IMPLEMENTASI METODE FINITE STATE MACHINE UNTUK PENGELOLAAN STATE DINAMIS PADA GAME THE ARCHITECT: REBIRTH OF SERAYA

Satria Adji Subagyo, Paulus Harsadi
Informatika, Universitas Tiga Serangkai Surakarta
Jl. K.H. Samanhudi No. 84-86 57142 Laweyan Jawa Tengah
satriagale9@gmail.com

ABSTRAK

Pemahaman kausalitas kebijakan publik oleh generasi muda merupakan tantangan krusial di Indonesia, namun metode pembelajaran konvensional yang bersifat linear dinilai kurang efektif dalam menumbuhkan keterlibatan emosional peserta didik. Permasalahan utama yang diangkat dalam penelitian ini adalah tingginya kompleksitas manajemen kondisi (*state management*) pada game simulasi konvensional yang umumnya mengandalkan pendekatan percabangan kondisional (*if-else/switch-case*) yang masif, sehingga menghasilkan *spaghetti code* yang menghambat skalabilitas dan *maintainability* sistem. Tujuan penelitian adalah merancang dan mengimplementasikan arsitektur Finite State Machine (FSM) berbasis *State Pattern* menggunakan bahasa C# pada Godot Engine 4.x pada *serious game* berjudul "The Architect: Rebirth of Seraya" untuk mengelola tiga sistem utama: Sistem Kebijakan, *Dynamic Difficulty Adjustment* (EDDA), dan Sistem Skor. Rencana penyelesaian masalah dilakukan melalui penerapan *State Pattern* yang memisahkan setiap kondisi ke dalam kelas mandiri dengan metode Enter(), Execute(), dan Exit(), serta menggunakan kerangka Game Development Life Cycle (GDLC) sebagai metodologi pengembangan. Pengujian dilakukan melalui *Black Box Testing* untuk validasi fungsionalitas dan Game Engagement Questionnaire (GEQ) terhadap 50 responden. Hasil menunjukkan seluruh 26 butir instrumen valid (r hitung > 0,20) dengan Cronbach Alpha seluruh dimensi di atas 0,70. Rata-rata keseluruhan skor GEQ sebesar 4,087 dari skala 5,00 (kategori Baik), mengonfirmasi bahwa implementasi FSM berhasil mewujudkan sistem game yang modular, deterministik, dan efektif sebagai media edukasi interaktif.

Kata kunci: *Finite State Machine, serious game, kebijakan publik, GDLC, Godot Engine, city-builder*

1. PENDAHULUAN

Pemahaman kebijakan publik merupakan tantangan krusial bagi generasi muda Indonesia. Data Badan Pusat Statistik (BPS) per Maret 2023 menunjukkan Indonesia masih menghadapi isu kemiskinan sebesar 25,9 juta jiwa dan prevalensi stunting yang tinggi, mencerminkan hambatan nyata dalam mencapai kesejahteraan sosial-ekonomi yang merata. Metode pembelajaran konvensional yang bersifat linear dan satu arah sering gagal menjembatani kerumitan data tersebut sehingga isu krusial hanya dipahami secara teoretis tanpa keterlibatan emosional [1].

Sebagai respons inovatif, *serious game* hadir bukan sekadar media hiburan, melainkan sebagai instrumen simulasi yang terbukti efektif dalam mempromosikan pengambilan keputusan strategis [2]. Genre simulasi pembangunan kota (*city-building*) khususnya memiliki kemampuan alami untuk memodelkan sistem kompleks secara visual, memposisikan pemain sebagai pembuat keputusan yang harus menyeimbangkan berbagai aspek yang saling bertentangan seperti pertumbuhan ekonomi versus kelestarian lingkungan [3].

Tantangan utama pengembangan game simulasi kebijakan yang kompleks adalah risiko terjadinya *spaghetti code* akibat pendekatan konvensional percabangan kondisional (*if-else/switch-case*) yang masif dan bertumpuk. Fenomena ini menyebabkan alur logika program menjadi kusut, sulit ditelusuri, dan mengakibatkan rendahnya keterbacaan kode serta menghambat skalabilitas pengembangan [4]. Sebagai solusi teknis, Finite State Machine (FSM) diintegrasikan sebagai arsitektur modular yang memungkinkan pengelolaan transisi kondisi secara deterministik dan terstruktur.

Berdasarkan permasalahan tersebut, tujuan penelitian ini adalah merancang dan mengimplementasikan arsitektur Finite State Machine (FSM) berbasis *State Pattern* menggunakan bahasa C# pada Godot Engine 4.x untuk mengelola tiga sistem utama pada game “The Architect: Rebirth of Seraya”, yaitu Sistem Kebijakan, Dynamic Difficulty Adjustment (EDDA), dan Sistem Skor, sekaligus memvalidasi efektivitasnya sebagai media edukasi kausalitas kebijakan publik bagi generasi muda.

Rencana penyelesaian masalah dalam penelitian ini dilakukan melalui beberapa tahapan. Pertama, merancang arsitektur FSM yang memisahkan setiap kondisi game ke dalam kelas-kelas *state* mandiri berdasarkan *State Pattern* dengan metode Enter(), Execute(), dan Exit(). Kedua, mengimplementasikan arsitektur tersebut pada Godot Engine 4.x menggunakan bahasa C# .NET 8. Ketiga, mengintegrasikan algoritma EDDA berbasis *rule-based* untuk penyesuaian kesulitan secara adaptif dan real-time. Keempat, melakukan pengujian menggunakan Black Box Testing dan GEQ terhadap 50 responden untuk mengukur fungsionalitas sistem serta tingkat keterlibatan pemain.

2. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Beberapa penelitian sebelumnya telah mengeksplorasi penggunaan *Finite State Machine* (FSM) dalam pengembangan game. Sintaro et al. (2023) menerapkan FSM berbasis *State Pattern* khusus untuk mengelola pergerakan karakter utama agar lebih stabil dan rapi. Sementara itu, Estevam et al. (2023) meneliti penggunaan *State Machine* sebagai arsitektur sistem game secara keseluruhan untuk meningkatkan skalabilitas dan modularitas. Perbedaan utama dengan penelitian ini adalah fokus implementasi FSM yang tidak hanya pada satu aspek, melainkan pada tiga sistem makro sekaligus, yaitu sistem Kebijakan, EDDA (*Dynamic Difficulty Adjustment*), dan Skor dalam konteks *serious game* edukasi. Selain itu, penelitian ini secara khusus mengintegrasikan FSM dengan algoritma adaptif dan divalidasi menggunakan GEQ (*Game Engagement Questionnaire*) pada aspek edukasi kebijakan publik. Terkait aspek adaptivitas, Mortazavi dan Moradi (2024) melakukan tinjauan literatur sistematis mengenai metode DDA dan menegaskan bahwa DDA terbukti mampu meningkatkan keterlibatan pemain dengan menyesuaikan tantangan secara otomatis. Penelitian ini melangkah lebih jauh dari sekadar tinjauan literatur dengan mengimplementasikan varian EDDA yang berorientasi pada keterlibatan (*engagement*) yang dieksekusi melalui arsitektur FSM. Dalam hal implementasi teknis pada Godot Engine, Alchimowicz dan Plechawska-Wójcik (2024) memberikan dasar yang kuat melalui analisis perbandingan bahasa *scripting*, yang menunjukkan keunggulan signifikan C# dibanding GDScript untuk operasi aritmatika berat dan perulangan kompleks. Temuan ini dimanfaatkan dalam penelitian ini sebagai landasan pemilihan C# untuk mengimplementasikan kalkulasi algoritma EDDA secara *real-time*.

2.2. Finite State Machine (FSM)

FSM adalah model komputasi matematis yang membagi logika perilaku sistem menjadi sejumlah status (*states*) terbatas, transisi, dan aksi. Dalam konteks pengembangan game, FSM menjadi standar industri untuk mengendalikan perilaku sistem karena sifatnya yang deterministik dan ringan secara komputasi. Sintaro et al. (2023) membuktikan bahwa penggunaan FSM berbasis *State Pattern* berhasil mengelola logika pergerakan karakter secara stabil dan rapi [5]. Dalam penelitian ini, FSM diimplementasikan sebagai *entity controller component* yang mengelola tiga sistem utama secara modular.

2.3. Dynamic Difficulty Adjustment (DDA) & EDDA

DDA didefinisikan sebagai metode komputasi yang memungkinkan sistem mengubah parameter permainan secara otomatis dan *real-time* berdasarkan kemampuan dan performa pemain [6]. Penelitian ini mengimplementasikan varian *Engagement-Oriented Dynamic Difficulty Adjustment* (EDDA) berbasis aturan (*rule-based*) yang menjadikan keterlibatan pemain (*player engagement*) sebagai metrik utama, dengan algoritma yang memantau variabel ekonomi seperti saldo koin dan populasi untuk mendeteksi tanda-tanda kebosanan atau frustrasi pemain.

2.4. Serious Game & Game Engagement Questionnaire

Serious game didefinisikan sebagai permainan yang dirancang dengan tujuan melampaui hiburan, yakni untuk pendidikan atau penyampaian pesan sosial. Chen & He (2024) menerangkan bahwa keterlibatan emosional dalam game dapat meningkatkan retensi pengetahuan dan empati pemain [1]. Game Engagement Questionnaire (GEQ) adalah instrumen psikometrik baku untuk mengukur tingkat keterlibatan pemain berdasarkan dimensi Immersion, Presence, Flow, dan Absorption [7].

2.5. Godot Engine 4 & C#

Godot Engine 4.x adalah *game development framework* lintas platform bersumber terbuka yang menerapkan arsitektur unik berbasis hierarki Node dan Scene. Alchimowicz & Plechawska-Wójcik (2024) membuktikan bahwa C# memiliki keunggulan signifikan dibandingkan GDScript dalam kecepatan komputasi untuk operasi aritmatika berat dan perulangan kompleks yang dieksekusi ribuan kali per detik [8], menjadikannya pilihan ideal untuk kalkulasi algoritma adaptif secara *real-time*.

3. METODE PENELITIAN

Pengembangan sistem menggunakan kerangka Game Development Life Cycle (GDLC), sebuah metodologi rekayasa perangkat lunak yang dirancang khusus untuk memfasilitasi integrasi kompleks antara logika komputasi adaptif, pemrosesan algoritma matematis, serta sinkronisasi aset visual dan audio. GDLC terdiri dari lima tahapan: Inisialisasi, Pra-Produksi, Material Collecting, Produksi, dan Rilis [9].

3.1. Tahapan Pengembangan Sistem

Pengembangan sistem dibagi menjadi lima tahapan utama:

a. Inisialisasi: Penentuan genre *City Building* dan target platform PC Desktop (Windows 64-bit).
b. Pra-Produksi: Perancangan *Game Design Document* (GDD), pemetaan alur FSM, dan perumusan matematis algoritma EDDA.
c. Produksi: Implementasi kode menggunakan bahasa C# .NET 8 pada *engine* Godot 4.x.
d. Pengujian: Validasi *Black Box Testing* terhadap seluruh skenario aksi pemain.
e. Rilis: Kompilasi akhir aplikasi *standalone*.

3.2. Arsitektur FSM yang Diimplementasikan

FSM diimplementasikan pada tiga sistem utama game menggunakan pola *State Pattern* dalam C#. Setiap sistem memiliki kelas abstrak *BaseState* yang

diwarisi oleh kelas-kelas *state* konkret. Fungsi kunci adalah *ChangeDDAState(DDAState newState)* yang mengontrol transisi antar status secara deterministik.

Tabel 1. Tiga Sistem FSM pada Game

Sistem	State-state	Trigger Transisi
Kebijakan	*Locked, Active, Cooldown*	Keputusan pemain, timer cooldown
EDDA/DDA	*Inactive, Monitoring, DisasterActive*	Mode adaptif, bencana aktif
Skor/SDGs	*Calculating, Updating, Idle*	Interval timer evaluasi

Tabel 1 merangkum tiga sistem FSM dan transisi antar-kondisinya yang dikelola secara modular dan independen.

3.3. Implementasi FSM: Sistem DDA (EDDA)

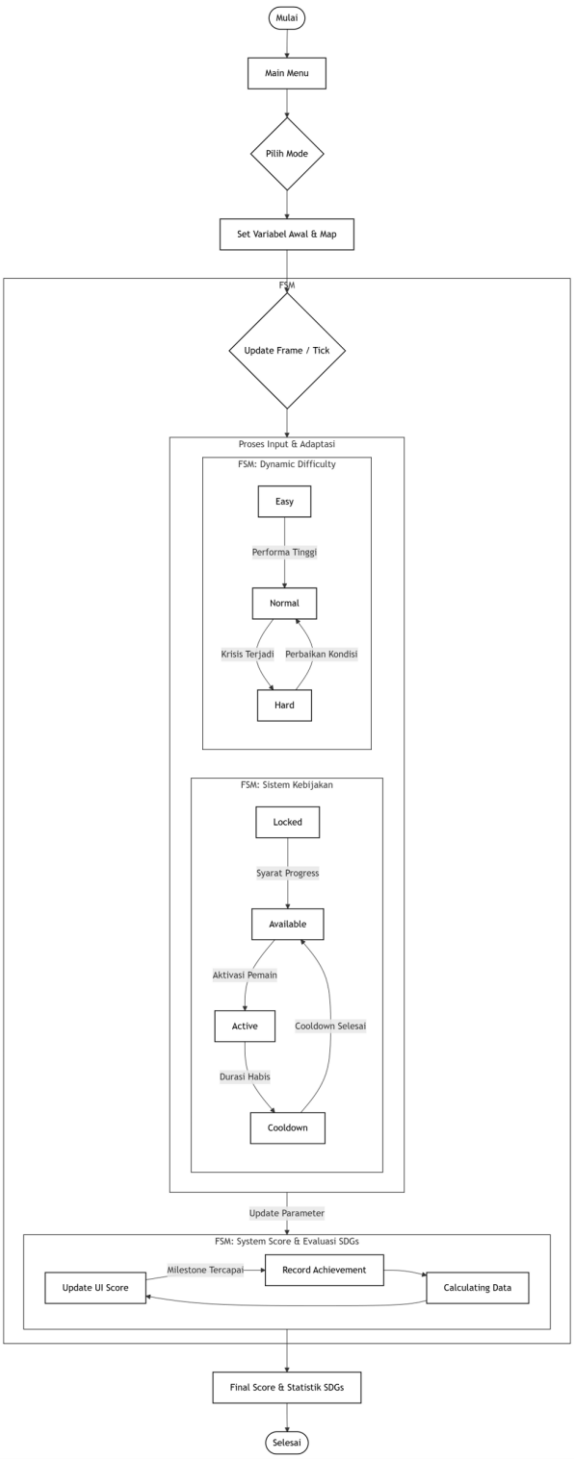
FSM DDA memiliki tiga state utama. State *Inactive* aktif pada mode Kasual di mana seluruh timer AI dimatikan, menjadikan beban CPU dari AI sebesar 0%. State *Monitoring* menjalankan tiga timer independen: *_ddaEvaluationTimer* (setiap 5 detik) untuk kalkulasi *Player Skill Score* (PSS); *_ddaEffectTimer* (setiap 10 detik) untuk mengeksekusi bantuan atau penalti; dan *_externalDisasterTimer* (dasar 900 detik) yang diperbarui dinamis oleh *_difficultyModifier*. State *DisasterActive* berfungsi sebagai *safety net* yang menghentikan timer bencana baru saat krisis sedang berlangsung agar skor pemain tidak turun tidak adil.

3.4. Kalkulasi Player Skill Score & Klasifikasi Kesulitan

PSS dikalkulasi dengan normalisasi variabel multivariat menggunakan teknik *Min-Max Scaling* parsial. Saldo uang dinormalisasi dengan *Mathf.Clamp(playerMoney / 300.0f, 0, 100)* agar setara skala dengan variabel persentase lain. Klasifikasi kesulitan menggunakan enumerasi empat level: Pemula (PSS < 30, modifier 0,5x), Normal (30–60, 1,0x), Menengah (61–80, 1,5x), dan Ahli (> 80, 2,0x). Transisi modifier dilakukan dengan fungsi *Mathf.MoveToward* delta 0,05f untuk menghasilkan transisi yang organik.

3.5. Implementasi Teknis FSM: Sistem Kebijakan

FSM Sistem Kebijakan mengontrol aliran status kebijakan secara atomik. Kebijakan bertransisi dari kondisi *Locked* (belum tersedia) menuju *Active* (sedang berlaku) hingga *Cooldown* (jeda pasca-aktif) secara sekuensial tanpa tumpang tindih logika. Setiap state diimplementasikan sebagai kelas terpisah dengan metode *Enter(), Execute(),* dan *Exit(),* memastikan isolasi logika yang sempurna.



Gambar 1. Diagram Alur Implementasi FSM

Gambar 1 mengilustrasikan alur kerja keseluruhan sistem Finite State Machine (FSM) yang diimplementasikan pada game "The Architect: Rebirth of Seraya". Alur dimulai dari titik *Main Menu* di mana pemain memilih mode permainan (Kasual atau Adaptif), kemudian sistem menginisialisasi variabel awal dan peta melalui proses *Set Variabel Awal & Map*. Setelah inisialisasi selesai, sistem memasuki siklus utama (*Update Frame/Tick*) yang berjalan secara kontinu selama permainan berlangsung. Pada setiap *tick*, tiga modul FSM dieksekusi secara

berurutan: pertama, FSM Sistem Kebijakan yang mengelola transisi status kebijakan dari *Locked* → *Available* → *Active* → *Cooldown* → kembali *Available* berdasarkan keputusan pemain dan timer; kedua, FSM Dynamic Difficulty yang mengklasifikasikan performa pemain ke dalam kondisi *Easy*, *Normal*, atau *Hard* berdasarkan evaluasi metrik kota, dengan mekanisme perbaikan kondisi jika krisis terdeteksi; dan ketiga, FSM Sistem Skor & Evaluasi SDGs yang memantau pencapaian *milestone* kota, merekam *achievement*, lalu mengkalkulasi data skor secara dinamis. Setelah semua kondisi kemenangan atau kekalahan terpenuhi, sistem keluar dari siklus utama dan menghasilkan *Final Score & Statistik SDGs* sebagai output akhir yang ditampilkan kepada pemain.

3.6. Metode Pengujian

Pengujian dilakukan melalui dua instrumen. Pertama, *Black Box Testing* untuk memvalidasi akurasi transisi state fungsional berdasarkan skenario input-output yang telah dirancang. Kedua, GEQ dengan 7 dimensi (4 dimensi GEQ standar dan 3 dimensi edukasi) disebarkan kepada 50 responden menggunakan kombinasi *purposive sampling* dan *snowball sampling* dengan formula Lemeshow (1997, d=0,15) yang menghasilkan minimum 43 responden. Analisis statistik menggunakan Korelasi Pearson dan Cronbach's Alpha.

4. HASIL DAN PEMBAHASAN

4.1. Tampilan Aplikasi Game

Gambar 1. Antarmuka Permainan dan Panel Pemantau Algoritma FSM

Gambar 2 menampilkan *Heads-Up Display* (HUD) utama dalam simulasi tata kota. Panel 'Monitor FSM' pada sisi kiri secara *real-time* memvisualisasikan kalkulasi *State* yang diproses di latar belakang. Panel ini menyajikan transparansi data state dan transisi yang terjadi selama permainan.

4.2. Hasil Black Box Testing FSM

Tabel 2. Hasil Black Box Testing FSM

ID	Skenario	Hasil Aktual	Status
BB-01	Konstruksi valid: uang cukup	Uang berkurang, bangunan terinstansiasi presisi, SFX berbunyi	Valid
BB-02	Konstruksi ditolak: uang kurang	Bangunan batal, peringatan UI merah, SFX Eror berbunyi	Valid
BB-03	Validasi syarat makro gagal (Edukasi < 70%)	Peletakan diblokir, umpan balik alasan spesifik	Valid
BB-04	Demolisi bangunan aktif (Bulldoze FSM)	Bangunan terhapus, refund 50%, kemiskinan naik	Valid

Seluruh 4 skenario Black Box Testing menghasilkan status Valid, membuktikan transisi FSM berjalan deterministik sesuai spesifikasi GDD.

4.3. Pengujian Algoritma Adaptif

Tabel 3. Hasil Pengujian Logika FSM & EDDA

Skenario	Parameter	Respons FSM & EDDA
Kritis (Game Over)	Kas ≈ 0, kemiskinan tinggi	PSS < 30 (Pemula). FSM hentikan timer bencana, aktifkan Relief Event (+150 koin)
Dominasi (Overpowered)	Kas surplus, faksi 100% puas	PSS > 80 (Ahli). FSM paksa trigger krisis tingkat tinggi
Pembuktian Random Forest	Pajak ekstrem, polusi maksimal	Majority Voting mengklasifikasikan 'Kapitalis/Oligarki' pada layar akhir

Ketiga skenario logika terkonfirmasi Valid, membuktikan sinkronisasi FSM dan EDDA berjalan sesuai rancangan.

Evaluasi pada aspek logika diterapkan demi menguji akurasi kalkulasi matematis serta responsivitas dinamis dari algoritma EDDA, sistem FSM, hingga modul perhitungan parameter kota berdasarkan berbagai situasi spesifik yang disimulasikan. Berdasarkan data yang dihimpun, seluruh arsitektur logika tersebut mampu memberikan respons yang tepat terhadap setiap skenario uji, seperti yang dipaparkan secara mendetail pada Tabel 3.

4.4. Uji Validitas Instrumen GEQ

Uji validitas menggunakan Korelasi Pearson Product Moment (r tabel = 0,20, N = 50). Seluruh 26 butir instrumen dinyatakan valid dengan nilai r hitung berkisar antara 0,5988 (butir B6) hingga 0,8608 (butir C12). Butir-butir yang mengandung indikator teknis tersembunyi EDDA (B7, B8, B11), FSM (B9, B10,

C6, C7, C8), dan Random Forest (C9, C10) seluruhnya valid, mengindikasikan bahwa pengujian sistem komputasi secara implisit berhasil tersampaikan melalui pengalaman bermain responden.

4.5. Uji Reliabilitas Instrumen GEQ

Tabel 4. Hasil Uji Reliabilitas per Dimensi

Dimensi	Butir	Cronbach's α	Keterangan
Immersion	3	0,8580	Reliabel
Presence	3	0,7248	Reliabel
Flow	5	0,8799	Reliabel
Absorption	3	0,8908	Reliabel
Kausalitas	5	0,9287	Sangat Reliabel
Mekanik Kebijakan	3	0,8539	Reliabel
Evaluasi Kepemimpinan	4	0,9023	Sangat Reliabel

Seluruh tujuh dimensi memiliki nilai Cronbach's Alpha di atas 0,70 sehingga dinyatakan reliabel. Dimensi Kausalitas memperoleh nilai tertinggi (α = 0,9287), mengonfirmasi konsistensi internal instrumen yang sangat baik.

Evaluasi keandalan instrumen *Game Engagement Questionnaire* (GEQ) modifikasi dilakukan melalui uji reliabilitas menggunakan metode Cronbach's Alpha terhadap data dari N = 50 responden. Hasil kalkulasi menunjukkan bahwa seluruh tujuh dimensi pengukuran dinyatakan reliabel karena secara konsisten melampaui ambang batas standar statistik (> 0,70). Pada klaster keterlibatan standar, instrumen mencatatkan nilai yang stabil pada dimensi *Immersion* (α = 0,8580), *Presence* (α = 0,7248), *Flow* (α = 0,8799), dan *Absorption* (α = 0,8908). Kestabilan ini disempurnakan oleh klaster dimensi kustom edukasi dan mekanik, di mana dimensi Mekanik Kebijakan (FSM) meraih α = 0,8539, Evaluasi Kepemimpinan (RF) meraih α = 0,9023, serta dimensi Kausalitas Kebijakan yang mencatatkan skor konsistensi tertinggi sebesar α = 0,9287. Tingginya capaian seluruh koefisien ini membuktikan secara empiris bahwa penyamaran fungsi teknis komputasi ke dalam narasi pengalaman bermain berhasil dipahami secara seragam oleh responden, sehingga data kuantitatif yang dihasilkan terbukti ajeg, bebas dari bias kesalahan acak, dan sangat sah untuk digunakan dalam analisis hasil akhir penelitian

4.6. Statistik Deskriptif & Pembahasan per Sistem

Rata-rata keseluruhan instrumen sebesar 4,087 dari skala maksimum 5,00 (kategori "Baik"). Berikut pembahasan per sistem komputasi yang diuji secara implisit melalui GEQ:

Tabel 5. Rekapitulasi Mean per Dimensi GEQ

Dimensi	Mean	Interpretasi
Immersion	4,120	Baik
Presence	4,147	Baik (Tertinggi GEQ)
Flow (EDDA & FSM)	4,044	Baik
Absorption	3,813	Baik
Kausalitas	4,156	Baik
Mekanik Kebijakan (FSM)	4,180	Baik (Tertinggi Edukasi)
Evaluasi Kepemimpinan (RF)	4,120	Baik

Tabel 5 menunjukkan seluruh dimensi berada pada kategori "Baik" dengan mean ≥ 3,81.

a. EDDA (B7, B8, B11): Rata-rata ketiga butir sebesar 4,07. Sistem EDDA berhasil menyesuaikan kurva kesulitan secara adaptif; pemain merasakan tantangan yang seimbang dan konsisten. Butir B8 ("Saat kesulitan, tantangan menjadi lebih manageable") memperoleh mean tertinggi (4,12), mengonfirmasi bahwa mekanisme Relief Event FSM bekerja efektif.
b. FSM (B9, B10, C6, C7, C8): Butir B9 ("Kemunculan krisis terasa wajar") memperoleh mean 3,90 — nilai terendah di antara butir Flow — mengindikasikan masih ada ruang penyempurnaan pada parameter timing FSM krisis. Namun, butir C6–C8 yang mengukur dampak kognitif dari cooldown kebijakan memperoleh rata-rata 4,18, mengonfirmasi bahwa mekanisme FSM berhasil mendorong pemain berpikir lebih strategis.
c. Random Forest (C9, C10): Rata-rata 4,14 (kategori "Baik"). Nilai r hitung C10 yang tinggi (0,8142) menunjukkan bahwa konsistensi penjelasan profil kepemimpinan sangat berkorelasi dengan kepuasan pengalaman bermain secara keseluruhan, membuktikan bahwa sistem Majority Voting LINQ berhasil mengklasifikasikan gaya kepemimpinan dengan akurasi yang dirasakan memadai oleh responden.

4.7. Mitigasi Kompleksitas Kode (*Spaghetti Code*) Melalui State Pattern

Permasalahan utama dalam pengembangan game simulasi tata kota konvensional adalah tingginya kompleksitas manajemen kondisi (state management) yang kerap menghasilkan spaghetti code akibat pendekatan percabangan kondisional (if-else atau switch-case) yang masif. Melalui implementasi FSM berbasis State Pattern menggunakan bahasa C# pada Godot Engine 4.x, logika tata kelola kota berhasil didekonstruksi menjadi modul status yang terisolasi secara mandiri. Keberhasilan seluruh skenario dalam Black Box Testing membuktikan sifat deterministik dari arsitektur ini. Ketika pemain mengeksekusi aksi konstruksi atau pembongkaran, manipulasi variabel telemetri kependudukan hanya terjadi pada konteks status yang sedang aktif melalui metode Enter(), Execute(), dan Exit(). Pemisahan ini secara signifikan

meningkatkan skala pengembangan (scalability) dan keterpeliharaan (maintainability) sistem game yang memiliki variabel multivariat kompleks, sejalan dengan prinsip arsitektur modular yang diajukan oleh Estevam et al. (2023).

4.8. Korelasi Arsitektur FSM Terhadap Keterlibatan dan Retensi Pemain

Keunggulan teknis dari pemisahan status modular ini terbukti memberikan dampak positif yang selaras pada pengujian pengalaman pengguna (User Experience). Berdasarkan data statistik deskriptif GEQ, dimensi Mekanik Kebijakan yang dikontrol penuh oleh FSM sekuensial (Locked – Available - Active - Cooldown) berhasil menorehkan nilai rata-rata (Mean) tertinggi pada klaster edukasi, yaitu sebesar 4,180. Secara mekanik, keberadaan status Cooldown bertindak sebagai penjamin stabilitas permainan. Jeda waktu tunggu yang dipaksakan secara atomik oleh sistem terbukti secara implisit mencegah pemain melakukan eksploitasi tombol kebijakan (button spamming). Pemain dipaksa secara kognitif untuk melakukan kontemplasi taktis, menganalisis kausalitas anggaran, dan memikirkan dampak jangka panjang terhadap stabilitas kota sebelum mengambil keputusan berikutnya. Proses kontemplasi inilah yang memicu tingginya nilai dimensi Kausalitas Kebijakan (4,156). Angka ini membuktikan bahwa simulasi interaktif berbasis FSM efektif mentransformasi pemahaman teori tata negara yang awalnya kaku pada metode konvensional menjadi sebuah laboratorium eksperimen virtual yang dinamis bagi generasi muda.

4.9. Evaluasi Kritis Kestabilan Sistem Kesulitan Adaptif (EDDA & FSM)

Integrasi antara algoritma EDDA sebagai pengambil keputusan tingkat kesulitan dan FSM sebagai eksekutor peristiwa adaptif berhasil menjaga kenyamanan bermain responden. Hal ini ditunjukkan oleh dimensi Flow yang meraih skor 4,044. Sistem FSM terbukti responsif dalam mengeksekusi perubahan Difficulty Modifier melalui fungsi Mathf.MoveToward secara halus saat membaca fluktuasi skor kemahiran pemain (Player Skill Score). Mekanisme status DisasterActive yang bertindak sebagai safety net juga teruji empiris melalui butir B8 (skor 4,12), di mana sistem berhasil mematikan timer krisis baru ketika pemain berada dalam kondisi kritis, sehingga menghindari akumulasi bencana yang tidak adil bagi pemain. Meskipun demikian, analisis kritis tetap harus diberikan pada butir B9 ("Kemunculan krisis/bencana terasa wajar") yang mencatatkan nilai terendah sebesar 3,90. Hasil ini merefleksikan adanya kelemahan pada kalibrasi parameter waktu (timing parameters) di dalam status Monitoring FSM. Ketika performa pemain diklasifikasikan pada tingkat Ahli (PSS > 80), pengali kesulitan 2,0x mempercepat laju penurunan metrik lingkungan secara terlalu agresif. Akibatnya, transisi menuju status DisasterActive dipicu dalam interval yang terlalu rapat, menimbulkan lompatan kesulitan yang mendadak bagi beberapa responden. Temuan kritis pada butir B9 ini mengonfirmasi bahwa meskipun arsitektur FSM sukses menjaga keterbacaan kode pemrograman dari spaghetti code, penentuan angka konstan pada batas ambang (threshold) transisi di dalam script C# masih memerlukan optimasi lebih lanjut. Solusi untuk mengatasi kelemahan ini adalah dengan menerapkan struktur Hierarchical Finite State Machine (HFSM) pada pengembangan selanjutnya, sehingga status krisis makro dapat dipecah menjadi beberapa tingkatan sub-status yang lebih toleran terhadap dinamika performa pemain.

4.10. Komparasi Teknis Pengelolaan Kondisi Game

Tabel 5. Rekapitulasi Mean per Dimensi GEQ

Parameter Analisis Teknis	Pendekatan Konvensional (Non-FSM)	Arsitektur FSM (*State Pattern* C#)
Arsitektur Kode	Monolitik tunggal dengan keterikatan erat (*tight coupling*).	Modular terpisah dengan keterikatan longgar (*loose coupling*).
Struktur Logika	Percabangan kondisional *if-else/switch-case* bertumpuk.	Kelas objek status mandiri berdasarkan pewarisan kelas abstrak.
Tingkat Keterbacaan	Rendah, alur logika kusut dan berisiko memicu *spaghetti code*.	Tinggi, alur eksekusi bersih karena logika tiap kondisi diisolasi penuh
Skalabilitas Fitur	Sulit, penyisipan fitur baru rentan merusak kestabilan fungsi lama.	Sangat mudah, cukup menambahkan kelas konkret baru tanpa membongkar kode lama.
Risiko Eksploitasi Perintah	Tinggi, pengelolaan jeda waktu mengandalkan penanda variabel boolean manual.	Sangat rendah, dicegah secara implisit oleh status *Cooldown* secara atomik.
Beban Komputasi & Stabilitas	Menengah hingga tinggi, memeriksa seluruh parameter kota di setiap *frame tick*.	Sangat ringan, mampu mematikan fungsi *timer* AI serta memiliki *safety net* krisis.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang dan mengimplementasikan arsitektur Finite State Machine (FSM) berbasis State Pattern pada serious game "The Architect: Rebirth of Seraya" untuk mengelola tiga sistem utama: Sistem Kebijakan, EDDA, dan Sistem Skor. Black Box Testing membuktikan seluruh transisi state berjalan deterministik dan valid. Pengujian GEQ terhadap 50 responden menghasilkan rata-rata keseluruhan 4,087 (kategori "Baik") dengan seluruh

26 butir valid dan 7 dimensi reliabel ($\alpha > 0,70$), mengonfirmasi bahwa implementasi FSM berhasil mewujudkan sistem game yang modular, bebas spaghetti code, dan efektif sebagai media edukasi kausalitas kebijakan publik berbasis game-based learning. Untuk pengembangan selanjutnya, disarankan eksplorasi Hierarchical FSM (HFSM) untuk manajemen state yang lebih dalam, peningkatan parameter timing FSM krisis berdasarkan temuan butir B9, serta porting ke platform mobile untuk memperluas jangkauan edukasi.

DAFTAR PUSTAKA

[1] Chen, X., & He, Y. (2024). Exploring the role and mechanisms of environmental serious games in promoting pro-environmental decision-making. Frontiers in Psychology, Vol. 15, No. 1357283, 2024. https://doi.org/10.3389/fpsyg.2024.1357283

[2] Brauner, P., et al. (2025). More than chair circles? A qualitative systematic review of serious games related to water governance. Water International, Vol. 50, No. 3, pp. 1–25, 2025. https://doi.org/10.1080/02508060.2025.2501511

[3] Cañete Sanz, L., et al. (2025). The Case of Cities: Skylines Versions Affordances in Urban Planning Education. Media and Communication, Vol. 13, No. 1, pp. 1–15, 2025. https://doi.org/10.17645/mac.8747

[4] Estevam, V. B., et al. (2023). State Machine-Based Architecture for Scalable Game Systems. Proceedings of SBGames 2023 – Brazilian Symposium on Computer Games and Digital Entertainment, pp. 1–10, 2023.

[5] Sintaro, S., et al. (2023). Implementation and Comparison in Using State Pattern on Main Character Movement. Barekeng: Jurnal Ilmu Matematika dan Terapan, Vol. 17, No. 2, pp. 0955–0968, 2023. https://doi.org/10.30598/barekengvol17iss2pp0955-0968

[6] Mortazavi, M., & Moradi, H. (2024). A Systematic Literature Review on Dynamic Difficulty Adjustment in Video Games. Entertainment Computing, Vol. 49, No. 100630, 2024.

[7] Brockmyer, J. H., et al. (2009). The development of the Game Engagement Questionnaire. Journal of Experimental Social Psychology, 45(4), 624–634.

[8] Alchimowicz, B., & Plechawska-Wójcik, M. (2024). Comparative analysis of implementation performance using selected scripting languages in the Godot game engine. Journal of Computer Sciences Institute, Vol. 31, pp. 1–8, 2024. https://doi.org/10.35784/jotes.5342

[9] Winardy, et al. (2023). Game Development Life Cycle: A Framework for Interactive Multimedia Production. Jurnal Teknik Informatika, Vol. 16, No. 3, pp. 1–10, 2023.

[10] Mi, X., & Gao, Y. (2025). Player Engagement Metrics in Adaptive Game Systems. IEEE Transactions on Games, Vol. 17, No. 1, pp. 1–12, 2025.

[11] Ghodsvali, M., et al. (2022). An online serious game for decision-making on food-water-energy nexus policy. Sustainable Cities and Society, Vol. 85, No. 104061, 2022.

[12] Dewanata, I., & Willis, R. (2025). Design and Construction of the Action Role-Playing Game: Supply Odyssey Using the GDLC. Journal of Games, Game Art and Gamification, Vol. 7, No. 2, 2025. https://doi.org/10.46336/jgeet.v9i1.532