p-ISSN 2085-8507
e-ISSN 2722-3280

ASTRA polytechnic member of ASTRA

TECHNOLOGIC

VOLUME 13 NOMOR 1 | JUNI 2022

POLITEKNIK ASTRA

Jl. Gaya Motor Raya No. 8 Sunter II Jakarta Utara 14330
Telp. 021 651 9555, Fax. 021 651 9821
www.polman.astra.ac.id
Email : editor.technologic@polman.astra.ac.id

DEWAN REDAKSI
Technologic

Kantor Editor:
Politeknik Astra
Jl. Gaya Motor Raya No. 8 Sunter II Jakarta Utara 14330
Telp. 021 651 9555, Fax. 021 651 9821
www.polman.astra.ac.id
Email : editor.technologic@polman.astra.ac.id

EDITORIAL

Pembaca yang budiman,

Puji syukur kita dapat berjumpa kembali dengan Technologic Volume 13 No. 1, Edisi Juni 2022.

Pembaca, Jurnal Technologic Edisi Juni 2022 kali ini berisi 14 manuskrip dan ada perubahan nama institusi penerbit dari Politeknik Manufaktur Astra menjadi Politeknik Astra.

Atas nama Redaksi dan Editor, kami doakan semoga dalam keadaan sehat selalu, seiring dengan semakin menurunnya kasus pandemic Covid-19. Kami haturkan terima kasih atas kepercayaan para peneliti dan pembaca, serta selamat menikmati dan mengambil manfaat dari terbitan Jurnal Technologic kali ini.

Selamat membaca!

DAFTAR ISI

ASTRA polytechnic member of ASTRA
TECHNOLOGIC, VOLUME 13, NOMOR 1
Politeknik Astra
Juni 2022

DETEKSI DINI IDENTIFIKASI INSIDEN PADA KEJADIAN ANOMALI PERANGKAT LUNAK DENGAN SISTEM PENDETEKSI ANOMALI PERANGKAT LUNAK STUDI KASUS DI ASTRA LIFE

Sasmito Budi Utomo[1], Mela Hidayah[2], dan Noer Lisna Anjani[3]
Program Studi Manajemen Informatika, Jurusan Informatika, Politeknik Astra, Jalan Gaya Motor Raya No. 8 Sunter, Jakarta 14330, Indonesia
E-mail : sasmito.budi@polman.astra.ac.id[1], melamela.hidayah@gmail.com[2], noerlisnaa@gmail.com[3]

***Abstrak*--PT Astra Aviva Life atau Astra Life merupakan sebuah perusahaan yang bergerak pada bidang asuransi khususnya pelayanan asuransi jiwa sering mengalami anomali atau penyimpangan pada perangkat lunak utamanya. Anomali ini menghambat proses layanan bisnis asuransi karena sistem tidak mampu berfungsi dengan semestinya. Deteksi dan pemulihan anomali membutuhkan waktu untuk memeriksa dan memperbaiki perangkat lunak agar berfungsi kembali sebagaimana mestinya. Penelitian ini bertujuan membangun sebuah sistem pemantau perangkat lunak untuk mendeteksi lebih awal anomali atau penyimpangan perangkat lunak yang mungkin muncul sehingga tetap terjaga stabilitasnya, yang dinamakan Sistem Pendeteksi Anomali Perangkat Lunak (SIMPEL). SIMPEL merupakan aplikasi berbasis web yang dibangun dengan menggunakan metodologi Scrum, bahasa pemrograman Java, *framework* Spring MVC, basis data Oracle, server web Tomcat, dan menggunakan arsitektur jaringan terdistribusi *three-tier-client-server*. Kesimpulannya menunjukkan SIMPEL mampu mendeteksi lebih dini anomali yang terjadi dengan mamantau target *apps* secara otomatis dan berdampak memperpendek durasi penanganan anomali perangkat lunak dengan menurunkan waktu identifikasi insiden hingga 53%.**

***Kata Kunci:* Identifikasi Insiden, Sistem Pendeteksi, Anomali Perangkat Lunak, Astra Aviva Life.**

I. PENDAHULUAN

PT Astra Aviva Life atau Astra Life merupakan perusahaan yang bergerak di bidang asuransi khususnya pelayanan asuransi jiwa. Selama menjalankan bisnis asuransinya, Astra Life sangat bergantung dengan berbagai perangkat lunak yang dimilikinya. Kehandalan dan ketersediaan layanan perangkat lunak menjadi sangat penting. Sekecil apapun kegagalan atau penyimpangan perangkat lunak dalam menjalankan fungsinya akan menyebabkan bisnis Astra Life terganggu. Selama ini perusahaan menyebut kegagalan atau penyimpangan ini dengan anomali.

Kondisi saat ini terjadinya anomali dilaporkan oleh pengguna perangkat lunak kepada *help desk* (Departemen IT), di mana proses ini disebut identifikasi insiden. Selama proses penanganan pelaporan, *help desk* membutuhkan waktu yang relatif lama untuk melakukan pemeriksaan maupun perbaikan kejadian anomali tersebut. Selama proses pemulihan ini sistem berhenti berfungsi yang akan mengakibatkan pelayanan bisnis terganggu.

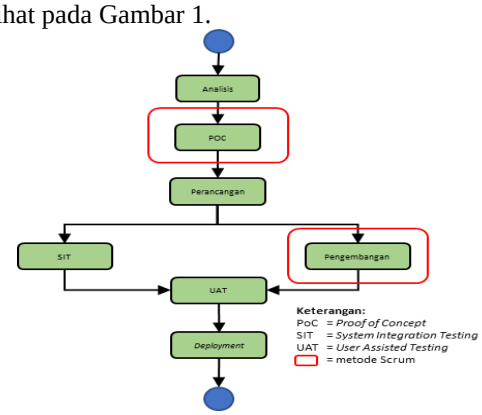
Masalah mendeteksi anomali aplikasi ini cukup menarik. Beberapa pembahasan pendeteksian anomali telah dilakukan, misalnya kerangka kerja pendeteksi anomali otomatis menurut Syed Shariyar Murtaza [1] pada penelitian yang berjudul TotalADS: Automated Software Anomaly Detection System. Selain itu M.S. Josephine [2] telah melakukan perancangan sistem pakar untuk mendeteksi kesalahan dan koreksi dalam Pengembangan Perangkat Lunak. Sedangkan Hemank Lamba [3] mengembangkan pendekatan berbasis model untuk mendeteksi anomali di arsitektur perangkat lunak. Dan menurut Sheng-Chi Yang [4] yang mengusulkan prosedur mendeteksi anomali otomatis dalam Aplikasi *Near-Real Time.* Anomali merupakan suatu keganjilan, keanehan atau penyimpangan dari yang biasa atau dari keadaan normal yang berbeda dari kondisi mayoritas [5]. Anomali perangkat lunak merupakan suatu keganjilan, keanehan atau penyimpangan dari yang biasa atau dari keadaan normal yang berbeda dari kondisi mayoritas yang terjadi pada program yang digunakan untuk mengoperasikan komputer [6]

Tujuan penelitian ini adalah membangun aplikasi pemantau stabilitas perangkat lunak yang dinamakan Sistem Pendeteksi Anomali Perangkat Lunak atau disingkat SIMPEL. SIMPEL diharapkan mampu mendeteksi lebih dini anomali yang terjadi pada perangkat lunak bisnis asuransi dengan studi kasus di Astra Life. SIMPEL merupakan aplikasi berbasis web yang dikembangkan dengan menggunakan bahasa pemrograman JAVA, *framework* Spring MVC, basis data Oracle, server web Tomcat, dan dengan menggunakan arsitektur jaringan terdistribusi *three-tier-client-server*. SIMPEL mendeteksi anomali dengan memantau target *apps* dengan melakukan pengambilan data dari aplikasi yang dipantau (target *apps*) secara otomatis dan waktu nyata, melakukan proses validasi, dan menampilkan hasilnya secara

daring dan waktu nyata. SIMPEL merupakan bentuk pemanfaatan teknologi untuk melakukan perubahan proses sesuai konsep *Business Process Improvement* (BPI) [7]. SIMPEL memangkas proses pelaporan oleh pengguna ke *customer service*, sehingga total proses menjadi ringkas dan hanya melibatkan *help desk*. Kontribusi penelitian ini adalah memperkuat penelitian sebelumnya terkait pendeteksian anomali aplikasi dan membantu Astra Life meningkatkan efisiensi kerja dengan melakukan *business process improvement* (BPI) [7] untuk memangkas proses penanganan anomali dengan aplikasi SIMPEL.

II. TAHAPAN DAN METODOLOGI

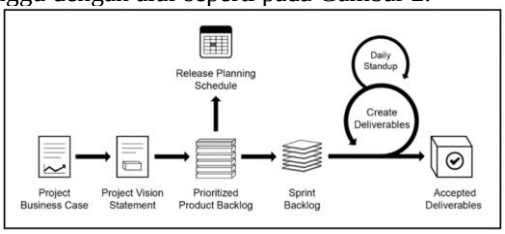
SIMPEL dibangun menggunakan standar tahapan pengembangan aplikasi di Astra Life, yaitu Analisis, *Proof of Concept* (PoC), Perancangan, Pengembangan, *System Integration Testing* (SIT), *User Assisted Testing* (UAT) [8], dan Deployment Production. Untuk tahap PoC dan tahap Pengembangan menggunakan konsep metodologi scrum [9]. Tahap pengembangan dan SIT dilakukan secara bersamaan. Tahapan yang dilakukan dapat dilihat pada Gambar 1.



Gambar 1. Tahapan Pembangunan SIMPEL

Analisis dilakukan untuk mendapatkan kondisi yang sedang terjadi, permasalahan yang ada, dan permintaan solusi oleh pengguna. Analisis dilakukan dengan mewawancarai pengguna, *customer service*, *help desk*, dan pimpinan kerja yang terkait. Selain wawancara, analisis dilakukan dengan observasi di lapangan terkait terjadinya anomali. Hasil analisis diperoleh dalam bentuk masalah yang muncul dan solusi yang ditawarkan yang didokumentasikan dalam bentuk dokumen *user stories*. *Proof of Concept* (PoC) dilakukan untuk menunjukkan dan memverifikasi bahwa solusi yang ditawarkan berpotensi untuk dikembangkan di lingkungan sebenarnya [9]. Dalam PoC, purwarupa dirancang untuk mentukan kelayakan finansial maupun teknis, membantu memahami permintaan pengguna, dan membantu untuk pengambilan keputusan pembuatan desain proyek di awal proses [9]. Tahap perancangan merancang kebutuhan data yang digambarkan dengan *Entity Relationship Diagram* (ERD) dan *Physical Data Model* (PDM) [11][12]. Selain itu, tahap perancangan juga merancang antarmuka pengguna sesuai purwarupa yang dikembangkan di PoC.

Pengembangan SIMPEL menggunakan metodologi scrum. Scrum merupakan salah satu metodologi *agile* yang bertumpu pada kekuatan kolaborasi tim [9]. Secara umum Scrum memiliki lima tahapan, yaitu memulai (*initiate*), persiapan dan perkiraan (*plan and estimate*), pelaksanaan (*implement*), tinjauan ulang dan perbaikan (*review and retrospect*), dan rilis (*release*) [9]. Tahapan-tahapan tersebut dilakukan dalam satu periode pengembangan produk dengan durasi tertentu yang terdiri dari beberapa *sprint*. *Sprint* adalah batasan waktu untuk pengembangan produk yang biasanya berdurasi 1-4 minggu dengan alur seperti pada Gambar 2.



Gambar 2. Alur Proses Scrum Satu Kali *Sprint* [9]

Dibanding dengan metodologi pengembangan perangkat lunak yang lain, metodologi Scrum memiliki kelebihan sifat yang adaptif, iteratif, cepat, fleksibel, dan efektif. Selain itu, metodologi Scrum menggunakan strategi fleksibel, di mana hasil yang diperoleh mulai terlihat pada awal fase pengerjaan. Dengan demikian proses eksekusi program dapat diubah sewaktu-waktu, sekalipun pada saat pengerjaan tugas tengah dilakukan [10]. Namun di sisi lain, Scrum menuntut komitmen tim dengan komitmen tinggi dan penguasaan teknologi yang akan diimplementasikan. Berdasarkan kelebihan maupun kekurangan tersebut, metodologi ini dipilih karena sangat sesuai dengan kondisi dan situasi yang terjadi pada lingkungan di mana penelitian ini dilakukan termasuk ketersediaan tim dengan penguasaan teknologi yang berpengalaman. Pengembangan SIMPEL dilakukan melalui 12 *sprint* dengan durasi setiap *sprint* rata-rata satu minggu dan untuk *sprint planning* dilakukan di setiap awal minggu.

Pengujian SIMPEL dilakukan dengan dua cara, yaitu SIT dan UAT. Tahapan SIT dilakukan secara paralel di dalam proses pengembangan. Pada tahap ini, *developer* melakukan pengujian secara mandiri terhadap setiap fungsi yang telah diselesaikan.

Tahapan UAT di Astra Life dilakukan oleh *developer* bersama pengguna SIMPEL dari Astra Life. UAT menguji aplikasi SIMPEL sesuai permintaan pengguna yang didefinisikan dalam *project charter*.

Selanjutnya dilakukan *deployment* SIMPEL untuk dijalankan di server *production*. Sebelum dijalankan di server *production*, dilakukan pengujian pembandingan antara proses penanganan anomali yang lama dan yang baru. Pengujian dilakukan dengan membandingkan durasi waktu penanganan anomali sebelum dan sesudah instalasi SIMPEL. Data durasi waktu penanganan sebelum adanya SIMPEL diperoleh dari pihak Astra Life berdasarkan kejadian yang pernah dialami sebelumnya yang dipilih berdasarkan waktu tercepat identifikasi insiden yang pernah terjadi, sedangkan durasi waktu setelah adanya SIMPEL diperoleh berdasarkan jadwal yang berjalan pada masing-masing perangkat lunak yang dipantau selama pengujian.

Data yang digunakan untuk pengujian berupa nama aplikasi, nama *job*, persentase penurunan waktu identifikasi insiden, dan waktu durasi identifikasi insiden dalam satuan jam. Data tersebut ditempatkan dalam sebuah tabel, sehingga akan terlihat nilai perbandingannya. Persentase penurunan waktu identifikasi insiden dihitung dengan persamaan (1).

$$P = 100\% - \left(\frac{x2}{x1} x\ 100\%\right) \qquad (1)$$

Keterangan:
P = Persentase penurunan waktu identifikasi insiden
x1 = Waktu identifikasi sebelum adanya SIMPEL
x2 = Waktu identifikasi sesudah adanya SIMPEL

Setelah angka presentasi penurunan diperoleh, selanjutnya dihitung rata-rata waktu identifikasi insiden setelah adanya SIMPEL dengan persamaan (2).

$$R = \frac{x1+x2+x3+\cdots+xn}{n} \qquad (2)$$

Keterangan:
R = Rata-rata waktu identifikasi insiden
n = Jumlah aplikasi yang dipantau
x1 = Penurunan waktu identifikasi job aplikasi ke-1
x2 = Penurunan waktu identifikasi job aplikasi ke-2
x3 = Penurunan waktu identifikasi job aplikasi ke-3
xn = Penurunan waktu identifikasi job aplikasi ke-n

III. HASIL DAN PEMBAHASAN

Hasil analisis menunjukkan bahwa penanganan anomali di Astra Life dilakukan melalui proses identifikasi insiden, investigasi, *troubleshooting*/ *bug fixing*, dan administratif penyelesaian insiden. Investigasi perangkat lunak yang digunakan hanya dilakukan ketika terjadi anomali pada perangkat lunak yang dilaporkan pengguna melalui telepon. Selama analisis, perangkat lunak yang dipantau yaitu: *Relationship Management Customer Service* (RMCS), iProsper, iPlus dan *Integration Email*. Setelah melakukan pamantauan, permasalahan yang muncul sebagai berikut:

1. *Job* COA Balance MTD dan Generate Journal Report pada aplikasi iPlus tidak berjalan, maka anomali ini akan berdampak pada kesalahan penghitungan laporan keuangan perusahaan.
2. Anomali pada aplikasi RMCS dan iProsper yang langsung dirasakan oleh nasabah, menghambat proses penjualan kepada nasabah.
3. Pelaporan *job Decrease Contract Sum Assured* yang lambat mengakibatkan nilai klaim nasabah berpotensi untuk mendapatkan nilai maksimal, sehingga dapat merugikan perusahaan.
4. Anomali pengiriman polis menggunakan Integration Email menyebabkan nasabah tidak menerima polis, sehingga berpotensi nasabah membatalkan pengajuan asuransi apabila permasalahan tersebut tidak diketahui oleh pihak perusahaan.
5. Anomali pada *job Generate Billing* akan mengakibatkan kesalahan pembayaran premi nasabah, di mana nasabah tidak akan menerima tagihan pembayaran.

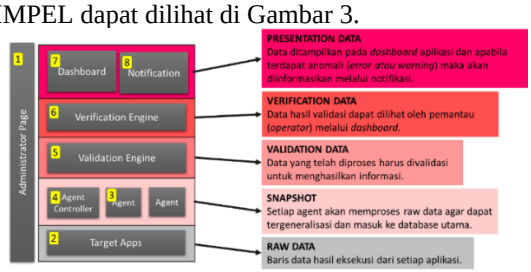
Berdasarkan hasil analisis dan permasalahan yang ditemukan, maka dibutuhkan sebuah sistem yang mampu mendeteksi anomali dan mempercepat proses penanganannya yang dinamakan SIMPEL.

SIMPEL dirancang terdiri dari lima lapisan data (data *layer*), yaitu *raw* data, *snapshot*, validasi data, verifikasi data, dan presentasi data. Konsep data *layer* digunakan untuk mendukung desain SIMPEL yang modular. Dari kelima lapisan tersebut, SIMPEL memiliki delapan modul, yaitu modul *administrator page*, *target apps*, *agent*, *agent controller*, *validation engine*, *verification engine*, *dashboard*, dan *notification*.

1. Administrator Page adalah modul untuk mendaftarkan aplikasi dan *rule* yang akan dipantau.
2. Target Apps adalah aplikasi yang akan dipantau.
3. Agent adalah modul untuk mengambil data dari aplikasi yang akan dipantau. Agent dapat mengambil data dari basis data, file system, dan web service.
4. Agent Controller adalah modul untuk mengatur jalannya modul agent.
5. Validation engine adalah modul untuk menyimpan aturan keberhasilan scheduler dari setiap aplikasi. Pada modul ini terdapat validation engine dan rule engine yang bekerja sama untuk melakukan proses verifikasi data.

6. Verification engine adalah modul untuk menyimpan hasil verifikasi dari operator.
7. Dashboard adalah modul untuk menampilkan hasil eksekusi secara keseluruhan (presentation).
8. Notification adalah modul untuk memberitahukan kepada operator.

Ruang lingkup pengembangan delapan modul SIMPEL dapat dilihat di Gambar 3.

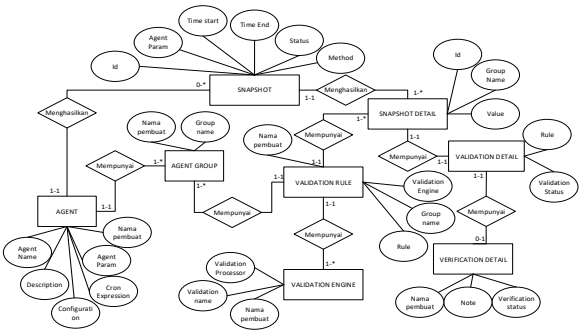


Gambar 3. Pemetaan Lapisan Data SIMPEL

Setelah melalui proses Proof of Concept (PoC), di mana dibuat purwarupa sebagai model untuk memastikan kelayakan modul dan lapisan data dapat diterapkan dan dapat dikembangkan, maka dilakukan pengembangan SIMPEL. Pengembangan dimulai oleh Tim Scrum dengan pemodelan data untuk lapisan data dan modul, membuat *use case diagram*, dan membuat *product backlog*. Pemodelan data menghasilkan delapan entitas yaitu: *agent*, *agent group*, *validation engine*, *validation rule*, *snapshot*, *snapshot detail*, *validation detail*, dan *verification detail*. Untuk melihat hubungan antar-entitas yang terbentuk dilakukan dengan membuat *Entity Relationship Diagram* (ERD), yang digunakan menjadi acuan untuk pembuatan *Physical Data Model* (PDM) [13]. ERD yang dihasilkan pada pemodelan data ini dapat dilihat pada Gambar 4.

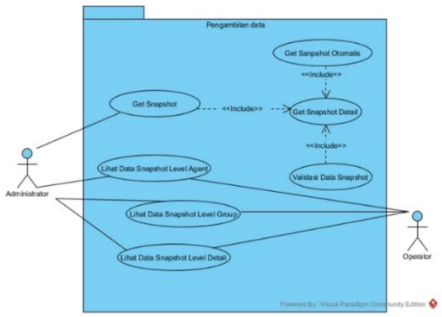
Selanjutnya dilakukan pemodelan fungsi menggunakan *use case diagram*. Pada pemodelan fungsi dilakukan pemodelan empat fungsi utama yaitu:

1. Kelola *agent manager* meliputi fungsi tambah, lihat, hapus, dan ubah *agent*.
2. Kelola *validation rule* meliputi fungsi tambah, lihat, hapus, dan ubah *rule*.
3. Pengambilan data meliputi fungsi *get snapshot*: *get snapshot* otomatis dan *get snapshot* detail, validasi data *snapshot*, dan lihat data *snapshot*: level *agent*, level detail, dan level grup.
4. Pemantauan meliputi fungsi mencari dan verifikasi data *snapshot* level grup, mencari dan verifikasi data *snapshot* level detail.



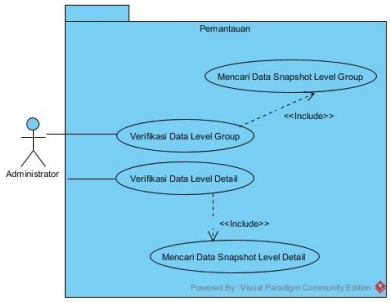
Gambar 4. ERD yang dihasilkan pada tahap pemodelan data untuk SIMPEL

Use case diagram Pengambilan Data dapat dilihat pada Gambar 5. sedangkan *use case* diagram Pemantauan dapat dilihat pada Gambar 6. Selain pemodelan data dan fungsi, pada pengembangan ini juga dimodelkan proses dengan menggunakan *Data Flow Diagram* (DFD). DFD yang dihasilkan adalah *Context Diagram* (DFD Level 0) hingga DFD level 2. Pemodelan ini memodelkan tujuh proses utama pada SIMPEL yaitu: mengelola data master, mengambil data dari target *apps*, melakukan validasi data, menampilkan validasi detail, menampilkan *snapshot*, melakukan verifikasi data, dan menampilkan data verifikasi. DFD level 1 dapat dilihat pada Gambar 7.

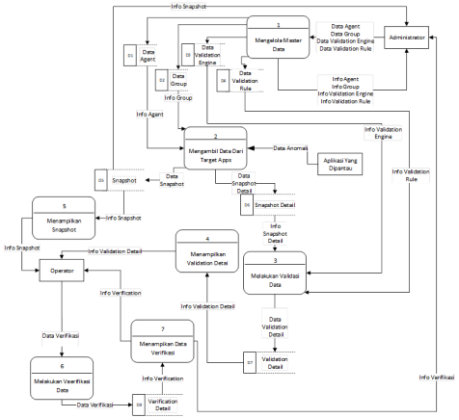


Gambar 5. Use Case Diagram Fungsi Pengambilan Data

Selanjutnya dilakukan penulisan kode program yang dibuat secara berurutan sesuai *sprint backlog* pada *sprint* yang sudah ditentukan, yaitu 12 *sprint* yang harus dijalankan. Berdasarkan pemodelan fungsi dan proses, penulisan kode program menghasilkan 126 *file* baik dalam bentuk java maupun html dan tersimpan ke dalam 101 direktori. Pada tahap pengembangan secara bersamaan dilakukan SIT. Pada SIT, *developer* melakukan pengujian secara mandiri terhadap setiap fungsi yang telah diselesaikan.



Gambar 6. Use Case Diagram Fungsi Pemantauan



Gambar 7. DFD pada pemodelan proses SIMPEL

Pada UAT ditemukan beberapa *bug* maupun perubahan minor fungsi yang disesuaikan dengan kebutuhan pengguna. *Bug* maupun perubahan fungsi dibuatkan *product backlog* untuk dilakukan perbaikan dan revisinya. Selanjutnya dilakukan UAT kembali untuk menguji setiap perbaikan *bug* maupun perubahan fungsi untuk memastikan keberhasilannya sebelum aplikasi digunakan. Proses ini berulang hingga seluruh fungsi berjalan sesuai permintaan pengguna. Setelah aplikasi memenuhi kebutuhan pengguna, lolos pengujian UAT, dan disetujui oleh pihak-pihak yang bertanggung jawab terhadap jalannya aplikasi di Astra Life, selanjutnya SIMPEL akan di-*deploy* ke server *production*.

Sebelum SIMPEL dijalankan di server *production*, dilakukan pengujian efektifitasnya. Pengujian ini bertujuan melihat nilai perbandingan waktu identifikasi insiden sebelum dan saat menggunakan SIMPEL. Pertama disiapkan sebuah tabel yang memiliki kolom nama aplikasi, nama *job*, kondisi keberadaan SIMPEL, persentase penurunan waktu identifikasi insiden, dan waktu durasi identifikasi insiden dalam jam. Nama aplikasi merupakan implemetasi dari modul target *apps*, yaitu aplikasi yang dipantau oleh SIMPEL. Aplikasi yang dipantau pada saat pengujian dilakukan ada empat, yaitu RMCS, iProsper, iPlus, dan Integration Email. Keempat aplikasi ini terpilih untuk dipantau karena merupakan aplikasi yang vital untuk menunjang bisnis perusahaan. Setiap aplikasi memiliki *job* masing-masing. *Job* tersebut merupakan obyek spesifik yang akan dipantau dari setiap aplikasi. Penentuan *job* ini dilakukan berdasarkan tingkat kritikal *job* sebagai penunjang utama pelayanan nasabah perusahaan.

Setelah data terkumpul, selanjutnya dihitung persentase penurunan waktu identifikasi insiden dengan Persamaan (1). Berdasarkan nilai persentase penurunan waktu identifikasi insiden yang diperoleh selanjutnya dihitung nilai rata-rata waktu identifikasi insiden saat menggunakan SIMPEL dengan Persamaan (2). Berikut ini contoh perhitungan untuk waktu identifikasi insiden dengan Persamaan (1).

Contoh nama aplikasi (*agent*): Integration Email:

1. $x1$ = 24 jam,
2. $x2$ = 1 jam,

dengan menggunakan Persamaan (1) menjadi:

$$P = 100\% - \left(\frac{x2}{x1} x 100\%\right)$$
$$P = 100\% - (\frac{1}{24} x 100\%)$$
$$P = 100\% - (4{,}16\%)$$
$$P = 95.833\%$$
$$P = 96\%$$

Hasil perhitungan pada contoh menunjukkan persentase penurunan waktu identifikasi insiden *agent* Integration Email = 96%. Proses perhitungan nilai ini diulang hingga seluruh *agent* yang dipantau selesai dihitung. Setelah semua nilai persentase penurunan waktu identifikasi insiden diperoleh, selanjutnya dihitung nilai rata-rata waktu identifikasi insiden dengan Persamaan (2). Dari perhitungan seluruh nilai persentase penurunan waktu identifikasi insiden diperoleh data sebagai berikut:

1. RMCS Submission ($x1$) = 0 %
2. iProsper Submission ($x2$) = 50 %
3. COA Balance MTD ($x3$) = 70 %
4. Generate Journal Report ($x4$) = 50 %
5. Decrease Contract Sum Assured ($x5$) = 58 %
6. Generate Billing ($x6$) = 50 %
7. Integration Email E-Policy ($x7$) = 96 %

dengan menggunakan Persamaan (1) menjadi:

$$R = \frac{x1+x2+x3+x4+x5+x6+x7}{7}$$
$$R = \frac{0\%+50\%+70\%+50\%+58\%+50\%+96\%}{7}$$
$$R = \frac{374\%}{7}$$
$$R = 53{,}428\%$$
$$R = 53\% \text{ (pembulatan)}$$

Hasil perhitungan Persamaan (2) menunjukkan nilai rata-rata persentase penurunan identifikasi insiden setelah ada SIMPEL 53%. Nilai ini menunjukkan bahwa pada uji coba menggunakan

SIMPEL berhasil mengurangi waktu identifikasi insiden sebesar 53%. Data perbandingan dan hasil pengujian efektifitas SIMPEL dapat dilihat pada Tabel 1.

Tabel 1. Perbandingan waktu sebelum dan saat menggunakan SIMPEL beserta nilai rata-rata persentasi penurunan waktu identifikasi insiden

Nama Aplikasi	**Nama *Job***	**Kondisi Keberadaan SIMPEL**	**Persentase Penurunan Waktu Identifikasi Insiden**	**Durasi Identifikasi Insiden (Jam)**
RMCS	RMCS Submission	Sebelum	0%	0.25
		Sesudah		0.25
iProsper	iProsper Submission	Sebelum	50%	2
		Sesudah		1
iPlus	COA Balance MTD	Sebelum	70%	10
		Sesudah		3
iPlus	Generate Journal Report	Sebelum	50%	6
		Sesudah		3
iPlus	Decrease Contract Sum Assured	Sebelum	58%	10
		Sesudah		5
iPlus	Generate Billing	Sebelum	50%	6
		Sesudah		3
Integration Email	E-Policy	Sebelum	96%	24
		Sesudah		1
Rata-rata persentase penurunan waktu identifikasi insiden			53%	

IV. KESIMPULAN

Berdasarkan hasil pengujian dengan data dan kasus uji pada empat target *apps*, SIMPEL mampu mendeteksi lebih dini anomali yang terjadi pada perangkat lunak dengan mengambil data secara otomatis dari target *apps* yang dipantau. SIMPEL efektif menurunkan waktu identifikasi insiden yang berdampak memperpendek durasi penanganan anomali perangkat lunak hingga 53%. Kemampuan SIMPEL mendeteksi anomali ini memperkuat dan memperkaya pembahasan terkait pendeteksian anomali perangkat lunak yang telah dilakukan sebelum penelitian ini.

V. DAFTAR PUSTAKA

[1] S. S. Murtaza1, A. Hamou-Lhadj, W. Khreich, and M. Couture, "TotalADS: Automated Software Anomaly Detection System," 2014.

[2] K. S. S. M.S.JOSEPHINE, "Software Error Detection and Correction (SEDC) Using Layer Based Approach in Expert System," *ISSN 2076-3328*, 2010.

[3] T. J. G. Hemank Lamba and J. P. Bradley Schmerl, Javier Cámara, David Garlan, "A Model-based Approach to Anomaly Detection in Software Architectures," *ACM ISBN 978-1-4503-2138-9*, 2016.

[4] M.-C. W.-M. K.-H. Y. Sheng-Chi Yang, "An Automated Anomaly Detection Procedure for Hourly Observed Precipitation in Near-Real Time Application," *ISBN 978-981-15-5436-0*, 2020.

[5] H. S. M Echols John, *An English-Indonesian Dictionary*. Gramedia, 1995.

[6] Kemdikbud, "Kamus Besar Bahasa Indonesia Daring," 2020. [Online]. Available: https://kbbi.kemdikbud.go.id/entri/modular. [Accessed: 04-Mar-2020].

[7] T. Alan Dennis, Barbara Haley Wixom, *Systems Analysis Design UML Version 2.0 An Object-Oriented Approach Third Edition*, 3rd ed. United States of America: John Wiley & Sons, Inc., 2009.

[8] Hambling Brian & van Goethem Pauline, *User Acceptance Testing: A step-by-step guide*, BCS Learning and Development Ltd., 2013.

[9] Tridibesh Satpathy, *A Guide to the Scrum Body of Knowledge (SBOK GUIDE)*, 2016th ed. Phoenix: Scrum Study, 2016.

[10] Setiawan, E.I., Prakoso, H.K.B., Gunawan, T.P., Setyati E., Santoso, J., Aplikasi Mobile Untuk Memantau Body Mass Index Dengan Metodologi Scrum. *TEKNIKA, Volume 10(3), November2021. ISSN 2549-8037, EISSN 2549-8045. DOI: 10.34148/teknika.v10i3.405*

[11] R. M. R. Alan Dennis, Barbara Haley Wixom, *System Analysis & Design*, 6th ed. Hokoben: C. Baltzer AG Science Publishers, 2015.

[12] B. N. S. Elmasari R., *Fundamental of Database System Sixth Edition*. Addison-Wesley, 2011.

[13] Binus University, "Jenis Aplikasi Oracle," *Binus University*, 2020.