Penerapan *Defect Management Process* Dalam Pengujian *Mobile Game* (Studi Kasus: Dash Soccer: Premier League)

Farhan Ramadhan
Jurusan Informatika, Fakultas
Teknologi Industri
Universitas Islam Indonesia
Yogyakarta
18523104@students.uii.ac.id

Rahadian Kurniawan
Jurusan Informatika, Fakultas
Teknologi Industri
Universitas Islam Indonesia
Yogyakarta
rahadiankurniawan@uii.ac.id

Abstract*—PT Git Solution merupakan salah satu badan usaha milik yayasan AMIKOM Yogyakarta yang bergerak pada sektor jasa perancangan, pembangunan dan pengembangan sistem informasi berskala nasional. Selama ini, PT Git Solution belum menerapkan defect management secara maksimal dan belum menggunakan defect tracking tool khusus dalam pengujian game yang dikembangkan, sehingga bug report yang dihasilkan belum memuat informasi lengkap mengenai bug dan kekurangan komponen pengkategorian dan status penangganan. PT Git Solution tengah mengembangkan mobile game berjudul Dash Soccer: Premier League. Pada tahap pengujian proyek ini, PT Git Solution ingin bug report yang dihasilkan mengandung informasi-informasi yang mempermudah proses monitoring bug. Oleh karena itu defect management process dan defect life cycle digunakan sebagai pedoman dalam melakukan pengujian, ditambah menggunakan Jira Software untuk memantau dan mengelola bug report. Dengan penerapan defect management process dan defect life cycle dalam tahap pengujian mobile game, terbukti dapat menghasilkan bug report yang mengandung informasi lengkap dan mudah dipantau proses penanganannya.

Keywords— mobile game, PT Git Solution, bug report, defect management process, defect life cycle, Jira Software

I. PENDAHULUAN

Manajemen *bug* merupakan bagian penting dalam tahap pengujian sebuah *software*. Karena dalam pengembangan *software, bug* merupakan suatu hal yang sangat sulit untuk dihindari. Sebaik apapun pengembang membuat suatu *software, bug* akan tetap dapat ditemukan dalam *software* yang dikembangakn. Setiap *bug* yang teridentifikasi harus didokumentasikan untuk memudahkan *software engineer* dalam memahami *bug* dan memperbaikinya. Jumlah *bug* yang teridentifikasi selama pengemba *software* akan meningkat dengan tingkat kompleksitas *software* yang dikembangakn. Oleh karena itu dibutuhkan suatu proses manajemen *bug* untuk mengelola semua bug yang berhasil diidentifikasi.

PT Git Solution merupakan badan usaha milik Yayasan AMIKOM Yogyakarta yang bergerak pada sektor jasa perancangan, pembangunan dan pengembangan sistem informasi berskala nasional. Salah satu layanan yang disediakan perusahaan adalah layanan pembuatan *game* untuk berbagai *platform.* Dalam melakukan pengujian *game*, selama ini PT Git Solution belum menerapkan *defect management process* secara maksimal dan belum memanfaatkan *tool bug tracking* khusus untuk melakukan pencatatan dan pemantauan *bug*. Dalam praktiknya PT Git Solution masih menggunakan Google Docs untuk membuat *bug report* dan *monitoring bug.* Dalam *bug report* yang dibuat hanya mengandung informasi dasar mengenai *bug* seperti: ID *bug*, nama *bug*, deskripsi, *reporter*, *reported date*, *environment* (OS, *Platform*), *attachment* (*Screenshot*, rekaman, URL), *step to reproduce*, *expected result*, dan *actual result.* Kelemahan dari *bug report* yang dihasilkan adalah, kurangnya komponen-komponen seperti: kategori (*severity* dan *priority*), status penanganan, penjadwalan penangganan, dan komponen pelengkap lain. Akibatnya informasi dalam *bug report* kurang lengkap dan menyulitkan untuk dimonitoring proses penanganannya. Akibatnya tim *developer* atau tim yang menangani *bug* kesulitan memahami *bug* yang dilaporkan, selain itu akan menyulitkan bagi *Product Owner* atau tim lain diluar *Quality Assurance* untuk memahami status terkini penanganan *bug*.

PT Git Solution sedang mengembangkan *mobile game* untuk *platform* Android dengan judul Dash Soccer: Premier League untuk menambah koleksi *game* bertema sepak bola yang dimiliki perusahaan. *Game* ini dikembangkan untuk mengikuti tren piala dunia 2022 sekaligus untuk menargetkan *end user* penggemar sepak bola khususnya penggemar Premier League. Dalam tahap pengujiannya PT Git Solution ingin *bug report* yang dihasilkan memuat informasi yang memungkinkan *bug* dapat dipantau dengan mudah, selain itu juga memuat informasi yang lebih spesifik yang dapat membantu divisi *programmer* dalam memperbaiki *bug*. Untuk memenuhi dua hal tersebut, dalam pengujian *game* menerapkan *defect management process* dan *defect life cycle* sebagai pedoman dalam pengelolaan *bug*, selain itu juga memanfaatkan Jira Software sebagai alat untuk membantu memantau dan mengelola *bug* yang ditemukan. Tangkap layar tampilan *gameplay game* Dash Soccer: Premier League dapat dilihat pada gambar 1.

Gambar 1 Tangkap layar tampilan *gameplay game* Dash Soccer: Premier League.

Pada makalah ini akan dijelaskan bagaimana penerapan kombinasi antara *defect management process* dan *defect life cycle* dapat menghasilkan *bug report* yang mengandung

informasi lengkap dan mudah untuk dipantau proses penanganannya. Makalah ini juga akan membahas penggunaan Jira Software untuk mengelola *bug report* atau *bug* yang dilaporkan tim penguji. Dengan adanya makalah ini, diharapkan pengembang memahami manfaat penerapan *defect management process* dan *defect life cycle* dalam masa pengujian aplikasi.

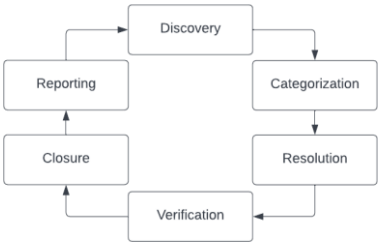
Struktur makalah dalam penelitian terbagi menjadi lima bagian yaitu:

1. Pendahuluan, berisikan latar belakang masalah, latar belakang pengembangan proyek dan latar belakang penerapan defect management process.
2. Landasan teori, berisikan landasan teori yang digunakan dalam pengujian proyek.
3. Metodologi, berisiskan penjelasan mengenai metode yang digunakan dalam pengujian proyek.
4. Hasil dan pembahasan, berisikan hasil pelaksanaan pengujian proyek.
5. Kesimpulan, rangkuman manfaat penerapan hasil penelitian

II. LANDASAN TEORI

A. *Defect Management Process*

Defect management process merupakan *framework* mengenai proses sistematis untuk mengidentifikasi dan memperbaiki *bug* [1]. Penerapan *defect management* yang dilakukan sejak tahap awal pengembanagn perangkat lunak dapat mengurangi waktu, biaya dan sumber daya untuk memperbaiki *bug* [2]. Deteksi *bug* secara dini dapat mencegah migrasi *bug* dari fase kebutuhan ke fase desain atau dari fase desain ke fase implementasi [3]. Siklus *defect management* terdiri dari enam tahapan, yaitu: *Discovery, Categorization, Resolution, Verification, Closure,* dan *Reporting*. Gambar 2 menggambarkan siklus *defect management.*



Gambar 2 Siklus *defect management*.

1. Discovery

Discovery merupakan tahap di mana tim penguji melakukan pengujian untuk menemukan *bug* dalam aplikasi. Berbagai teknik pengujian dapat digunakan untuk menemukan *bug* pada tahap ini, seperti *alpha testing* atau *beta testing*. Setelah *bug* ditemukan, penguji mengumpulkan informasi terperinci terkait *bug* ke dalam *bug report*. Kemudian *bug report* di analisis oleh *Test Manager* sebelum diteruskan ke tim *developer*.

2. Categorization

Categorization merupakan tahap di mana *Test Manager* mengkategorikan bug yang ditemukan pada tahap sebelumnya berdasarkan dua parameter, yaitu: *priority* dan *severity*. *Priority* merupakan tingkat urgensi *bug* untuk segera diperbaiki. *Severity* merupakan tingkat keparahan yang ditimbulkan *bug* terhadap aplikasi.

3. Resolution

Resolution merupakan tahap di mana *bug* yang dilaporkan mulai diperbaiki. Perbaikan *bug* dimulai dengan *Test Manager* menunjuk *developer* untuk memperbaiki *bug*. Kemudian *developer* menetapkan penjadwalan untuk memperbaiki *bug* berdasarkan prioritas *bug*. Setelah *bug* berhasil diperbaiki, *developer* memberikan laporan bahwa *bug* telah diperbaiki.

4. Verification

Verification merupakan tahap di mana tim penguji melakukan pengujian untuk memastikan *bug* yang berhasil diperbaiki tidak terjadi lagi. Teknik yang bisa digunakan dalam tahap ini seperti *regression testing.* Jika *bug* dinyatakan tidak muncul kembali, berlanjut ke tahap *closure.*

5. Closure

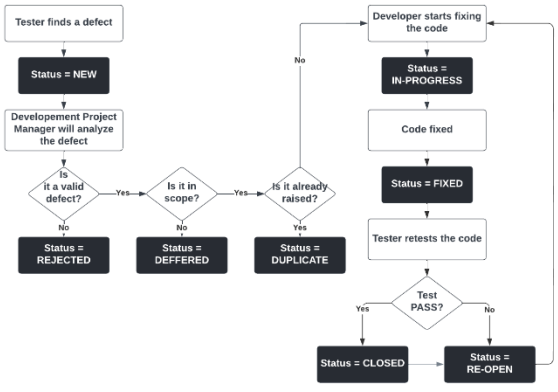
Closure merupakan tahap penutupan laporan *bug*. Jika *bug* tidak terjadi kembali maka laporan *bug* dapat ditutup. Namun jika *bug* masih terjadi, maka penguji atau *Test Manage*r perlu memberi tahu tim *developer* untuk melakukan pengecekan ulang terhadap *bug* yang terkait.

6. Reporting

Reporting merupakan tahap dimana *Test Manager* menyiapkan dan mengirim laporan *bug* ke *Project Manager* atau *Product Owner* untuk mendapat *feedback* mengenai proses penanganan *bug* dan status *bug*. Kemudian *Project Manager* atau *Product Owner* memberikan *feedback* atau memberikan dukungan lebih lanjut mengenai proses penanganan *bug*. Pelaporan *bug* pada tahap ini membantu mengkomunikasikan, melacak, dan menjelaskan *bug* secara rinci.

B. *Defect Life Cycle*

Defect life cycle dalam pengembangan perangkat lunak merupakan sekumpulan status spesifik yang dilalui *bug* sepanjang siklus hidupnya [3]. Tujuan dari penerapan *defect life cycle* adalah untuk mempermudah koordinasi dan komunikasi mengenai status *bug* terkini yang berubah-ubah dengan anggota tim pengembang lainnya dan membuat proses perbaikan *bug* menjadi sistematis dan efisien [4]. *Defect life cycle* dijelaskan pada gambar 3.



Gambar 3 Siklus hidup *bug*.

Siklus *defect life cycle* dimulai ketika penguji menemukan *bug*, *bug* baru tersebut diberi status *"NEW"*. Selanjutnya laporan *bug* diteruskan ke *Project Manager* untuk diputuskan apakah *bug* dapat diteruskan ke tim *developer* atau tidak. Jika *bug* yang dilaporkan tidak valid, status *bug* diubah menjadi *"REJECTED"*. Jika *bug* yang dilaporkan berada diluar *scope* pengujian, status *bug* diubah menjadi *"DEFFERED"*. Jika *bug* yang dilaporkan merupakan duplikat dari *bug* yang pernah dilaporkan sebelumnya, maka status *bug* diubah menjadi *"DUPLICATE"*. Namun jika *bug* yang dilaporkan layak untuk diteruskan ke tim *developer* untuk diperbaiki, status *bug* diubah menjadi *"IN-PROGRESS"*. Kemudian jika *bug* yang diterima tim *developer* telah berhasil diperbaiki, status *bug* diubah menjadi *"FIXED"*. Kemudian penguji melakukan pengujian ulang terhadap perbaikan *bug*, jika *bug* dinyatakan tidak terjadi kembali. Maka status *bug* diubah menjadi *"CLOSED"*. Namun jika *bug* masih terjadi, maka status *bug* diubah menjadi *"RE-OPEN"*. Jika *bug* yang sudah berstatus *CLOSED* kembali terjadi, maka status *bug* tersebut diubah menjadi *"RE-OPEN"*. Kemudian laporan *bug* yang memiliki status *RE-OPEN* dikembalikan ke tim *developer* untuk diperbaiki ulang..

C. Bug Report

Bug report dalam pengujian perangkat lunak merupakan dokumen terperinci mengenai *bug* yang ditemukan pada aplikasi perangkat lunak yang diuji. *Bug report* dibuat oleh penguji untuk membantu *developer* atau programer dalam memahami *bug* yang dilaporkan. *Bug report* berisi tentang setiap detail informasi *bug*, seperti deskripsi, nama penguji yang menemukan, *developer* atau programmer yang memperbaikinya, dan langkah-langkah reka ulang [5].

D. Play Testing

Play testing adalah metode pengujian *game* yang menggunakan perspektif *end user* dalam pengujiannya. Tujuan utama *play testing* adalah untuk menguji apakah *game* menyenangkan untuk dimainkan atau apakah ada masalah pada sistem game, seperti: mekanisme game tidak berjalan seperti semestinya, ada area dalam game yang tidak dapat diakses *player*, atau pergerakan karakter *game* kurang luwes. Lebih jauh lagi, pengujian ini ditujukan untuk melakukan penyeimbangan dalam game, perubahan mekanisme aliran (*flow*) *game* agar lebih menyenangkan untuk dimainkan, dan apakah pemain mengerti maksud dari *game* tanpa dijelaskan oleh pengembang secara langsung [6].

E. Ad Hoc Testing

Ad hoc Testing dikenal juga sebagai *expert testing* atau *monkey testing*. *Ad hoc testing* adalah teknik pengujian sistem tanpa menggunakan skenario atau *test case* yang bertujuan untuk menemukan kesalahan atau *defect* dalam perangkat lunak atau aplikasi yang diuji. Teknik pengujian ini dilakukan tanpa perencanaan atau dokumentasi. Penguji atau *tester* bebas melakukan skenario uji apapun yang diinginkan, karena tidak ada batasan skenario uji yang boleh dilakukan oleh penguji[4]. *Defect* atau *bug* yang ditemukan selama pengujian *ad hoc testing* perlu dicatat untuk diserahkan kepada tim *developer* untuk perbaikan[7].

F. Regression Testing

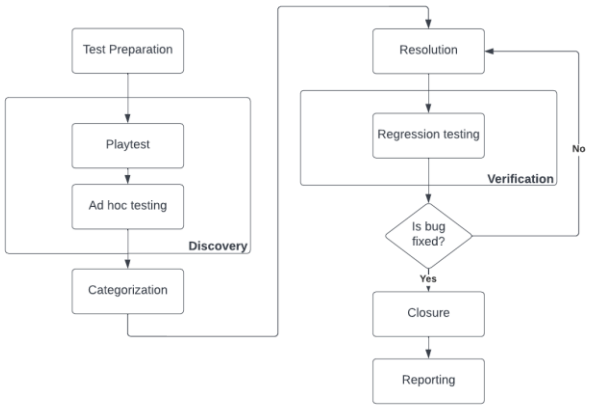
Regression testing adalah jenis pengujian perangkat lunak untuk memastikan bahwa perubahan kode baru tidak berdampak terhadap fitur lain. Pengujian dilakukan terhadap keseluruhan atau sebagian dari *test case* yang telah dilakukan sebelumnya untuk memastikan fungsionalitas yang ada bekerja dengan baik. Pengujian ini dilakukan untuk memastikan bahwa perubahan kode baru tidak menimbulkan efek samping yang tidak terduga pada fungsionalitas lain pada aplikasi yang diuji [8].

G. Penelitian Terdahulu

Survey yang dilakukan terhadap *software* seperti Mozilla, Eclipse, dan Apache menemukan bahwa informasi yang disajikan dalam *bug report* memiliki peran penting dalam perbaikan *bug*. Laporan yang tidak memadai dapat menyebabkan keterlambatan perbaikan *bug* atau bahkan menyebabkan melebihi *deadline* pengembangan proyek [9]. Untuk menghasilkan produk aplikasi atau *software* yang berkualitas tinggi dengan jumlah *bug* rendah, diperlukan penerapan strategi *defect management* yang efektif dalam proses pengembangan *software*. Karena dalam pengembangan tidak hanya membutuhkan pemrograman atau *coding* saja, melainkan penanganan dan manajemen *bug* adalah hal yang penting dalam pengembangan *software* [10]. Pada penelitian ini akan membahas penerapan kombinasi antara *defect management process* danegan *defect life cycle* untuk menghasilkan *bug report* yang memiliki komponen yang mempermudah pemantauan proses penanganan *bug*, selain itu juga memuat informasi yang spesifik yang dapat membantu untuk memperbaiki *bug* yang terkait.

III. METODOLOGI

Tahapan metodologi dimulai dengan *test preparation* sampai dengan *reporting*. Implementasi *defect management* dan *defect life cycle* dimulai setelah tahap *test preparation.* Gambar 4 merupakan metodologi pengujian Dash Soccer: Premier League.



Gambar 4 *Metodologi pengujian* game *Dash Soccer: Premier League.*

A. Test Preparation

Test Preparation merupakan tahap persiapan sebelum dimulainya tahap pengujian *game* Dash Soccer: Premier League. Pada tahap ini tim yang akan melaksanakan pengujian diberikan dokumen desain *game* untuk dipahami dan dianalisis. Pada tahap ini tim penguji akan menetapkan model-model pengujian yang akan digunakan pada masa pengujian berdasarkan analisis terhadap dokumen desain *game.* Selain itu, tim penguji juga akan mempersiapkan *tool-tool* yang akan digunakan selama pengujian. Dokumentasi diskusi analisis desain *game* dapat dilihat pada gambar 5.

Gambar 5 Dokumentasi diskusi analisis desain *game* Dash Soccer: Premier League.

B. *Discovery*

Discovery merupakan tahap pengujian yang melibatkan pihak *internal* (tim penguji) dan *eksternal* (sempel *end user*). Pelaksanaan pengujian bertujuan untuk menemukan *bug* dalam *game* atau untuk mengetahui pengalaman bermain yang ditimbulkan setelah memainkan *game*. *Ad hoc testing* dan *play testing* merupakan model pengujian yang digunakan pada tahap ini. Setiap *bug* yang ditemukan pada tahap ini akan didokumentasikan di Jira Software, kemudian diperiksa oleh *Test Manager* apakah memenuhi 3 persyaratan diteruskan ke tim *developer* yaitu: *bug report* mengandung informasi yang lengkap, *bug* masih dalam *scope* pengujian, dan *bug* bukan duplikat dari *bug* yang sudah ada.

C. *Categorization*

Categorization merupakan tahap pengkategorian *bug* yang ditemukan pada tahap *discovery*. *Bug* dikategorikan dalam 2 parameter yaitu: *priority* dan *severity*. Parameter *priority* dan *severity* akan dicantumkan dalam setiap *bug report* pada *bug* yang terdaftar dalam *database* Jira Software oleh *Test Manager*.

D. *Resolution*

Resolution merupakan tahap perbaikan *bug* yang lolos pemeriksaan oleh *Test Manager* pada tahap discovery. Perbaikan dilakukan oleh tim *developer* atau tim lain yang diberi amanah untuk memperbaiki. Pada tahap ini tim yang bertanggung jawab memperbaiki *bug* akan membuat penjadwalan kapan *bug* selesai diperbaiki, kemudian memberikan laporan bahwa *bug* siap untuk di tes ulang ke *Test Manager* jika *bug* sudah berhasil diperbaiki.

E. *Verification*

Verification merupakan tahap pengujian ulang terhadap *bug-bug* yang dilaporkan sudah diperbaiki. Pada tahap ini menggunakan model pengujian *regression testing* untuk menguji ulang setiap *bug*. Jika *bug* yang diuji pada tahap ini masih terjadi, *bug* tersebut akan dikembalikan lagi ke tim yang memperbaiki untuk dicek kembali.

F. *Closure*

Closure merupakan tahap penutupan *bug* yang sudah terverifikasi berhasil diperbaiki. Penutupan *bug* dilakukan dengan mengubah status *bug* menjadi *CLOSED*. Jika *bug* yang sudah ditutup muncul kembali di masa pengujian berikutnya, status *bug* akan diubah menjadi *RE-OPEN* dan *bug* perlu memasuki tahap *resolution* kembali untuk diperbaiki.

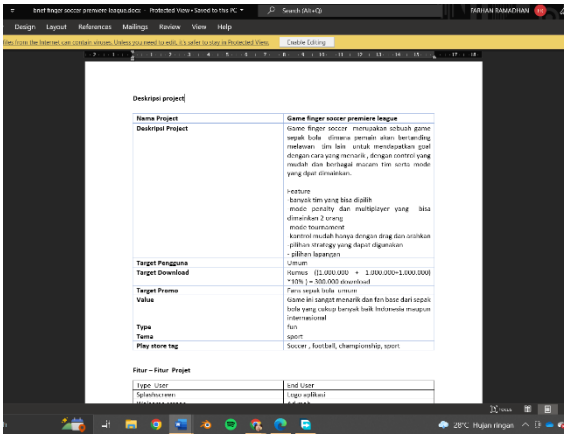
G. *Reporting*

Reporting merupakan tahap pelaporan hasil pengujian yang dilakukan oleh *Test Manager*. Laporan hasil pengujian mencakup keseluruhan penyelenggaraan pengujian yang dilakukan tim *Quality Assurance* selama pengujian. Tujuan dari pelaporan ini adalah untuk mengevaluasi proses pengujian dan mendapat *feedback* dari *Product Owner*, *Project Manager*, atau tim lain dalam pengembnagnan proyek. Selain itu, laporan hasil pengujian digunakan untuk mengukur kelayakan *game* Dash Soccer: Premier League dirilis ke publik.

IV. HASIL DAN PEMBAHASAN

A. *Test Preparation*

Sebelum melaksanakan pengujian, tim penguji diberikan dokumen desain *game* untuk membantu memahami fitur-fitur *game* yang akan diuji dan membantu untuk menentukan model pengujian apa yang akan digunakan. Pada tahap ini diputuskan model pengujian yang akan digunakan dalam pengujian adalah *play testing*, *ad hoc testing* dan *regression testing*. Selain itu, diputuskan akan memanfaatkan Jira Software sebagai alat pelaporan dan monitoring *bug* selama pengujian. Gambar 6 menunjukkan dokumen desain *game* Dash Soccer: Premier League.

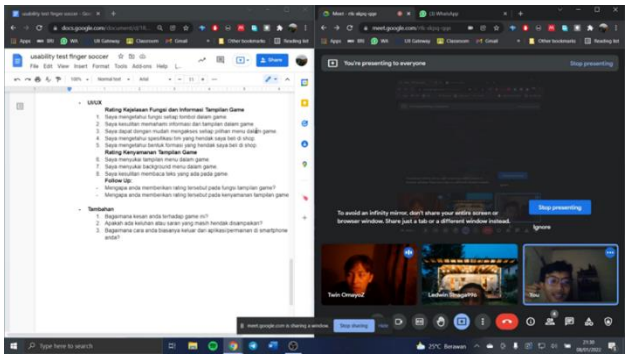


Gambar 6 dokumen desain *game*.

B. *Discovery*

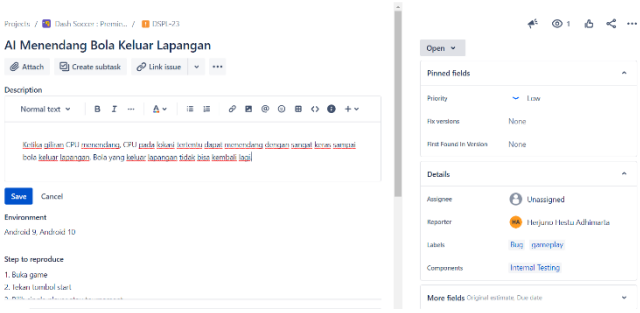
Tahapan *discovery* dimulai setelah desain *game* sudah dipahami, model pengujian sudah ditetapkan dan Jira Software siap digunakan. Model pengujian yang dilakukan pada tahap ini adalah *play testing* dan *ad hoc testing*. Pengujian pada tahap ini melibatkan pihak *internal* dan *eksternal*. Pengujian *play testing* dilakukan melibatkan 6 orang yang menjadi sempel *end user*, sedangkan *ad hoc testing* dilakukan oleh tim *Quality Assurance*.

Pengujian *play testing* dilakukan dengan mengobservasi sempel *end user* memainkan *game* Finger Soccer (template *game* Dash Soccer: Premier League) tanpa ada intervensi berupa arahan atau perintah penguji, kemudian sempel *end user* diwawancarai untuk mengetahui pengalam bermainnya. Hasil dari pengujian ini yaitu: informasi mengenai konten yang ada dalam *game* kurang jelas, konten *game* masih sedikit dan pemain kesulitan melawan musuh AI(artificial intelligence). Hasil dari pengujian digunakan sebagai masukan bagi *Game Designer* untuk memperbaiki desain *game* selama masa pengembangan. *Screenshot wawancara play testing* melalui Google Meet dapat dilihat pada gambar 7.



Gambar 7 Dokumentasi diskusi analisis desain *game* Dash Soccer: Premier League.

Pengujian *ad hoc testing* dilakukan dengan memainkan *game* secara langsung dengan mengandalkan pemahaman terhadap desain *game*, dengan tujuan untuk menemukan *bug*. *Bug* yang teridentifikasi, dianalisis dan informasi mengenai *bug* ditambahkan kedalam *bug report* menggunakan Jira Software dengan status *bug "NEW"*. Gambar 8 menunjukkan tampilan *bug report*, *bug* AI menendang bola keluar lapangan. Tabel 1 adalah tabel *bug* yang ditemukan selama pengujian.



Gambar 8 tampilan *bug report*, *bug* AI menendang bola keluar lapangan.

Tabel 1 *bug* yang ditemukan selama pengujian.

Bug_ID	Nama
DSPL-1	Mode *tournament* tidak bisa dibuka
DSPL-4	*Disable audio* hanya berlaku di *main screen*
DSPL-5	Nama *game* belum sesuai
DSPL-6	Belum ada *tutorial*
DSPL-7	Tim *penalty random*
DSPL-8	Tim masih menggunakan aset lama atau belum liga premier
DSPL-9	*Button exit* digunakan sebagai *button tutorial*
DSPL-10	Admob belum diintegrasikan dalam *game*
DSPL-11	Icon tim di *shop stretched*
DSPL-12	*Toast massage* koin kurang
DSPL-13	*Buy formation* di *shop* masih menggunakan logo Manchester United
DSPL-15	Warna *button next back*, kurang menyala
DSPL-16	Tampilan *gameplay* belum sesuai desain UI/UX
DSPL-17	Menutup *gap* kosong tampilan
DSPL-18	Warna *text* jumlah trofi dan koin terlalu menyatu dengan *background*
DSPL-19	Aset *background tournament bracket* tidak sesuai desain UI/UX
DSPL-20	Aset formasi masih menggunakan aset *game* template
DSPL-21	Banner admob tidak *center* dan menutupi aset lain
DSPL-22	Inkonsistensi *visual stat* tim
DSPL-23	AI menendang bola keluar lapangan

Setelah penguji selesai melakukan pengujian dan berhasil menginput informasi mengenai *bug* yang ditemukan selama pengujian, giliran *Test Manager* untuk menganalisis setiap *bug* yang dilaporkan. Analisis dilakukan dengan memahami *bug report* pada setiap *bug* yang dilaporkan, kemudian memutuskan status *bug* diubah menjadi *REJECTED*, *DIFFERED*, atau *DUPLICATE*. Hasil dari analisis yang dilakukan 3 dari 20 *bug* yang dilaporkan tidak diteruskan ke tahap selanjutnya karena dinilai berstatus *REJECTED* atau *DIFFERED*. Penjelasan mengenai *bug* yang gagal diteruskan ke tahap selanjutnya bisa dilihat pada tabel 2.

Tabel 2 Hasil analisis status *bug*.

Bug_ID	Status	Alasan
DSPL-9	*REJECTED*	Pengembangan fitur *tutorial* akan dilakukan pada proyek selanjutnya atau sampai waktu yang belum ditentukan, mengingat waktu pengembangan proyek Dash Soccer: Premier League sudah hampir habis.
DSPL-6	*REJECTED*	Pengembangan fitur *tutorial* akan dilakukan pada proyek selanjutnya atau sampai waktu yang belum ditentukan, mengingat waktu pengembangan proyek Dash Soccer: Premier League sudah hampir habis.
DSPL-7	*DIFFERED*	Dianggap sebagai *improvement* untuk *game* atau diluar scope pengujian. Fokus utama masih dalam memperbaiki bug yang ada.

C. Categorization

Setelah *bug* pada tahap sebelumnya dianalisis terkait statusnya. Kemudian *Test Manager* menganalisis *bug* untuk dikategorikan berdasarkan *priority* dan *severity*. Hasil analisis tahap *categorization* dapat dilihat pada tabel 3.

Tabel 3 Hasil analisis tahap *categorization*.

Bug_ID	Priority	Severity
DSPL-1	*Highest*	*Highest*

DSPL-4	*Highest*	*High*
DSPL-5	*Highest*	*Low*
DSPL-8	*Highest*	*High*
DSPL-10	*Highest*	*Low*
DSPL-11	*Medium*	*Low*
DSPL-12	*Medium*	*Lowest*
DSPL-13	*Medium*	*Low*
DSPL-15	*Low*	*Lowest*
DSPL-16	*Medium*	*Low*
DSPL-17	*Highest*	*High*
DSPL-18	*High*	*Lowest*
DSPL-19	*High*	*Low*
DSPL-20	*Medium*	*Low*
DSPL-21	*Highest*	*High*
DSPL-22	*Highest*	*Low*
DSPL-23	*Medium*	*Highest*

D. Resolution

Setelah *bug* dianalisis oleh *Test Manager*, selanjutnya *bug-bug* tersebut di *assign* ke tim *developer* atau tim lain yang diberi amanah untuk memperbaiki *bug*. Bug yang sudah ditugaskan ke tim yang bertanggung jawab memperbaiki, statusnya diubah menjadi IN-PROGRESS. Kemudian tim tersebut membuat penjadwalan kapan *bug* harus selesai sebelum masa *regression testing* yang jatuh pada 2 Februari 2022 dan 7 Februari 2022. Status *bug* diubah menjadi *FIXED* ketika *bug* berhasil diperbaiki oleh tim yang menangani *bug* terkait. Penjadwalan perbaikan bug dan status bug dapat dilihat pada tabel 4. Pada tahap ini tim *programmer* belum bisa memperbaiki *bug* DSPL-17 dan DSPL-23, keterangan mengenai *bug* ini dapat dilihat pada tabel 5.

Tabel 4 Penjadwalan dan status bug pada tahap *resolution.*

Bug_ID	Diselesaikan sebelum	Status
DSPL-1	30 Januari 2022	*FIXED*
DSPL-4	30 Januari 2022	*FIXED*
DSPL-5	30 Januari 2022	*FIXED*
DSPL-8	30 Januari 2022	*FIXED*
DSPL-10	31 Januari 2022	*FIXED*
DSPL-11	1 Februari 2022	*FIXED*
DSPL-12	1 Februari 2022	*FIXED*
DSPL-13	1 Februari 2022	*FIXED*
DSPL-15	2 Februari 2022	*FIXED*
DSPL-16	1 Februari 2022	*FIXED*
DSPL-17	2 Februari 2022	*IN-PROGRESS*
DSPL-18	3 Februari 2022	*FIXED*
DSPL-19	3 Februari 2022	*FIXED*
DSPL-20	3 Februari 2022	*FIXED*
DSPL-21	4 Februari 2022	*FIXED*
DSPL-22	6 Februari 2022	*FIXED*
DSPL-23	6 Februari 2022	*IN-PROGRESS*

Tabel 5 Keterangan *bug* DSPL-17 dan DSPL-23.

Bug_ID	Keterangan
DSPL-17	*Game* gagal menyesuaikan lebar tampilan *game* dengan lebar layar, sehingga menyebabkan *gap* atau jarak kosong pada sisi kanan dan kiri layar. Akan tetapi game tetap bisa berjalan dengan baik, hanya saja dengan lebar tampilan game yang tidak menutup seluruh layar perangkat.
DSPL-23	AI dapat menendang bola pada kondisi yang belum diketahui pasti. *Bug* ini jarang terjadi. Jika *bug* ini terjadi, satu-satunya jalan untuk memperbaikinya hanya dengan *restart* pertandingan.

E. Verification

Pada tahap ini model pengujian *regression testing* digunakan untuk memastikan *bug* benar-benar sudah diperbaiki. Pada tahap sebelumnya hampir semua *bug* berhasil diperbaiki, kecuali *bug* DSPL-17 dan DSPL-23. Sehingga *bug* DSPL-17 dan DSPL-23 tidak termasuk dalam daftar pengujian *regression testing*. Hasil dari *regression testing* bisa dilihat pada tabel 6.

F. Closure

Setiap *bug* yang lolos pengujian *regression testing* atau terbukti *bug* tidak muncul kembali, status *bug*nya diubah menjadi *CLOSED*. Karena tidak ada *bug* yang muncul kembali tim penguji tidak mengembalikan laporan *bug* ke tim yang memperbaiki *bug* di tahap sebelumnya. Hasil dari *regression testing* bisa dilihat pada tabel 6.

Tabel 6 Hasil *regression testing*.

Bug_ID	Status
DSPL-1	*CLOSED*
DSPL-4	*CLOSED*
DSPL-5	*CLOSED*
DSPL-8	*CLOSED*
DSPL-10	*CLOSED*
DSPL-11	*CLOSED*
DSPL-12	*CLOSED*
DSPL-13	*CLOSED*
DSPL-15	*CLOSED*
DSPL-16	*CLOSED*
DSPL-18	*CLOSED*
DSPL-19	*CLOSED*
DSPL-20	*CLOSED*
DSPL-21	*CLOSED*
DSPL-22	*CLOSED*

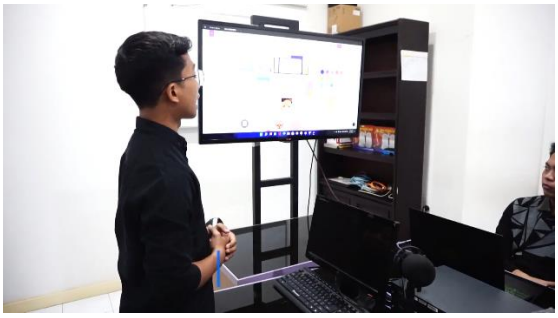
G. Reporting

Tahap terakhir dari pengujian *game* Dash Soccer: Premier League adalah membuat laporan mengenai hasil pengujian. Laporan yang dihasilkan bisa digunakan untuk menilai kelayakan *game* untuk dirilis. Hasil akhirnya dari total 20 *bug* yang ditemukan selama pengujian, ada 15 yang bisa diperbaiki, sedangkan 2 *bug* masih belum bisa diperbaiki karena tingkat kompleksitas bug yang tinggi, dan 3 *bug* tidak valid karena di luar scope pengujian. Laporan hasil pengujian bisa dilihat pada tabel 7.

Tabel 7 Laporan hasil pengujian *game* Dash Soccer: Premier League.

Bug_ID	Nama	Status	Severity	Priority
DSPL-1	Mode *tournament* tidak bisa dibuka	*CLOSED*	*Highest*	*Highest*
DSPL-4	*Disable audio* hanya berlaku di *main screen*	*CLOSED*	*High*	*Highest*
DSPL-5	Nama *game* belum sesuai	*CLOSED*	*Low*	*Highest*
DSPL-6	Belum ada *tutorial*	*REJECTED*	-	-
DSPL-7	Tim *penalty random*	*DIFFERED*	-	-
DSPL-8	Tim masih menggunakan aset lama atau belum liga premier	*CLOSED*	*High*	*Highest*
DSPL-9	*Button exit* digunakan sebagai button *tutorial*	*REJECTED*	-	-
DSPL-10	Admob belum diintegrasikan dalam *game*	*CLOSED*	*Low*	*Highest*
DSPL-11	*Icon* tim di *shop stretched*	*CLOSED*	*Low*	*Medium*
DSPL-12	*Toast massage* koin kurang	*CLOSED*	*Lowest*	*Medium*
DSPL-13	*Buy formation* di *shop* masih menggunakan logo Manchester United	*CLOSED*	*Low*	*Medium*
DSPL-15	Warna *button next back*, kurang menyala	*CLOSED*	*Lowest*	*Low*
DSPL-16	Tampilan *gameplay* belum sesuai desain UI/UX	*CLOSED*	*Low*	*Medium*
DSPL-17	Menutup *gap* kosong tampilan	*IN-PROGRESS*	*High*	*High*
DSPL-18	Warna *text* jumlah trofi dan koin terlalu menyatu dengan *background*	*CLOSED*	*Lowest*	*High*
DSPL-19	Aset *background tournament bracket* tidak sesuai desain UI/UX	*CLOSED*	*Low*	*High*
DSPL-20	Aset formasi masih menggunakan aset game template	*CLOSED*	*Low*	*Medium*
DSPL-21	Banner admob tidak *center* dan menutupi aset lain	*CLOSED*	*High*	*High*
DSPL-22	Inkonsistensi *visual stat* tim	*CLOSED*	*Low*	*Low*
DSPL-23	AI menendang bola keluar lapangan	*IN-PROGRESS*	*Highest*	*Highest*

Dua *bug* yang belum bisa diperbaiki adalah *bug* DSPL-17 dan DSPL-23 yang memiliki tingkat *severity high* dan *highest*, yang artinya *bug* memiliki dampak yang signifikan pada *game*. Akan tetapi *bug* DSPL-17 hanya berpengaruh terhadap tampilan *game* yang tidak bisa memenuhi seluruh layar perangkat, namun game tetap dapat berjalan dengan baik. Sedangkan *bug* DSPL-23 adalah *bug* yang jarang terjadi, akan tetapi termasuk *game breaking bug*. Jika *bug* terjadi akan menghalangi *game* atau pemain untuk menjalankan mekanisme fundamental *game* untuk sementara waktu, pemain masih dapat mengakali bug ini dengan restart pertandingan. Dengan pertimbangan masa pengembangan *game* sudah hampir habis dan *game* sudah sesuai dengan desain, maka *game* Dash Soccer: Premier League boleh dirilis dengan catatan tim *programmer* harus segera membuat perbaikan terhadap *bug* DSPL-17 dan DSPL-23 yang akan diimplementasikan sebagai *update game* setelah game dirilis di Play Store. Dokumentasi meeting hasil pengujian bisa dilihat pada gambar 9.



Gambar 9 Dokumentasi *meeting* hasil pengujian.

V. KESIMPULAN

Dari praktek penerapan defect management dan defect life cycle dalam masa pengujian game Dash Soccer: Premier League, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Penerapan *defect management process* membuat proses manajemen *bug* lebih sistematis, karena setiap *bug* yang teridentifikasi selama pengujian melalui serangkaian tahapan yang perlu dilalui sampai dengan *bug* dinyatakan berhasil diperbaiki atau tidak. Kemudian dengan adanya penerapan *defect management process* memberikan efek pada *bug report* yang dihasilkan, efek

tersebut berupa penambahan komponen kategori (*severity & priority*) dan penjadwalan perbaikan *bug*.

2. Penerapan *defect life cycle* membantu mengidentifikasi status pengerjaan *bug* yang dilaporkan secara *real time*, karena status *bug* akan diperbarui setiap berpindah dari satu tahapan ke tahapan lain dalam *defect management process*. Status tersebut juga dicantumkan dalam *bug report*.
3. Pemanfaatan Jira Software sebagai *bug tracking tool* mempermudah pelaporan dan monitoring *bug*. Karena setiap ada perkembangan atau perubahan informasi *bug* dapat dipantau secara *real time* menggunakan Jira Software.

REFERENSI

[1] Hamilton, T. (2022, April 30). Defect Management Process in Software Testing (Bug Report Template). Guru99. Retrieved June 2, 2022,J.

[2] Suma, V., & Nair, T. R. (2012). Defect management strategies in software development. *arXiv preprint arXiv:1209.5573*. from https://www.guru99.com/defect-management-process.html#10.

[3] Hamilton, Thomas. "Defect/Bug Life Cycle in Software Testing." *Guru99*, 23 April 2022, https://www.guru99.com/defect-life-cycle.html. Accessed 9 June 2022.

[4] Mutalik, Deepika. "What is Defect/Bug Life Cycle in Software Testing?" *Tools QA*, 7 July 2021, https://www.toolsqa.com/software-testing/defect-life-cycle/. Accessed 9 June 2022.

[5] Hamilton, T. (2022, April 30). Defect Management Process in Software Testing (Bug Report Template). Guru99. Retrieved June 2, 2022, from https://www.guru99.com/defect-management-process.html#10.

[6] SARAGIH, E. (2020, September 23). Mengenal Apa Itu Play Testing Pada Game? | Berita. Gamelab.ID. Retrieved May 31, 2022, from https://www.gamelab.id/news/234-mengenal-apa-itu-play-testing-pada-game.

[7] Hamilton, T. (2022, April 30). What is Adhoc Testing? Types with Example. Guru99. Retrieved May 20, 2022, from https://www.guru99.com/adhoc-testing.html.

[8] Hamilton, T. (2022, March 17). What is Regression Testing? Definition, Test Cases (Example). Guru99. Retrieved May 20, 2022, from https://www.guru99.com/regression-testing.html.

[9] Raja, K., Reddy, R., & Pradesh, A. (2012). Implementation Of Effective Defect Tracking System In Software Egineering. 1(8), 1–8.

[10] Sirshar, M., Noor, J., Rashid, R., & Daud, M. (2019). *A Framework for Software Defect Management Process in Software Quality Assurance*. *December*. www.preprints.org