Jurnal Informatika Universitas Pamulang **ISSN: 2541-1004**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2622-4615**
Vol. 6, No. 3, September 2021 (512-520) **10.32493/informatika.v6i3.10805**

Rancang Bangun Komponen Sandbox pada Game Unity dengan Menerapkan Design Patterns dan Arsitektur Microservices

Andhik Ampuh Yunanto[1], Fadilah Fahrul Hardiansyah[2], Adhiemas Andira Anantha Putra[3], Maulidan Bagus Afridian Rasyid[4], Ramadhany Candra Arif Putra[5]

[1,2,3]Departemen Teknik Informatika dan Komputer, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia, 60111
e-mail: [1]andhik@pens.ac.id, [2]fahrul@pens.ac.id, [3]dimasandiraa@it.student.pens.ac.id

[4,5]PT Maulidan Teknologi Kreatif, Klampis Ngasem, Sukolilo, Surabaya, 60117, Indonesia
e-mail: [4]contact@maulidangames.com, [5]contact@maulidangames.com

Submitted Date: May 31st, 2021 Reviewed Date: July 25th, 2021
Revised Date: August 20th, 2021 Accepted Date: October 12th, 2021

Abstract

Currently the game has covered a very broad market needs where games offer video content, products, Virtual Reality, and special events and game competitions. This offer is what triggers an increase in sales and an increase in revenue in the gaming industry. Due to increasingly fierce competition, game development companies must develop their products quickly and precisely in order to compete in today's game industry. However, rapid product development and the application of proper architecture are also problems, especially the products being developed are games with simulation or sandbox themes that have a high level of complexity. So we need an application that helps game development, especially the development of the sandbox component. The purpose of this study is to provide a library that helps the development of the sandbox component, where this library will apply the Design Patterns and Microservices Architecture to simplify the maintenance and development period of the game after release. This research is also working with the company for the development and validation. This research will develop several sandbox components and will apply a Design Pattern to each component. This proposed method has been tested by users who are experts in developing games. The results show that the application of the proposed method can be implemented and can run well. And the system built also gets a score with an average above 80% and good feedback from expert users. So that this research is expected to become an application that can help programmers later in developing a game application that is good and appropriate. As well as the applications built in this study can also provide recommendations in terms of development methods.

Keywords: Unity Game Engine; Sandbox Component; Microservices Architecture; Design Patterns

Abstrak

Saat ini game telah mencakup kebutuhan pasar yang sangat luas di kalangan masyarakat. Hal ini dikarenakan game menawarkan konten video, produk, Virtual Reality, dan event khusus serta kompetisi game yang membuat menarik perhatian publik. Penawaran inilah yang memicu peningkatan penjualan dan peningkatan pendapatan di industri game. Karena persaingan yang semakin ketat, perusahaan pengembang game harus mengembangkan produknya dengan cepat dan tepat agar dapat bersaing di industri game saat ini. Namun perkembangan produk yang pesat dan penerapan arsitektur yang tepat juga menjadi kendala, terlebih produk yang dikembangkan adalah game dengan tema simulasi atau sandbox yang memiliki tingkat kompleksitas yang tinggi. Sehingga diperlukan sebuah aplikasi yang membantu pengembangan game khususnya pengembangan komponen sandbox. Tujuan dari penelitian ini adalah untuk menyediakan suatu library yang membantu pengembangan komponen sandbox. Library yang dibangun ini akan menerapkan Design Patterns and Microservices Architecture untuk menyederhanakan masa pemeliharaan dan pengembangan game setelah rilis. Penelitian ini juga bekerja sama dengan perusahaan untuk

pengembangan dan validasi. Pada penelitian ini akan dikembangkan beberapa komponen sandbox dan akan menerapkan Design Pattern pada masing-masing komponen. Metode yang diusulkan ini telah diuji oleh pengguna yang ahli dalam mengembangkan game. Hasil penelitian menunjukkan bahwa penerapan metode yang diusulkan dapat bekerja dengan baik dan lancar. Dan sistem yang dibangun juga mendapat skor dengan rata-rata di atas 80% dan umpan balik yang baik dari pengguna ahli. Sehingga penelitian ini diharapkan dapat menjadi sebuah aplikasi yang dapat membantu programmer nantinya dalam mengembangkan aplikasi game yang baik dan sesuai. Serta aplikasi yang dibangun pada penelitian ini juga dapat memberikan rekomendasi dalam hal metode pembuatan game.

Kata Kunci: Unity Game Engine; Komponen Sandbox; Arsitektur Microservices; Design Patterns

1. Pendahuluan

Game dan pemrograman menjauhkan orang-orang dari memahami game secara tradisional dan mengerahkan untuk mengeksplorasi dampak dari mendefinisikan-ulang game pada masyarakat. Game memungkinkan untuk menjauhkan masalah-masalah dunia nyata dan mengarahkan pemain ke berbagai prioritas dan perubahan perilaku dalam kaitannya kehidupan sehari-hari. Saat ini game telah mencakup kebutuhan pasar yang sangat luas dimana game menawarkan konten video, produk, Virtual Reality, event-event spesial serta perlombaan game. Penawaran inilah yang memicu terjadinya peningkatan penjualan dan peningkatan pendapatan. Pendapatan di dalam industri game juga termasuk pendapatan dalam penjualan souvenir-souvenir pelengkap game dan pelayanan yang diberikan di dalam game, dimana hal tersebut menjadi potensi untuk menguatkan model bisnis dalam industri ini (Baltezarevic & Baltezarevic, 2018).

Kompleksitas yang ditawarkan dalam sebuah game juga terus berkembang. Dari game generasi lama seperti SpaceWar! atau Pong, hingga game modern seperi game sandbox dan simulasi yang populer sampai saat ini yaitu Minecraft. Bahkan hingga saat ini, game memiliki banyak sekali macam permainan seperti game edukasi (Yunanto, Herumurti, Rochimah, & Kuswardayan, 2019), game survival (Kuswadayan, et al., 2019), game aksi (Herumurti, Yunanto, Senna, Kuswardayan, & Arifiani, 2020), dan lain sebagainya. Tidak hanya itu, teknologi yang digunakan juga semakin modern seperti menerapkan kecerdasan buatan dan realitas virtual (Yunanto, Herumurti, Kuswadayan, & Rochimah, 2018).

Dalam beberapa tahun terakhir, beberapa game telah menghadirkan sistem permainan yang non-linear atau dikenal dengan sandbox. Sandbox adalah sebuah konsep gaya bermain yang bebas atau kurang diarahkan (breslin, n.d.). Pemain ditawarkan dengan dunia yang luas, terbuka, penuh kehidupan, serta memiliki kebebasan yang tinggi untuk melakukan sesuai keinginan mereka di dalam permainan. Dengan sistem permainan yang selalu berkembang dan semakin kompleks, maka tahap pengembangan game juga akan menjadi semakin kompleks. Selain itu dikarenakan persaingan pasar, banyak perusahaan yang harus membuat perubahan pada sistem mereka secepat mungkin dan tidak mempengaruhi sistem mereka yang sudah ada. Hal ini memerlukan desain, arsitektur dan proses pengembangan yang tepat. Oleh karena itu diperlukan dekomposisi pada sistem menjadi sistem yang lebih kecil untuk mendapatkan modularitas dan perawatan yang lebih baik.

Sebagai tambahan, terdapat penelitian game yang berjudul "Applying microservice architecture pattern to a design of an MMORPG backend" pada tahun 2017. Tujuan dari penelitian ini adalah untuk mengetahui bagaimana mendesain backend pada MMORPG menggunakan arsitektur microservice. Dalam konteks ini backend pada MMORPG merujuk pada sisi server dalam sistem game yang merupakan tempat ribuan orang tersambung dan bermain di dalam dunia virtual (Vähä, 2017). Penelitian "The Effect of GoF Design Patterns on Stability: A Case Study" bertujuan untuk mengetahui efek pada Gangs of Four Design Patterns dalam stabilitas class. Dalam penelitian ini, mereka menggunakan berbagai macam open source studi kasus pada Java untuk mengetahui probabilitas class yang berubah dikarenakan terjadinya perubahan pada class lain (Ampatzoglou, Chatzigeorgiou, Charalampidou, & Avgeriou, 2015).

Design Patterns adalah sistem yang dapat digunakan untuk menyelesaikan permasalahan yang sifatnya sangat mirip yang muncul saat mengembangkan game. Sama seperti struktur data STL, koleksi yang reusable yang dapat digunakan saat diperlukan untuk menyelesaikan masalah tertentu, design pattern dapat digunakan untuk memecahkan masalah logical dalam sebuah kode. Penggunaan kembali kode dapat mengurangi

jumlah baris kode yang digunakan dalam game, yang mengarah pada kode yang lebih stabil dan mudah dikelola, yang keduanya berarti dapat mengembangkan game yang lebih baik dan lebih cepat. Menggunakan Factory Pattern adalah cara yang berguna untuk mengabstraksi pembuatan objek dinamis secara langsung. Sebuah Factory untuk keperluan berikut hanyalah sebuah fungsi yang mengambil jenis objek sebagai parameter dan mengembalikan pointer ke instance objek baru. Objek yang memanggil Factory Method memiliki tanggung jawab pada objek yang dikembalikan oleh Factory Method untuk memastikan bahwa objek tersebut dihapus dengan tepat. Pada Gambar 10, menunjukkan Factory Method yang telah dibuat untuk instasiasi berbagai jenis objek pilihan yang digunakan dalam game Text Adventure (Sutherland, 2014).

2. Kontribusi Penelitian

2.1. Pengembangan Sandbox Library

Belum adanya framework maupun library khusus yang membantu pengembangan komponen-komponen sandbox pada game simulasi dapat menyebabkan beban kerja yang menumpuk pada karyawan perusahaan, sehingga dibutuhkan sebuah library khusus untuk membantu pengembangan game dengan tema simulasi maupun sandbox.

2.2. Implementasi Arsitektur Microservices dan Design Patterns

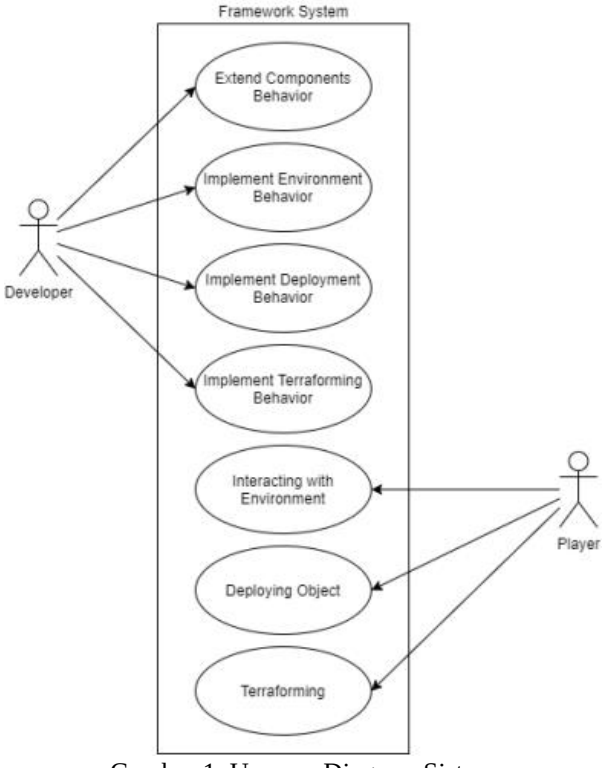
Selain mempermudah pengembang game untuk mengembangkan game dengan tema simulasi dan sandbox, framework atau library yang dibutuhkan juga harus memiliki arsitektur yang tepat sehingga memudahkan perawatan sistem di kemudian hari. Oleh karena itu, framework atau library yang dibutuhkan harus menerapkan Design Pattern dan arsitektur microservice agar mudah dilakukan perawatan dan memiliki fleksibilitas yang tinggi.

3. Metode

Tujuan dari penelitian ini adalah memberikan sebuah library yang akan membantu dalam pengembangan komponen sandbox dalam game. Library ini juga akan mengimplementasikan design pattern dan arsitektur microservices, sehingga produk yang dihasilkan dari library ini diharapkan mudah untuk dirawat. Kualitas dari produk yang dihasilkan dengan library ini juga diharapkan mencapai kualitas Product Revision didalam McCall Quality Model.

Pada Gambar 1 menjelaskan mengenai Use Case Diagram. Pada use case diagram terdapat dua aktor yaitu Developer dan Player yang dapat melakukan hal-hal sebagai berikut:

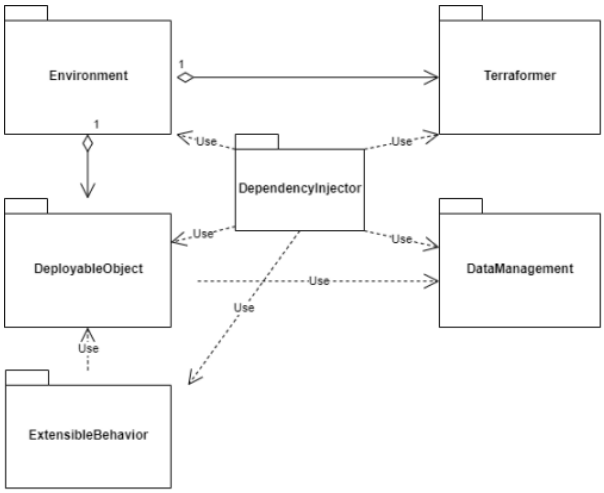
- Extends Behavior, artinya developer dapat menambahkan behavior komponen sandbox dengan cara ekstensi
- Implement Environment Behavior, artinya developer dapat menggunakan komponen environment pada sandbox yang terdapat di dalam library
- Implement Deployment Behavior, Developer dapat menggunakan komponen deployment pada sandbox yang terdapat di dalam library
- Implement Terraforming Behavior, Developer dapat menggunakan komponen terraforming pada sandbox yang terdapat di dalam library
- Interacting with Environment, Player dapat berinteraksi dengan komponen environment
- Deploying Object, Player dapat memasang objek kedalam environment
- Terraforming, artinya Player dapat melakukan terraforming atau merubah kondisi environment



Gambar 1. Usecase Diagram Sistem

Gambar 2 adalah ilustrasi desain arsitektur yang diterapkan di dalam library yang akan dibuat. Pada library, terdapat 6 jenis komponen yaitu *Environment, Terraformer, DeployableObject,*

DataManagement, *ExtensibleBehavior*, dan *DependencyInjector*.



Gambar 2. Desain Arsitektur

Masing-masing komponen memiliki hubungan satu sama lain supaya fungsionalitas sistem bisa berjalan dengan baik. Hubungan antara *Environment* dan *Terraformer* adalah Agregasi, begitu juga dengan *Environment* dan *DeployableObject*. Komponen *Environment* memiliki komponen *Terraformer* dan *DeployableObject*. Selanjutnya hubungan *DeployableObject* dan *DataManagement* adalah Dependency. Hubungan yang sama untuk *DeployableObject* dan *ExtensibleBehavior* dengan relasi agregasi. Hubungan *DependencyInjector* dengan package yang lain adalah Dependency. Hubungan Dependency hanya menggunakan objek class lain tanpa membuatnya di dalam class tersebut dan bukan merupakan variabel di dalam class tersebut. Hubungan-hubungan komponen ini harus sesuai agar efisiensi juga tetap terjaga.

Environment adalah komponen yang berhubungan dengan sistem lingkungan di dalam game, contohnya seperti rotasi kamera, zoom, klik pada objek dan lain sebagainya. Untuk rancangan yang ada pada package Environment, diperkirakan akan menggunakan Design Pattern yaitu Factory Method.

Terraformer adalah komponen yang berisi mengenai kemampuan untuk mengubah keadaan lingkungan atau tempat pengembangan, misalnya membuat kolam pada tanah, atau membuat gundukan tanah. Pada package ini dapat diperkirakan akan mengimplementasikan Strategy Pattern.

DeployableObject merupakan komponen yang berisi objek-objek yang dapat dimasukkan ke dalam lingkungan, contohnya hewan-hewan, bangunan, air, dan sebagainya. Pada package ini dapat diperkirakan akan mengimplementasikan Composite Pattern.

DataManagement adalah komponen yang menyimpan data-data yang diperlukan oleh DeployableObject. Pada package ini dapat diperkirakan akan mengimplementasikan Visitor Pattern.

ExtensibleBehavior adalah komponen yang memungkinkan untuk menambahkan kemampuan-kemampuan baru pada DeployableObject. Sebagai contohnya, komponen ini dapat membantu para developer nantinya Ketika ingin menambahkan ftur-firur baru seperti kecerdasan buatan dan lain sebagainya.

DependencyInjector yang merupakan komponen berisi injektor-injektor dependency yang berguna untuk melonggarkan coupling yang mungkin terjadi antar class. Secara singkat coupling ini dapat menyebabkan class menjadi sulit digunakan Kembali. Oleh karena itu, terkait kualitas dan efisiensi, dependency injector adalah komponen yang harus juga diimplementasikan di semua aplikasi.

Table 1. Example of table caption

Model	Akurasi	AUC
Naïve Bayes	88,51%	0,838
ROS, AdaBoost, dan Naïve Bayes	78,30%	0,856
RUS, AdaBoost, dan Naïve Bayes	74,33%	0,804

4. Pengujian dan Pembahasan

Pada penelitian ini dilakukan eksperimen berupa validasi kepada user. Validasi pada penelitian ini dilakukan kepada 2 user yaitu Ramadhany Candra Arif Putra dan Anggakara Hendra Nandana. Ramadhany Candra Arif Putra merupakan salah satu pembimbing dari pihak industri yang ikut memberikan masukan dan mengetahui proses pengembangan pada penelitian ini, sehingga dikategorikan sebagai Innovator. Anggakara Hendra Nandana merupakan salah satu programmer Unity Engine yang akan menggunakan Sandbox Library ini pertama kali, sehingga dikategorikan sebagai Early Adopter.

4.1. Software Quality Metrics

Pada penelitian ini penulis melakukan validasi menggunakan Software Quality Metrics. Software Quality Metrics digunakan untuk mengukur metrik yang ada pada McCall Quality

Model (Arthur, 1985). McCall Quality Model adalah suatu model yang digunakan untuk mengukur sebuah kualitas produk secara nyata dan memiliki sebelas faktor kualitas untuk diukur. Sebelas faktor kualitas tersebut dibagi menjadi 3 tingkatan yakni Product Operation, Product Revision, dan Product Transition (Cavano & McCall, 1978). Penulis pada saat ini masih melakukan validasi pada metrik Correctness dan Reliability. Untuk metrik Correctness dibutuhkan faktor-faktor seperti Completeness, Consistency, serta Traceability. Perhitungan pada Software Quality Metrics dilakukan dengan bantuan Expert Judgement yaitu Ramadhany Candra Arif Putra yang merupakan Innovator pada penelitian ini.

$$\text{Completeness} = \frac{\text{Total fitur yang selesai}}{\text{Total fitur yang didesain}}$$

Tabel 1. Tabel Quality Factor Completeness

	Nilai
Total fitur yang selesai	12
Total fitur yang didesain	12
Completeness	1

Pada Tabel 1 dapat dilihat bahwa pada kualitas faktor Completeness terdapat total fitur yang didesain yaitu 12, serta total fitur yang selesai yaitu 12. Berdasarkan perhitungan, maka didapatkan nilai kualitas faktor Completeness yaitu 1.

$$\text{Consistency} = \frac{\text{Total fitur yang sesuai desain}}{\text{Total fitur yang didesain}}$$

Tabel 2. Tabel Quality Factor Consistency

	Nilai
Total fitur yang sesuai desain	12
Total fitur yang didesain	12
Consistency	1

Pada Tabel 2 dapat dilihat bahwa pada kualitas faktor Consistency terdapat total fitur yang sesuai desain yaitu 12, dan total fitur yang didesain yaitu 12. Berdasarkan perhitungan, maka didapatkan nilai kualitas faktor Consistency yakni 1.

$$\text{Traceability} = \frac{\text{Total fitur yang selesai dan terdokumentasi}}{\text{Total fitur yang didesain}}$$

Tabel 3. Tabel Quality Factor Traceability

	Nilai
Total Fitur yang selesai dan terdokumentasi	6
Total fitur yang didesain	12
Traceability	0,5

Pada Tabel 3 dapat dilihat bahwa pada kualitas faktor Traceability terdapat total fitur yang selesai dan terdokumentasi yaitu 6, dan total fitur yang didesain adalah 12. Berdasarkan persamaan, maka didapatkan nilai kualitas faktor Traceability yaitu 0,5.

Tabel 4. Tabel Quality Metrics Correctness

Quality Factor	**Nilai**
Completeness	1
Consistency	1
Traceability	0,5
Correctness	0,83

Berdasarkan data-data yang didapatkan pada kualitas faktor yang dibutuhkan pada metrik Correctness dan penulis menentukan bahwa bobot dari masing-masing faktor adalah sama besar, maka nilai Correctness didapatkan dari nilai rata-rata kualitas faktor yang menyusun Correctness. Sehingga didapatkan nilai Correctness yaitu 0,83.

Untuk metrik Reliability dibutuhkan faktor-faktor seperti Accuracy, Consistency, Error Tolerance, Modularity, dan Simplicity.

$$\text{Accuracy} = 1 - \frac{\text{Total akurasi pada total fungsi}}{\text{Total fungsi}}$$

Tabel 5. Tabel Quality Factor Accuracy

	Nilai
Total akurasi pada total fungsi	2,275
Total fungsi	42
Accuracy	0,95

Pada Tabel 5 dapat dilihat bahwa kualitas faktor Accuracy terdapat total akurasi pada total fungsi mempunyai nilai 2,275 dan total fungsi yaitu 42. Berdasarkan persamaan diatas maka didapatkan nilai Accuracy yaitu 0,95.

$$\text{Error Tolerance} = \frac{\text{Total fungsi yang toleransi terhadap input luar}}{\text{Total fungsi}}$$

Tabel 6. Tabel Quality Factor Error Tolerance

	Nilai
Total fungsi yang toleransi terhadap input diluar domain	36
Total fungsi	42
Error Tolerance	0,86

Pada Tabel 6 dapat dilihat bahwa kualitas faktor Error Tolerance terdapat total fungsi yang toleransi terhadap input selain domain yaitu 36, dan total fungsi yaitu 42. Sehingga berdasarkan persamaan maka didapatkan nilai Error Tolerance yaitu 0,86.

$$\text{Modularity} = \frac{\text{Total Class yang Loose coupling}}{\text{Total Class}}$$

$$\text{Coupling (C)} = 1 - \frac{1}{d_i + 2 \times c_i + d_o + 2 \times c_o + g_d + 2 \times g_c + w + r}$$

d_i, merupakan jumlah input parameter data
d_o, merupakan jumlah output parameter data
c_i, merupakan jumlah input parameter control
c_o, merupakan jumlah output parameter control
g_d , merupakan jumlah global variabel yang digunakan untuk data
g_c , merupakan jumlah global variabel yang digunakan untuk control
w, merupakan jumlah modul yang dipanggil (fanout)
r, merupakan jumlah modul yang memanggil (fanin)

Persamaan coupling ini juga perlu dihitung untuk mengetahui tingkat dependency antar komponen. Semakin kecil nilai coupling, maka semakin baik juga arsitektur yang dibangun.

Tabel 7. Tabel Quality Factor Modularity

	Nilai
Total Class yang Loose Coupling	14
Total Class	18
Modularity	0,78

Pada Tabel 7 dapat dilihat bahwa kualitas faktor Modularity terdapat total Class yang Loose Coupling yaitu 14, dan total Class yaitu 18. Perhitungan Loose Coupling didapatkan berdasarkan Expert Judgement yang merupakan salah satu Innovator pada Proyek Akhir ini yaitu Ramadhany Candra Arif Putra menggunakan persamaan perhitungan Coupling. Kemudian dari nilai yang didapatkan, suatu Class dinyatakan Loose Coupling apabila mendapatkan nilai kurang dari sama dengan 0,67 dan didapatkanlah terdapat 14 Class yang Loose Coupling. Sehingga berdasarkan persamaan perhitungan Modularity maka didapatkan nilai Modularity yaitu 0,78.

$$\text{Simplicity} = \frac{\text{Total fungsi yang Clean Code}}{\text{Total fungsi}}$$

Tabel 8. Tabel Quality Factor Simplicity

	Nilai
Total fungsi yang clean code	39
Total fungsi	42
Simplicity	0,93

Pada Tabel 8 dapat dilihat bahwa kualitas faktor Simplicity terdapat total fungsi yang clean code yaitu 39 dan total fungsi yaitu 42. Dimana nilai clean code didapatkan berdasarkan Expert Judgement yang merupakan salah satu Innovator pada penelitian ini yaitu Ramadhany Candra Arif Putra. Berdasarkan persamaan diatas maka didapatkan nilai Simplicity yaitu 0,93.

Tabel 9. Tabel Quality Metrics Reliability

Quality Factor	**Nilai**
Accuracy	0,95
Consistency	1
Error Tolerance	0,86
Modularity	0,78
Simplicity	0,93
Reliability	0,9

Berdasarkan data-data yang didapatkan pada kualitas faktor yang dibutuhkan metrik Reliability serta bobot yang ditentukan penulis adalah sama besar, maka nilai Reliability didapatkan dari nilai rata-rata kualitas faktor yang menyusun Reliability. Sehingga didapatkan nilai Reliability yaitu 0,9.

Selanjutnya penulis menentukan nilai minimum untuk menyatakan apakah library yang dikembangkan memenuhi kriteria Quality Metrics Correctness dan Reliability. Penulis menentukan nilai minimum yaitu 0,7 dalam skala 0 hingga 1. Penulis menilai bahwa nilai 0,7 sudah dirasa cukup dan berada diatas batas tengah skala 0 hingga 1 yaitu 0,5. Sehingga nilai Correctness yaitu 0,83 dan nilai Reliability 0,9 yang nilainya lebih tinggi daripada nilai minimum yaitu 0,7 dapat menyatakan bahwa library yang dikembangkan telah memenuhi kriteria Quality Metrics Correctness dan Reliability.

4.2. Observasi

Validasi yang selanjutnya adalah dengan cara observasi pada pengguna dari library ini, yaitu programmer dari Maulidan Games (Maulidan Game, n.d.). Observasi dilakukan dengan cara mengundang kedua user untuk melakukan online conference, kemudian user melakukan share screen dan membuat sebuah proyek kecil mengikuti tutorial penggunaan library yang telah penulis buat. Lalu penulis akan mengamati bagaimana cara user untuk menggunakan library.

Untuk user Ramadhany Candra Arif Putra sebagai Innovator didapatkan hasil observasi sebagai berikut:

1. User tidak ada kendala ketika menginstall persyaratan yang dibutuhkan untuk menggunakan library.
2. User memahami fungsionalitas dasar pada komponen Environment.
3. User terlihat bingung dalam memahami beberapa variabel untuk mengubah sistem tanah.
4. User tidak kebingungan untuk menambahkan object baru yang ingin user spawn pada sistem *random spawn object* yang ada pada Environment.
5. User tidak kebingungan untuk menambahkan object baru yang ingin user deploy ke dalam game pada komponen Deployable Object.
6. User juga memberikan saran bahwa bisa ditambahkan base script untuk komponen Deployable Object sehingga pada bagian Button UI untuk melakukan deploy object tidak memerlukan komponen Unity yaitu GameObject melainkan spesifik pada komponen Deployable Object.
7. User juga menyarankan untuk meminimalisir kesalahan dalam penggunaan Liskov Substitution Principle.
8. User tidak mengalami kendala dalam mencoba menambahkan sistem memindahkan dan menghapus object yang telah di deploy dan mengaplikasikannya pada projek yang sedang dikerjakan.
9. User memberi masukan dalam hal Reliability pada sistem yang sudah ada agar lebih ditingkatkan lagi.
10. User tidak memiliki kendala dalam mengikuti tutorial yang diberikan pada saat mencoba komponen Extensible Behavior.
11. User memberikan saran agar menambahkan class umum dari pathfinding, sehingga nantinya user dapat mengubah algoritma pathfinding sesuai dengan kebutuhan user tetapi tetap memenuhi kebutuhan dari sistem yang ada.
12. User dapat mengimplementasikan sesuai dengan tutorial yang ada pada saat mencoba komponen Terraformer.
13. User menyatakan bahwa komponen Terraformer dapat ditingkatkan lagi dari sisi fungsionalitasnya, tetapi dalam hal Flexibility sudah sesuai dengan harapan user.
14. User juga mengusulkan untuk menambahkan data class yang bisa digunakan untuk mengontrol sistem Terraformer dan memperjelas satuan dari value yang ada pada semua komponen pada library.

Untuk user Anggakara Hendra Nandana sebagai Early Adopter didapatkan hasil observasi sebagai berikut:

1. User tidak ada kendala ketika menginstall persyaratan yang dibutuhkan untuk menggunakan library pada penelitian ini.
2. User dapat dengan mudah memahami tutorial pada komponen Environment.
3. Ketika user ingin mengubah variabel-variabel tertentu pada komponen Environment user masih sedikit kebingungan karena tidak ada penjelasan mengenai variabel tersebut. Tetapi setelah user mencoba mengubah variabel yang diinginkan user dapat memahami tujuan dari variabel tersebut.
4. User dapat dengan mudah menambahkan object yang ingin ditambahkan untuk *random spawn* pada Environment.
5. User sudah dapat mengikuti tutorial saat mencoba komponen Deployable Object.
6. User menemukan kendala pada saat melakukan deploy dengan benda yang lebih besar ukurannya, dan sistem ternyata dapat melakukan deploy walaupun pada tempat tersebut terdapat benda yang lain.
7. User mencoba menambahkan sistem memindahkan dan menghapus object yang berhasil di deploy dan berhasil mengimplementasikan sesuai dengan tutorial yang ada, tetapi pada saat mencoba pada game user agak kebingungan dalam pengoperasian sistem memindahkan dan menghapus object walaupun dalam hal fungsionalitas sudah berjalan dengan baik.
8. User juga memberikan masukan agar menambahkan fungsionalitas *state selection* yaitu menambahkan tampilan object ketika sedang di select oleh user.

9. User melanjutkan dengan mencoba komponen Extensible Behavior dan user berhasil mengimplementasikan sesuai dengan tutorial. Menurut user sistem pathfinding yang ada sudah sesuai, tetapi perlu memeriksa kembali pada bagian pengecekan apakah pada path yang dipilih terdapat benda atau tidak.
10. User menyarankan untuk memberikan interface pada bagian pathfinding sehingga akan lebih mudah melakukan injeksi apabila user ingin menggunakan algoritma pathfinding yang lain.
11. User dapat mengimplementasikan komponen Terraformer sesuai dengan tutorial yang diberikan.
12. User mengalami kendala ketika sudah melakukan terraform ke atas dan user ingin melakukan terraform ke bawah fungsionalitas terraform tidak bisa dijalankan. Sehingga user menyarankan untuk memperbaiki lagi fungsionalitas pada bagian terraform, tetapi untuk basic dari Terraformer sudah baik dan sesuai.

4.3. Wawancara

Validasi yang terakhir adalah dengan melakukan wawancara pada pengguna library ini. Wawancara dilakukan setelah dilakukannya observasi dari masing-masing pengguna. Wawancara dilakukan dengan user yang sama dengan observasi sebelumnya yaitu Ramadhany Candra Arif Putra dan Anggakara Hendra Nandana. Penulis melemparkan beberapa pertanyaan untuk masing-masing pengguna.

1. Apa yang ada dalam pikiran anda saat mencoba library ini ?
2. Apa kendala yang anda miliki ketika mencoba library ini ?
3. Menurut anda bagaimana saya bisa memperbaiki kualitas dari library ini menjadi lebih baik ?

Berikut adalah jawaban dari wawancara bersama Ramadhany Candra Arif Putra.

1. Library ini sudah hampir sesuai dengan tujuan penelitian ini. Karena pada saat mencoba usability dari penelitian ini tentu adanya dokumentasi dan dalam hal ini merupakan video tutorial yang sangat membantu dalam penggunaan library. Lalu berdasarkan fungsionalitas dasar user berpendapat bahwa library ini sudah bisa digunakan, akan tetapi apakah dapat digunakan untuk membuat game yang lebih besar user masih belum bisa menentukan secara keseluruhan, karena user belum mengetahui fungsi-fungsi mana saja yang bisa diedit sehingga user menyarankan untuk membuat dokumentasi berupa bentuk dokumen.
2. Dikarena user merupakan pengguna yang sudah sering menggunakan Unity Game Engine, sehingga user tidak memiliki kendala tertentu dalam menggunakan library ini. Tetapi user tidak bisa menentukan apakah library ini dapat digunakan dengan mudah oleh pengguna yang belum terbiasa menggunakan Unity Game Engine.
3. Menurut user dalam komponen Environment ingin mendapatkan fleksibilitas yang lebih dalam hal menentukan ukuran luas Environment yang dibuat dan kurang jelasnya variabel-variabel yang harus diisikan pada komponen Environment, sehingga lebih baik diperjelas penggunaan variabel-variabel tersebut. User juga menyarankan untuk membuat dokumentasi berupa bentuk dokumen agar pengguna library mengerti fungsi-fungsi mana saja yang bisa dikembangkan lagi dan fungsi-fungsi mana yang tidak boleh diubah. Kemudian user juga menyarankan untuk value dari sebuah variabel lebih baik lebih spesifik lagi bergantung dengan kebutuhan. Lalu pada komponen Deployable Object perlu diperbaiki lagi pada bagian Reliability, dan sudah memenuhi untuk bagian Correctness. Untuk Extensible Behavior perlu dibuatkan base class berupa *Pathfinding* yang mengimplementasi interface *IPathfinding* misalnya, sehingga pengguna dapat menentukan algoritma pathfindingnya sendiri. Untuk Terraformer dalam hal Flexibility sudah baik, tetapi ada beberapa Correctness yang belum jalan sehingga perlu dicek lagi fungsionalitasnya.

Selanjutnya, berikut adalah jawaban dari wawancara bersama Anggakara Hendra Nandana.

1. Keperluan dasar dari library ini sudah sangat membantu programmer. Karena fungsionalitas yang diperlukan dalam mengembangkan game merupakan hal-hal yang berulang dan dengan adanya library ini hal-hal tersebut sangat berguna dan membantu programmer.
2. Tidak adanya dokumentasi berupa dokumen membuat user sedikit kesulitan dalam menentukan komponen Unity apa saja yang

harus dimasukkan kedalam object yang dibuat, seperti collider, mesh atau semacamnya.

Menurut user perlu ditingkatkan lagi dalam hal Flexibility dalam hal kodingan agar user bisa menambahkan behavior-behavior tertentu sesuai dengan kebutuhan user, sehingga apabila library ini bisa memberikan fleksibilitas yang lebih akan menjadi lebih bagus.

5. Kesimpulan

Pada paper ini penulis telah mengembangkan library sandbox yang ditujukan untuk mempermudah pengembangan komponen sandbox game. Walaupun status library yang telah dikembangkan ini belum sepenuhnya selesai, namun library dapat berjalan dengan baik pada sistem. Selain itu, hasil dari validasi membuktikan bahwa library telah memenuhi kualitas faktor Correctness dan Reliability dimana artinya mendapatkan feedback yang positif dari user. Selain itu menurut user, library ini telah memenuhi kebutuhan dasar dari user. Akan tetapi perlu untuk dikembangkan lebih lanjut lagi untuk meningkatkan kualitas dari library ini.

6. Saran

Penelitian selanjutnya adalah mengembangkan sistem ini supaya bisa dapat diimplementasikan secara real di dunia Industri. Selain itu juga meningkatkan kualitas lain seperti *integrity* dan *efficiency*.

Referensi

Ampatzoglou, A., Chatzigeorgiou, A., Charalampidou, S., & Avgeriou, P. (2015). The Effect of GoF Design Patterns on Stability: A Case Study. *IEEE Transactions on Software Engineering, 41*(8), 781 - 802.

Arthur, L. J. (1985). *Measuring Programmer Productivity and Software Quality*. New York, United States: John Wiley & Sons, Inc.

Baltezarevic, R., & Baltezarevic, V. N. (2018). The video gaming industry: From play to revenue. *International Review Journal*, 71-76.

breslin, s. (n.d.). *The History and Theory of Sandbox Gameplay*. (Gamasutra) Retrieved May 2021, from https://www.gamasutra.com/view/feature/132470/the_history_and_theory_of_sandbox_.php?print=1

Cavano, J. P., & McCall, J. A. (1978). A framework for the measurement of software quality. *Proceedings of the software quality assurance workshop on Functional and performance issues*, (pp. 133-139).

Herumurti, D., Yunanto, A. A., Senna, G. A., Kuswardayan, I., & Arifiani, S. (2020). Development of First-Person Shooter Game with Survival Maze Based on Virtual Reality. *6th Information Technology International Seminar (ITIS)*. Surabaya.

Kuswadayan, I., Herumurti, D., Hariadi, R. R., Wildianurahman, M., Yunanto, A. A., & Arifiani, S. (2019). Survival Education for User on Unknown Islands using Simulation Games. *12th International Conference on Information & Communication Technology and System (ICTS)*. Surabaya, Indonesia.

Maulidan Game. (n.d.). *Maulidan Games Profile*. Retrieved July 20, 2021, from https://maulidangames.com/

Sutherland, B. (2014). Useful Design Patterns for Game Development. In *C++ Game Development Primer* (pp. 17-32). Apress, Berkeley, CA.

Vähä, M. (2017). *Applying microservice architecture pattern to a design of an MMORPG backend.* University of Oulu.

Yunanto, A. A., Herumurti, D., Kuswadayan, I., & Rochimah, S. (2018). Intelligent System for Agent in Educational Game Using Dynamic Gram Similarity. *2018 International Seminar on Application for Technology of Information and Communication*. Semarang.

Yunanto, A. A., Herumurti, D., Rochimah, S., & Kuswardayan, I. (2019). English Education Game using Non-Player Character Based on Natural Language Processing. *The Fifth Information Systems International Conference*. Surabaya, Indonesia.