuk menyimpan data untuk proses simulasi. Atribut *input* menampung *input* yang diberikan *user* pada aplikasi, lalu mengirimkan *input* tersebut ke *class* State dengan *method* Kirim*Input*().*Method* Ambil*Output*() akan

mengambil *output* dari *class* State, sedangkan AmbilStok() berfungsi untuk mengambil jumlah stok barang yang ada di *database*.



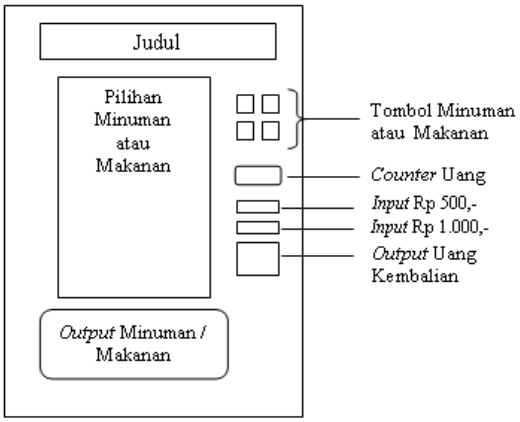
Gambar 6 (a) *Class Diagram* Sistem, (b) *Deployment Diagram* Sistem

*Update*Stok() berfungsi untuk memperbaharui jumlah barang yang ada di *database* bila *user* melakukan pembelian di *vending machine*. *Class* State memiliki atribut nama, yaitu nama *state*, lalu jumlah_uang untuk mendeklarasikan jumlah uang yang ada pada *state* tersebut, dan juga input dan output, yang berfungsi untuk menampung *input* dari *class* Simulasi Vending yang diterima oleh *method* Ambil*Input*() dan mendefinisikan *output* yang harus dikeluarkan menurut *input* yang didapat dan mengirimkannya ke *class* Simulasi Vending dengan *method* Kirim*Output*(). Sementara itu *database* memiliki *field* tabel kode_barang, nama_barang, harga_barang, dan juga jumlah_barang untuk mengidentifikasi stok barang. Operasi yang dibutuhkan untuk *database* ini adalah *Update*Stok() untuk memperbaharui jumlah stok barang.

Deployment diagram berfungsi untuk menampilkan rancangan fisik jaringan dalam sistem. Gambar 6 (b) adalah *deployment diagram* sistem bila aplikasi simulasi *vending machine* dibuat dalam sebuah jaringan. Pada Gambar 6(b), yang dibutuhkan hanyalah sebuah PC untuk menjalankan aplikasi simulasi *vending machine*. Pada PC tersebut terdapat dua buah kompunen, yaitu aplikasi *vending machine* dan basisdata untuk menyimpan data stok barang.

Perancangan Antarmuka Aplikasi

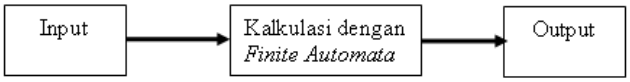
Rancangan *interface* (antarmuka) aplikasi simulasi *vending machine* yang akan dibuat adalah seperti yang ditunjukkan Gambar 7.



Gambar 7 Rancangan Antarmuka Aplikasi *Vending Machine*

Perancangan *Prototype*

Rencana proses langkah kerja aplikasi simulasi *vending machine* adalah seperti yang ditunjukkan pada Gambar 8.



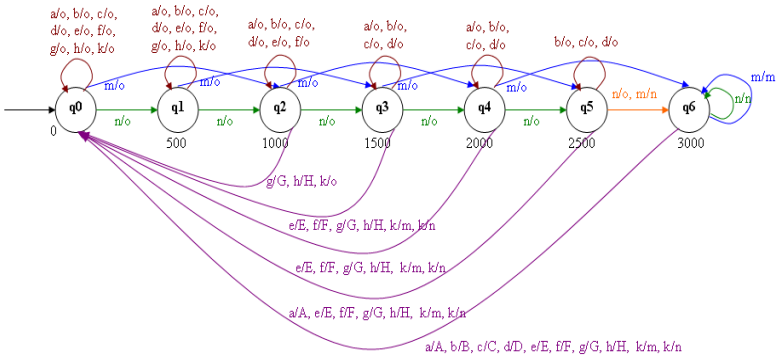
Gambar 8 Langkah-langkah Kerja Aplikasi Simulasi *Vending Machine*

Keterangan proses pada Gambar 8 tersebut adalah a) *Input*: *Input* berupa masukan angka 500 atau 1000 sebagai uang, serta jenis minuman atau makanan yang dimasukkan oleh *user* yang ingin mencoba simulasi *vending machine*. b) Kalkulasi dengan *Finite Automata*: Setelah *input* selesai dimasukkan, program akan melakukan proses kalkulasi sesuai konsep *Mealy machine* untuk menentukan *output* apa yang akan dikeluarkan oleh *vending machine* tersebut sesuai dengan *input* yang diterima. Agar dapat melaksanakan proses kalkulasi sesuai konsep *Mealy machine*, perlu dilakukan tahap-tahap sebagai berikut:

1. Merancang diagram *state*

Untuk memulai rancangan *prototype* aplikasi, yang perlu dilakukan adalah membuat sebuah diagram *state* sebagai patokan pendefinisian tupel dan alur program.

Pada batasan masalah telah disebutkan bahwa *input* adalah uang koin Rp 500,- dan Rp 1.000,-, dan *output*nya adalah empat macam minuman ringan atau makanan ringan. Empat macam minuman tersebut adalah *Frestea* kotak seharga 2.500 rupiah, *Sprite*, *Fanta* dan *Coca-Cola* kaleng seharga 3.000 rupiah. Sedangkan empat macam makanan ringannya adalah *Cheetos* dan *JetZ* seharga 1.000 rupiah, serta *Lays* dan *Chitato* seharga 1.500 rupiah. Berdasarkan batasan masalah tersebut, maka dapat dibuat diagram *state* seperti Gambar 9.



Gambar 9 Rancangan Diagram *State* Simulasi *Vending Machine*

Ada sebelas *input* pada mesin ini, yaitu a (memilih *Frestea*), b (memilih *Fanta*), c (memilih *Coca-cola*), d (memilih *Sprite*), e (memilih *Lays*), f (memilih *Chitato*), g (memilih *Cheetos*), h (memilih *JetZ*), m (uang koin Rp 1.000,-), n (uang koin Rp 500,-), dan k (mengambil uang kembalian).

Sedangkan *output*nya ada sembilan, antara lain A (mengeluarkan *Frestea*), B (mengeluarkan *Sprite*), C (mengeluarkan *Sprite*), D (mengeluarkan *Coca-Cola*), E (mengeluarkan *Lays*), F (mengeluarkan *Chitato*), G (mengeluarkan *Cheetos*), H (mengeluarkan *JetZ*), dan o (tidak melakukan apa-apa).

2. Mendefinisikan tupel

Mealy machine didefinisikan dengan enam tupel, dengan rumus:

$Me=(Q, \Sigma, \delta, S, \Delta, \lambda)$

Dimana:

Q = himpunan *state*, Σ = himpunan simbol *input*, δ = fungsi transisi ($\delta : Q \times \Sigma \rightarrow Q$), S = *state* awal (*initial state)*, Δ = himpunan simbol *output*, λ = fungsi *output* untuk setiap transisi ($\delta : Q \times \Sigma \rightarrow \Delta$)

Sehingga dapat dibuat definisi sebagai berikut:

$Q = \{q0, q1, q2, q3, q4, q5, q6\}$

$\Sigma = \{a, b, c, d, e, f, g, h, m, n, k\}$

δ = fungsi transisi

$\delta(q0, a) = q0$; $\delta(q0, b) = q0$; $\delta(q0, c) = q0$; $\delta(q0, d) = q0$; $\delta(q0, e) = q0$; $\delta(q0, f) = q0$; $\delta(q0, g) = q0$; $\delta(q0, h) = q0$; $\delta(q0, m) = q2$; $\delta(q0, n) = q1$; $\delta(q0, k) = q0$;

$\delta(q1, a) = q1$; $\delta(q1, b) = q0$; $\delta(q1, c) = q1$; $\delta(q1, d) = q1$; $\delta(q1, e) = q1$; $\delta(q1, f) = q0$; $\delta(q1, g) = q1$; $\delta(q1, h) = q1$; $\delta(q1, m) = q3$; $\delta(q1, n) = q2$; $\delta(q1, k) = q1$;

δ(q2, a) = q2; δ(q2, b) = q2; δ(q2, c) = q2; δ(q2, d) = q2; δ(q2, e) = q2; δ(q2, f) = q2;
δ(q2, g) = q0; δ(q2, h) = q0; δ(q2, m) = q4; δ(q2, n) = q3; δ(q2, k) = q0;
δ(q3, a) = q3; δ(q3, b) = q3; δ(q3, c) = q3; δ(q3, d) = q3; δ(q3, e) = q0; δ(q3, f) = q0;
δ(q3, g) = q0; δ(q3, h) = q0; δ(q3, m) = q5; δ(q3, n) = q4; δ(q3, k) = q0;
δ(q4, a) = q4; δ(q4, b) = q4; δ(q4, c) = q4; δ(q4, d) = q4; δ(q4, e) = q0; δ(q4, f) = q0;
δ(q4, g) = q0; δ(q4, h) = q0; δ(q4, m) = q6; δ(q4, n) = q5; δ(q4, k) = q0;
δ(q5, a) = q0; δ(q5, b) = q5; δ(q5, c) = q5; δ(q5, d) = q5; δ(q5, e) = q0; δ(q5, f) = q0;
δ(q5, g) = q0; δ(q5, h) = q0; δ(q5, m) = q6; δ(q5, n) = q6; δ(q0, k) = q0;
δ(q6, a) = q0; δ(q6, b) = q0; δ(q6, c) = q0; δ(q6, d) = q0; δ(q6, e) = q0; δ(q6, f) = q0;
δ(q6, g) = q0; δ(q6, h) = q0; δ(q6, m) = q6; δ(q6, n) = q6; δ(q6, k) = q0;
Bila dipetakan dalam tabel, menjadi seperti pada Tabel 4.

Tabel 4 Tabel Transisi *Input*

δ	a	b	c	d	e	f	g	h	m	n	k
q0	q0	q0	q0	q0	q0	q0	q0	q0	q2	q1	q0
q1	q1	q1	q1	q1	q1	q1	q1	q1	q3	q2	q1
q2	q2	q2	q2	q2	q2	q2	q0	q0	q4	q3	q0
q3	q3	q3	q3	q3	q0	q0	q0	q0	q5	q4	q0
q4	q4	q4	q4	q4	q0	q0	q0	q0	q6	q5	q0
q5	q0	q5	q5	q5	q0	q0	q0	q0	q6	q6	q0
q6	q0	q0	q0	q0	q0	q0	q0	q0	q6	q6	q0

Tabel 4 menjelaskan tentang transisi *input*, yaitu perpindahan *state* apabila ada *input* tertentu yang masuk.

S = {q0}

Δ = {A, B, C, D, o}

λ = fungsi *output* untuk setiap transisi

λ(q0, a) = o ; λ(q0, b) = o ; λ(q0, c) = o ; λ(q0, d) = o ; λ(q0, e) = o ; λ(q0, f) = o ;
λ(q0, g) = o ; λ(q0, h) = o ; λ(q0, m) = o ; λ(q0, n) = o ; λ(q0, k) = o ;
λ(q1, a) = o ; λ (q1, b) = o ; λ(q1, c) = o ; λ(q1, d) = o ; λ(q1, e) = o ; λ (q1, f) = o ;
λ(q1, g) = o ; λ(q1, h) = o ; λ(q1, m) = o ; λ(q1, n) = o ; λ(q1, k) = o ;
λ(q2, a) = o ; λ(q2, b) = o ; λ(q2, c) = o ; λ(q2, d) = o ; λ(q2, e) = o ; λ(q2, f) = o ;
λ(q2, g) = G ; λ(q2, h) = H ; λ(q2, m) = o ; λ(q2, n) = o ; λ(q2, k) = o ;
λ(q3, a) = o ; λ(q3, b) = o ; λ(q3, c) = o ; λ(q3, d) = o ; λ(q3, e) = E ; λ(q3, f) = F ;
λ(q3, g) = G ; λ(q3, h) = H ; λ(q3, m) = o ; λ(q3, n) = o ; λ(q3, k) = m, n ;
λ(q4, a) = o ; λ(q4, b) = o ; λ(q4, c) = o ; λ(q4, d) = o ; λ(q4, e) = E ; λ(q4, f) = F ;
λ(q4, g) = G ; λ(q4, h) = H ; λ(q4, m) = o ; λ(q4, n) = o ; λ(q4, k) = m, n ;
λ(q5, a) = A ; λ(q5, b) = o ; λ(q5, c) = o ; λ(q5, d) = o ; λ(q5, e) = E ; λ(q5, f) = F ;
λ(q5, g) = G ; λ(q5, h) = H ; λ(q5, m) =o ; λ(q5, n) = o ; λ(q5, k) = m, n ;
λ(q6, a) = A ; λ(q6, b) = B ; λ(q6, c) = C ; λ(q6, d) = D ; λ(q6, e) = E ; λ(q6, f) = F ;
λ(q6, g) = G ; λ(q6, h) = H ; λ(q6, m) = m ; λ(q6, n) = n ; λ(q6, k) = m,n ;
Bila dipetakan dalam tabel, menjadi seperti pada Tabel 5.

Tabel 5 Tabel Transisi *Output*

δ	a	b	c	d	e	f	g	h	m	n	k
q0	o	o	o	o	o	o	o	o	o	o	o
q1	o	o	o	o	o	o	o	o	o	o	o
q2	o	o	o	o	o	o	G	H	o	o	o
q3	o	o	o	o	E	F	G	H	o	o	m,n
q4	o	o	o	o	E	F	G	H	o	o	m,n
q5	A	o	o	o	E	F	G	H	o	o	m,n
q6	A	B	C	D	E	F	G	H	m	n	m,n

Tabel 5 menunjukkan transisi *output*. Lewat Tabel 6 ini, dapat diketahui *output* apa yang harus dikeluarkan oleh aplikasi *vending machine*. Contohnya pada *state* q6. Bila *input* yang masuk adalah a, sesuai Gambar 13, maka *output*-nya adalah A (*Frestea*). Bila mendapat *input* b, *output*-nya adalah B (*Fanta*). Bila mendapat *input* c, *output*-nya yaitu C (*Coca-Cola*). Bila mendapat *input* d, *output*-nya adalah D (*Sprite*). Bila mendapat *input* e, *output*-nya adalah E (*Lays*). Bila mendapat *input* f, *output*-nya adalah F (*Chitato*). Bila mendapat *input* g, *output*-nya adalah g (*Cheetos*). Bila mendapat *input* h, *output*-nya adalah H (*JetZ*). Bila mendapat *input* m (1.000), maka *output*-nya adalah m lagi, dan bila *input*nya n (500), *output*-nya juga n. Sebenarnya *state* q6 adalah *state* terakhir, sehingga bila diberi *input* nilai uang lagi (m dan n), maka *state* tidak akan beranjak dan memuat nilai kelebihan uang tersebut. Bila *input*nya adalah k (mengambil kembalian), maka kelebihan dari m dan n tadi akan dikeluarkan sesuai dengan nilainya.

c) *Output*: Dari hasil kalkulasi, diperoleh *output* yang akan dikeluarkan aplikasi. *Output* berupa gambar minuman atau makanan yang dipilih, serta angka yang menyatakan uang kembalian (jika ada).

4. Hasil dan Pembahasan

Form Menu Utama

Pada aplikasi simulasi *vending machine*, ada dua tipe *user*, yaitu *user* dan *admin*. Ketika aplikasi dijalankan, akan keluar *Form* Menu Utama seperti yang terlihat pada Gambar 10.

Gambar 10 Tampilan *Form* Menu Utama pada Aplikasi Simulasi *Vending Machine*

Ada dua jenis *vending machine*, yaitu *vending machine* minuman ringan dan makanan ringan. *User* dapat memilih jenis *vending machine* dengan memilih pada *Combo Box* pilihan, kemudian memilih tombol "OK". Jika memilih *vending machine* minuman ringan, maka *Form* yang tampil seperti pada Gambar 11. Sedangkan jika memilih *vending machine* makanan ringan, maka *Form* yang tampil adalah seperti pada Gambar 11.

Form Simulasi *Vending Machine*

Tampilan kedua *Form* Simulasi *Vending Machine* dapat dilihat pada Gambar 11. Karena pada dasarnya fungsi dan fasilitas yang terdapat pada kedua jenis *Form* Simulasi *Vending Machine* adalah sama, maka sebagai contoh, yang akan dibahas adalah *Form* Simulasi *Vending Machine* untuk makanan ringan.

Gambar 11 Tampilan *Form* Simulasi *Veding Machine* Minuman dan Minuman Ringan

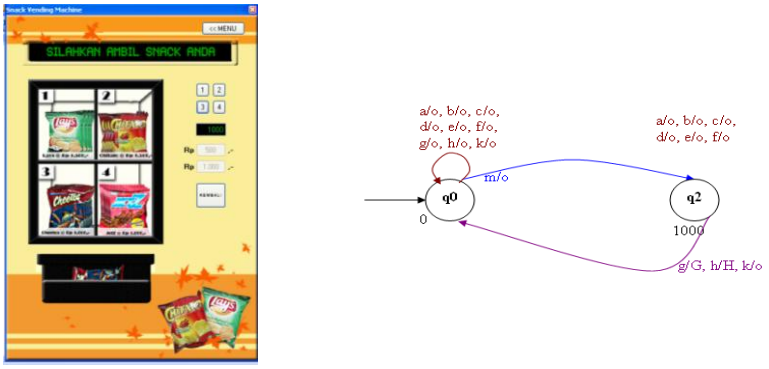
Bila setelah *Form* keluar, *user* langsung menekan tombol 1, maka tidak ada *output* yang dikeluarkan. Pada saat tersebut, *state* berada pada q0. Sesuai dengan rancangan diagram *state*, maka proses yang terjadi pada *state* ini adalah seperti pada Gambar 14(a).

Gambar 14 (a) Proses Saat *State* Berada Di Q0, (b) Perpindahan *State* Setelah Uang 1.000 Masuk

Pada Gambar 14(a), bila *input* yang masuk adalah 'a' (tombol 1), maka outputnya adalah 'o' yang berarti tidak melakukan apa-apa. Dalam hal ini, *output* yang diberikan oleh aplikasi sudah benar. Selanjutnya, misalnya *user* memilih tombol Rp 1.000,- yang ada pada aplikasi, maka menurut diagram *state*, seharusnya *state* berpindah dari q0 ke q2. Pada *state* q2, nilai uang adalah 1.000, seperti yang terlihat pada Gambar 14(b). Sedangkan *Text Box* yang berfungsi sebagai *counter* juga menunjukkan angka '1000' seperti pada Gambar 15. Ini berarti, proses perpindahan *state* sudah benar.

Gambar 15 Nilai Yang Ditunjukkan Oleh *Counter* Saat Uang 1.000 Masuk

Selanjutnya, apabila *user* memilih tombol '3', maka *output* yang akan keluar adalah makanan ringan *Cheetos* seperti pada Gambar 16(a). Bila dibandingkan dengan diagram *state* pada Gambar 16 (b), maka proses dan *output* yang dikeluarkan oleh aplikasi sudah sesuai dengan diagram tersebut.



Gambar 16 (a) *Output* Ketika Tombol 3 Dipilih, (b) Diagram *State* Ketika *User* Memilih *Cheetos*

Form Login Admin

Tipe *user admin* memiliki hak untuk mengakses *database* yang berisi stok persediaan barang yang ada pada masing-masing *vending machine*.

Gambar 17 (a) Tampilan *Form Login Admin*, (b) Tampilan *Form Admin*

Untuk mengakses *database* tersebut, *admin* dapat memilih *Menu Strip* "Menu" pada *Form* Menu Utama dan memilih "Admin". *Admin* harus terlebih dahulu mengisikan *username* dan *password* pada *Form Login Admin* seperti pada Gambar 17 (a). Jika proses *login* berhasil, maka *admin* dapat melihat *Form Admin* yang tampilannya seperti yang terlihat pada Gambar 17 (b). Pada *Form Admin* ini, *admin* dapat melihat stok barang yang masih ada pada *vending machine*, jumlah pembelian tiap-tiap jenis barang, barang yang paling banyak dibeli, serta fasilitas re-stok untuk mengisi ulang persediaan barang yang sudah atau hampir habis.

5. Simpulan

Berdasarkan hasil perancangan dan implementasi *Finite Automata* pada simulasi *vending machine*, dapat diambil kesimpulan bahwa *Finite Automata* dapat dijadikan sebagai logika dasar untuk membuat simulasi *vending machine*. Lewat rancangan *state diagram* berdasarkan konsep *Mealy machine* yang telah dibuat, maka apliksi simulasi *vending machine* dapat dibuat dan hasil dari setiap *input* yang dipilih oleh *user* pada aplikasi sesuai dengan hasil rancangan tersebut. Aplikasi simulasi *vending machine* dibuat dengan tampilan dan cara kerja menyerupai *vending machine* asli, dengan tujuan agar *user* dapat memperoleh pengalaman dalam mengoperasikan sebuah *vending machine* serta mengetahui cara menggunakan sebuah *vending machine*.

6. Daftar Pustaka

[1] Nugroho, Aryo. Penggunaan Algoritma Greedy Dalam Aplikasi Vending Machine. 2007. Fakultas Teknologi Informasi Institut Teknologi Bandung.

[2] Utdirartatmo, Firrar. 2001. *Teori Bahasa dan Otomata*. Yogyakarta: J & J Learning.

[3] Hariyanto, Bambang. 2004. *Teori Bahasa, Otomata, dan Komputasi serta terapannya*. Bandung: Penerbit Informatika.

[4] Sudkamp, Thomas A. 2005. *Languages and Machines: An Introduction to the Theory of Computer Science*. Boston: Addison Wesley Publishing Company.

[5] Sokolowsky, J.A, Banks C.M. 2009. *Principles of Modelling and Simulation: A Multidisciplinary Approach*. New Jersey: John Wiley & Sons.

[6] Segrave, Kerry. 2002. *Vending Machines: An American Social History*. Jefferson: Mc Farland & Company.