Survei Dampak Penggunaan Integrasi Berkelanjutan dalam Perusahaan Pengembangan Perangkat Lunak

Kharisma Monika Dian Pertiwi[1], Ana Tsalitsatun Ni'mah [2], Siti Rochimah[3]

***Abstract*—Continuous Integration (CI) is a software development technique adopted from the agile method. CI is widely used by software development companies, so there is a need for research to determine the impact of using CI in the software development industry. This study aims to analyze the impact of the use of CI on software and software development companies that are being developed. This research applies the Systematic Literature Review (SLR) research method. This study has two Research Questions, namely RQ, (1) "What is the impact of using Continuous Integration in software development?" (2) "What is the effect of using Continuous Integration on the company?". The impact of the use of CI was identified by conducting a literature search for CI which was published in 2012 until 2018. Literature search was conducted on the IEEE Xplore and Science Direct. From the search, a total of 6,514 literature regarding CI is found. Then, a screening process is carried out based on inclusion criteria, exclusion criteria, and quality assessment of literature. After screening, 14 literature were selected. The selected literature met the specified criteria and could represent to determine the impact of using CI. Out of the 14 selected literatures, 13 literatures were able to answer the two research questions. Based on the SLRs that have been done, it is shown that the use of CI in software development can have good and bad effects on software and software development companies.**

***Intisari*—Integrasi Berkelanjutan (*Continuous Integration*/CI) adalah teknik pengembangan perangkat lunak yang diadopsi dari metode *agile*. CI banyak digunakan oleh perusahaan pengembangan perangkat lunak, sehingga perlu adanya penelitian untuk mengetahui dampak penggunaan CI dalam industri pengembangan perangkat lunak. Makalah ini bertujuan untuk menganalisis dampak penggunaan CI terhadap perusahaan pengembangan perangkat lunak dan perangkat lunak yang sedang dikembangkan. Makalah ini menerapkan metode riset *Systematic Literature Review* (SLR). Dalam makalah ini terdapat dua *Research Question* atau RQ, yaitu (1) "Apa dampak menggunakan *Continuous Integration* dalam pengembangan perangkat lunak?" (2) "Apa efek menggunakan *Continuous Integration* pada perusahaan?". Dampak penggunaan CI diidentifikasi dengan melakukan pencarian literatur tentang CI yang terbit pada tahun 2012 hingga 2018. Pencarian literatur dilakukan pada IEEE Xplore dan Science Direct. Dari hasil pencarian ditemukan total 6.514 literatur mengenai CI. Selanjutnya, dilakukan penyaringan berdasarkan kriteria inkulusi, kriteria eksklusi, dan penilaian kualitas literatur. Setelah dilakukan penyaringan, terpilih 14 literatur. Literatur yang terpilih tersebut telah memenuhi kriteria yang ditentukan dan dapat mewakili untuk mengetahui dampak penggunaan CI. Dari 14 literatur yang terpilih, 13 literatur mampu menjawab kedua RQ. Berdasarkan SLR yang telah dilakukan, tampak bahwa penggunaan CI dalam pengembangan perangkat lunak dapat membawa dampak baik dan buruk bagi perangkat lunak dan perusahaan pengembangan perangkat lunak.**

Kata Kunci—Software Engineering, Continuous Integration, Agile, Systematic Literature Review.

I. PENDAHULUAN

Pada era teknologi ini, penggunaan perangkat lunak bukan hal yang asing dalam kehidupan sehari-hari seperti komunikasi, dokumentasi dan lain-lain. Perangkat lunak diperlukan untuk membantu manusia dalam mempercepat pekerjaan. Dengan menggunakan perangkat lunak, pekerjaan dinilai lebih eefektif dan efisien. Akibatnya, bisnis pengembangan perangkat lunak menjadi bisnis yang menarik selama beberapa dekade terakhir.

Pengembangan perangkat lunak tidak terlepas dari tahapan-tahapan yang harus dilakukan untuk pembuatan perangkat lunak. Telah banyak metode pengembangan perangkat lunak yang berkembang saat ini. Beberapa metode pengembangan perangkat lunak yang terkenal yaitu *waterfall, agile, dan Rapid Application Development* (RAD). Setiap metode pengembangan perangkat lunak memiliki kelebihan dan kekurangan masing-masing. Metode *waterfall* dilakukan secara berurutan sesuai tahapan, sehingga metode ini sulit beradaptasi dengan cepat terhadap perubahan *requirement*. Metode *agile* merupakan metode pengembangan perangkat lunak yang berdaptasi dengan cepat dengan perubahan.

Integrasi Berkelanjutan (*Continuous Integration*/CI) berasal dari *extreme programming*, yakni salah satu teknik terbaik dari metode pengembangan *Agile* [1]. Dalam CI, tim pengembang harus sering mengintegrasikan pekerjaannya, setidaknya setiap orang mengintegrasikan pekerjaannya setiap hari. Setiap integrasi akan dilakukan pembangunan secara otomatis untuk mendeteksi kesalahan secepat mungkin, sehingga dapat disimpulkan, CI merupakan alur kerja yang terintegrasi secara terus menerus dan bersifat otomatis [2].

Penggunaan CI sangat membantu dalam pengembangan perangkat lunak, terutama dalam mengurangi *bug* dan masalah integrasi antar anggota tim pengembang. CI muncul sebagai salah satu penemuan yang terbaik dalam rekayasa perangkat lunak otomatis. Sistem CI secara otomatis menggabungkan kompilasi, pembangunan, pengujian, dan pemasangan perangkat lunak [3].

Selama ini proses pembuatan dan pengembangan perangkat lunak tidak dapat direvisi, diuji, dan diintegrasikan secara efektif. Hal tersebut akan meningkatkan risiko kesalahan dalam pengembangan aplikasi [4]. Oleh karena itu, diperlukan sebuah

[1,2]*Mahasiswa, Program Studi Teknik Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember, Sukolilo, Surabaya, 60111 INDONESIA (email: kharisma.18051@mhs.its.ac.id)*

[3] *Dosen, Program Studi Teknik Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember, Sukolilo, Surabaya, 60111 INDONESIA*

penelitian yang membahas tentang dampak-dampak penggunaan CI pada perangkat lunak dan juga pada organisasi perusahaannya.

Referensi [5] menjelaskan cara CI diasimilasikan ke dalam organisasi. Hal ini dilakukan karena sangat sedikit pengetahuan tentang penerapan CI dalam organisasi. Dikatakan bahwa organisasi berubah melalui tahap asimilasi, peningkatan ambiguitas, memaksa pengembang untuk selalu memikirkan alternatif dan mengambil *trade-off*.

CI dapat didefinisikan sebagai suatu proses yang umumnya dilakukan secara otomatis dan terdiri atas langkah-langkah yang saling terhubung [6]. Selain segala bentuk otomatisasi, frekuensi dalam melakukan integrasi juga penting, agar tim pengembang dapat segera mendapatkan umpan balik, sehingga dengan CI, jika terjadi kegagalan dapat segera diatasi.

CI ini adalah bagian dari siklus pengembangan perangkat lunak. Perangkat lunak biasanya dikembangkan melalui fase yang berbeda dan oleh tim ahli [7]. Dalam siklus pengembangan perangkat lunak tradisional, integrasi perangkat lunak dilakukan sebagai langkah yang tergantung pada fase berikutnya, setelah implementasi perangkat lunak selesai. Hal ini akan menyebabkan integrasi perangkat lunak harus menunggu waktu yang lama dan dapat menyebabkan masalah saat integrasi, sehingga harus dilakukan perubahan terhadap aplikasi yang sudah selesai.

CI adalah salah satu pengembangan *Agile* yang terbaik dalam proses pengembangan perangkat lunak. Selama beberapa tahun terakhir telah banyak penelitian tentang CI, baik mengenai *tools*, maupun mengenai penerapannya terhadap perangkat lunak. Namun, sampai saat ini masih belum banyak, bahkan belum ada, *Systematic Literature Review* (SLR) tentang dampak dan manfaat penerapan CI terhadap perangkat lunak dan organisasi pengembangnya. Oleh karena itu, makalah ini menerapkan metode SLR terhadap penelitian-penelitian mengenai CI. Pencarian literatur dilakukan menggunakan dua perpustakaan digital, yaitu IEEE Xplore dan Science Direct dengan batasan tahun penelitian yaitu 2012 hingga 2018 dan hanya mengambil penelitian dengan kriteria jurnal dan konferensi.

Makalah ini mencoba untuk berkontribusi pada dunia penelitian yang mengumpulkan berbagai jenis penelitian yang menunjukkan manfaat dan dampak dari CI pada perangkat lunak dan perusahaan pengembangan perangkat lunak. Terdapat dua *research question* dalam makalah ini, yaitu (1) RQ1: "Apa dampak penggunaan *continuous integration* terhadap perangkat lunak yang dibangun?", dan (2) RQ2: "Apa dampak penggunaan *continuous integration* terhadap perusahan yang menggunakan teknik tersebut dalam pengembangan perangkat lunak?".

II. METODOLOGI

Metode penelitian yang dilakukan berdasarkan pada [8]. Tujuan dari penelitian ini untuk memberikan ulasan mengenai dampak dari penggunaan CI dalam pengembangan perangkat lunak. Dalam hal ini, dampak yang dimaksud adalah dampak terhadap perusahaan yang menggunakan teknik CI dalam pengembangan perangkat lunak, serta dampak terhadap perangkat lunak yang dikembangkan.

Gbr. 1 Metode penelitian *Systematic Literature Review*.

Penelitian ini terdiri atas beberapa tahapan yang ditunjukkan pada Gbr. 1. Tahap awal adalah tahap perencanaan, yaitu untuk menentukan ruang lingkup, RQ, dan langkah yang diperlukan untuk menjawab RQ. Tahap selanjutnya adalah pencarian literatur dalam dua sumber publikasi, proses penyaringan berdasarkan kriteria inklusi dan eksklusi yang dapat dilihat, penilaian kualitas, ekstraksi data, dan proses analisis. Tahap terakhir dari penelitian ini adalah tahap dokumentasi, yang meliputi penulisan makalah dan proses validasi.

Pada tahap perencanaan, langkah awal yang dilakukan adalah menentukan RQ yang akan menjadi bahasan. Langkah selanjutnya adalah mendefinisikan kata kunci yang digunakan untuk pencarian literatur, mendefinisikan kriteria inklusi dan eksklusi, dan mendefinisikan kriteria penilaian kualitas.

Setelah perencanaan, dilakukan pencarian literatur pada dua perpustakaan digital (IEEE Xplore dan ScienceDirect) berdasarkan kata kunci yang telah didefinisikan. Setelah literatur-literatur ditemukan, selanjutnya dilakukan pemilihan literatur-literatur yang sesuai dengan kriteria inklusi dan eksklusi. Selanjutnya, dari literatur-literatur yang terpilih tersebut dilakukan penilaian kualitas. Setelah melalui tahap penyaringan tersebut, dilakukan proses ekstraksi data untuk mencari informasi yang diperlukan dalam penelitian ini. Analisis data digunakan untuk menjawab RQ yang telah ditentukan.

Tahap akhir adalah mendokumentasikan kegiatan dalam penelitian ini. Pada tahap ini dilakukan pembuatan laporan yang disusun secara sistematis. Tahap selanjutnya adalah penilaian kualitas laporan. Revisi dan improvisasi akan dilakukan dalam tahap ini.

Pada penelitian ini, terdapat dua RQ yang akan dieksplorasi, dianalisis, dan disimpulkan mengenai penggunaan CI dalam pengembangan perangkat lunak. Yang pertama mengenai dampak penggunaan CI terhadap perangkat lunak yang dibangun, dan yang kedua adalah dampak penggunaan terhadap perusahan yang menggunakan teknik tersebut dalam pengembangan perangkat lunak.

TABEL I
HASIL PENCARIAN BERDASARKAN KATA KUNCI

Nomor Kata Kunci	IEEE Xplore	Science Direct	Sub-Total
1	2.002	3.002	5.004
2	147	321	468
3	28	8	36
4	3	0	3
5	301	317	618
6	23	3	26
7	7	0	7
8	63	270	333
9	8	7	15
10	2	2	4
Total			6.514

Berdasarkan pertanyaan-pertanyaan tersebut, dibuat daftar kata kunci yang digunakan dalam pencarian literatur. Pencarian dilakukan pada dua perpustakaan digital, yaitu IEEE Xplore (www.ieeexplore.ieee.org) dan Science Direct (www.sciencedirect.com). Perpustakaan digital tersebut dipilih karena dapat diakses dan diunduh secara gratis menggunakan akun institusi. Daftar kata kunci yang digunakan adalah sebagai berikut.

- *continuous integration*,
- *impact continuous integration*,
- *impact continuous integration software development*,
- *impact continuous integration software development companies*,
- *effect continuous integration*,
- *effect continuous integration software development*,
- *effect continuous integration software development companies*,
- *influence continuous integration*.

Kata kunci tersebut digunakan untuk pencarian literatur. Pencarian literatur dilakukan pada kedua perpustakaan digital dengan menggunakan filter tahun terbit antara tahun 2012 hingga 2018. Jumlah literatur hasil pencarian berdasarkan kata kunci tersebut diperlihatkan pada Tabel I. Total literatur yang dapat ditemukan berdasarkan kata kunci adalah 6.514. Literatur paling banyak menggunakan kata kunci *continuous integration*, dengan jumlah literature 5.004.

Selanjutnya dilakukan pemilihan literatur berdasarkan kriteria inklusi dan eksklusi. Kriteria inklusi dan eksklusi ditunjukkan pada Tabel II. Pada fase inklusi pertama, dilakukan pemilihan literatur berdasarkan judul. Dalam hal ini, dipilih judul literatur yang berkaitan dengan tujuan penelitian ini. Pada fase inklusi kedua, dilakukan pemilihan literatur dengan cara membaca intisari dari literatur tersebut. Dengan membaca intisari, dapat diketahui literatur yang sekiranya berkaitan dengan tujuan penelitian ini. Selanjutnya pada fase ketiga, dilakukan inklusi dengan membaca literatur secara keseluruhan. Dipilih literatur yang dapat menjawab RQ. Jumlah literatur setelah dilakukan pemilihan literatur berdasarkan kiteria inklusi dan eksklusi ditunjukkan pada Tabel III.

TABEL II
KRITERIA INKLUSI DAN EKSKLUSI YANG DIGUNAKAN DALAM PENELITIAN

Kriteria Inklusi	Kriteria Ekslusi
Literatur tentang *Continuous Integration*	Literatur bukan tentang *Continuous Integration*
Literatur dipublikasikan pada tahun 2012-2018	Literatur tidak dipublikasikan pada tahun 2012-2018
Literatur ditulis menggunakan bahasa Inggris	Literatur tidak ditulis menggunakan bahasa inggris
Tidak ada literatur dengan judul yang sama	Literatur memiliki judul yang sama

TABEL III
JUMLAH LITERATUR DAN HASIL PENYARINGAN

Fase	Detail	IEEE Xplore	Science Direct	Sub-Total
1	Ekslusi berdasarkan judul	125	22	147
2	Ekslusi berdasarkan abstrak	20	11	31
3	Ekslusi dengan membaca keseluruhan literatur	7	7	14

TABEL IV
FAKTOR PENILAIAN KUALITAS

Nomor QA	Isi *Quality Assesment* (QA)
QA1	Apakah literatur menyediakan informasi mengenai dampak *continuous integration*?
QA2	Apakah literatur menjawab pertanyaan penelitian 1
QA3	Apakah literatur menjawab pertanyaan penelitian 2

Hasil akhir dari tahapan pemilihan literatur berdasarkan kriteria inklusi dan eksklusi adalah terdapat 14 literatur yang akan menjadi panduan utama untuk menjawab RQ dalam penelitian ini. Literatur-literatur tersebut adalah sebagai berikut.

- *Modeling Continuous Integration Practice Differences in Industry Software Development* [9].
- *Continuous Software Engineering: A Roadmap and Agenda* [6].
- *A Standard-based Framework to Integrate Software Work in Small Settings* [10].
- *A Quality Framework for Software Continuous Integration* [7].
- *Modeling and Measuring Attributes Influencing DevOps Implementation in An Enterprise Using Structural Equation Modeling* [11].
- *The Continuity of Continuous Integration: Correlations and Consequences* [12].
- *Cinders: The Continuous Integration and Delivery Architecture Framework* [13].
- *Applying Continuous Integration for Reducing Web Applications Development Risks* [4].
- *Design and implementation of continuous integration scheme based on Jenkins and Ansible* [14].
- *Continuous integration processes for modern client-side web applications* [15].

TABEL V
TABEL HASIL PENILAIAN KUALITAS LITERATUR

Nomor Referensi	Tahun Publikasi	Tipe Paper	Jawaban RQ	*Quality*			*Score*			
				QA1	QA2	QA3	QA1	QA2	QA3	*Count*
[2]	2012	*Conference*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[3]	2016	*Conference*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[4]	2015	*Conference*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[6]	2017	*Journal*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[7]	2015	*Conference*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[11]	2014	*Journal*	RQ2	Y	N	Y	1	0	1	2
[12]	2017	*Journal*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[13]	2017	*Journal*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[14]	2017	*Journal*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[15]	2017	*Journal*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[16]	2018	*Conference*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[17]	2017	*Conference*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[18]	2014	*Conference*	RQ1, RQ2	Y	Y	Y	1	1	1	3
[19]	2014	*Conference*	RQ1, RQ2	Y	Y	Y	1	1	1	3

- *Implementing Continuous Integration towards rapid application development* [2].
- *Usage, costs, and benefits of continuous integration in open-source projects* [3].
- *Implementation of continuous integration and automated testing in software development of smart grid scheduling support system* [16].
- *Supporting continuous integration by mashing-up software quality information* [17].

Selanjutnya, terhadap 14 literatur tersebut dilakukan penilaian kualitas literatur. Terdapat tiga faktor yang digunakan untuk penilaian kualitas, yang ditunjukkan pada Tabel IV. Hasil penilaian kualitas berdasarkan ketiga faktor tersebut disajikan pada Tabel V. Berdasarkan penilaian kualitas tersebut, tampak bahwa rata-rata semua penelitian mendapat skor 2,93. Hanya satu makalah memiliki skor kurang dari 2,00, dan 13 makalah memiliki nilai 3,00.

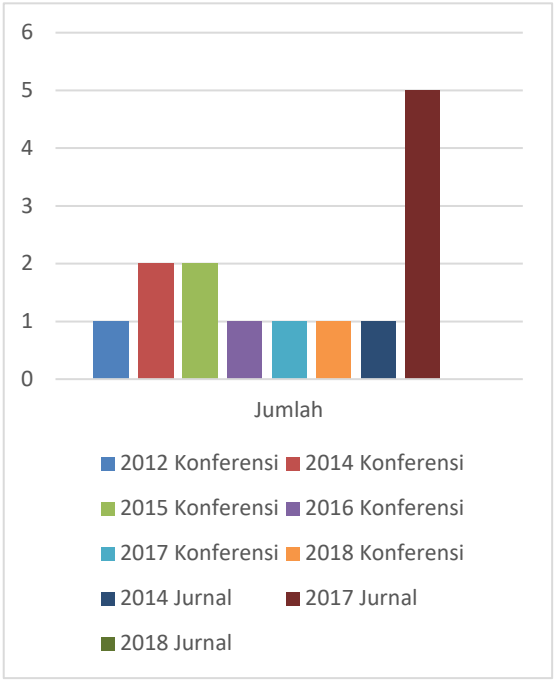
Semua literatur utama yang digunakan dalam penelitian ini, diterbitkan antara tahun 2012 hingga 2018. Distibusi literatur berdasarkan tahun terbit ditunjukkan pada Gbr. 2. Penilaian kualitas literatur berdasarkan tahun terbit disajikan pada Tabel VI. Hasil penilaian kualitas merupakan nilai total kualitas literatur di tahun tersebut dibagi jumlah literatur di tahun tersebut.

III. HASIL DAN PEMBAHASAN

Pada bagian ini dibahas hasil analisis untuk menjawab RQ1 dan RQ2. Dalam menjawab RQ1 dan RQ2, digunakan 14 literatur utama sebagai acuan.

A. *RQ1: Apa Dampak Penggunaan Continuous Integration Terhadap Perangkat Lunak Yang Dibangun?*

Pada subbagian ini dilakukan analisis mengenai dampak penggunaan CI terhadap perangkat lunak yang dibangun. CI adalah teknik untuk menghilangkan diskontinuitas tahap pengembangan dan pemasangan. Hubungan yang lebih kuat antara pengembangan dan pemasangan diperlukan untuk memastikan kesalahan terdeteksi dan diperbaiki sesegera mungkin. Sebagai hasilnya, kualitas dan ketahanan perangkat lunak meningkat [6].



Gbr. 2 Distribusi literatur berdasarkan tahun terbit.

Dalam pengembangan perangkat lunak, kualitas perangkat lunak berhubungan dengan kebutuhan fungsional dan nonfungsional. Dalam metode pengembangan perangkat lunak, *Agile*, jaminan kualitas didefinisikan sebagai pengembangan perangkat lunak yang dapat merespons perubahan setiap kali klien mengharuskannya untuk berubah. Integrasi berkelanjutan mengurangi risiko integrasi perangkat lunak, meningkatkan proses rekayasa perangkat lunak, dan meningkatkan kualitas perangkat lunak [7].

TABEL VI
KUALITAS LITERATUR BERDASARKAN TAHUN TERBIT

Tahun Publikasi	Rata-Rata Quality Score	Jumlah Penelitian	IEEE Xplore	Science Direct
2012	3,00	1	1	-
2014	2,67	3	2	1
2015	3,00	2	1	1
2016	3,00	1	1	-
2017	3,00	6	1	5
2018	3,00	1	1	-

Integrasi dapat dilakukan lebih mudah di dalam pengembangan perangkat lunak [2]. Jumlah pekerjaan untuk mengintegrasikan kode lebih rendah karena kode semakin sering diperiksa ke *server build* untuk kompilasi [2]. Bahkan integrasi perangkat lunak dan pengujian integrasi dapat mengurangi lebih dari 40% keseluruhan biaya proyek [7]. CI dan pengujian otomatis meningkatkan kualitas kode dan mengurangi risiko kesalahan perangkat lunak [16]. CI membantu untuk menghindari penyebaran informasi yang tidak konsisten pada pengembangan perangkat lunak [17].

CI membantu menangkap *bug* sebelumnya. CI juga memungkinkan dijalankannya pengujian secara *online* (*cloud*). CI membuat integrasi lebih mudah dan menghabiskan lebih sedikit waktu *debugging* [3]. CI yang diterapkan pada sistem berbasis web juga sangat bermanfaat. Kesalahan dan cacat aplikasi web dapat dideteksi, sehingga perbaikan dapat dilakukan segera. Dengan menggunakan CI pada sistem berbasis web, risiko pengembangan web yang tinggi dapat dikurangi [4]. Kesalahan dapat dideteksi sejak tahap awal karena setiap *commit* baru akan diverifikasi oleh server CI. Ketika kualitas perangkat lunak ditingkatkan, ini berarti *Quality Assurance* (QA) ditingkatkan secara akurat [15].

CI juga membantu meningkatkan kematangan proyek perangkat lunak, memberikan peningkatan proses *lean*, meningkatkan kualitas layanan perangkat lunak, dan mempromosikan evolusi sistem perangkat lunak berkualitas tinggi melalui pengoperasian dan pemeliharaan konstruksi mekanika [14].

Namun, penggunaan CI juga dapat menyebabkan penurunan kualitas perangkat lunak [12]. Hal tersebut dapat terjadi karena CI mengharuskan pengembang cepat beradaptasi dengan perubahan. Setiap pengembang dituntut untuk sering melakukan *commit*. Sebelum melakukan *commit*, pengembang harus memastikan bahwa kode benar dan dapat berjalan dengan baik, karena jika pengembang salah dalam melakukan *commit*, dapat terjadi konflik. Jika konflik terjadi, maka sistem tidak dapat digunakan dengan baik karena kesalahan kode dalam sistem.

Penggunaan CI juga dapat gagal karena beberapa faktor. Contohnya, jika pengujian pada tahap pemasangan gagal, maka pada tahap pemasangan juga otomatis gagal [9].

B. RQ2: Apa Dampak Penggunaan Continuous Integration Terhadap Perusahan Yang Menggunakan Teknik Tersebut Dalam Pengembangan Perangkat Lunak?

Selanjutnya, analisis mengenai dampak penggunaan CI terhadap perusahan yang menggunakan teknik tersebut dalam pengembangan perangkat lunak dilakukan. Perusahaan yang dimaksud dalam studi ini adalah perusahaan yang di dalamnya terdapat kegiatan pengembangan perangkat lunak. Banyak perusahaan yang sukses dalam menerapkan CI dalam pengembangan perangkat lunak [12]. Perusahaan perangkat lunak harus mengembangkan kemampuan perasanya, harus dapat menganalisis permintaan klien, dan harus dengan cepat beradaptasi dengan perubahan permintaan klien [6].

CI menghasilkan laporan perkembangan perangkat lunak secara cepat. Manajemen perusahaan akan menghargai fitur ini karena manajemen akan dapat melihat produk dan perubahan secara bertahap [2]. CI juga menghasilkan laporan perkembangan lebih sering sehingga perusahaan dapat mengontrol perubahan secara bertahap [3].

Ukuran organisasi dapat dilihat dari banyaknya pengembang yang menjadi anggota dari organisasi tersebut. Organisasi juga berhubungan dengan penggunaan CI [12]. Dalam hal ini, ukuran organisasi berhubungan erat dengan CI. Selain itu, ukuran organisasi juga dapat memengaruhi cara kerja, yang kerja dalam hal ini berhubungan dengan CI dan pengembangan perangkat lunak. Bagi perusahaan yang telah menggunakan CI , tingkat keseringan melakukan CI tidak berarti bahwa setiap pengembang juga sering melakukan *commit*. Semakin banyak pengembang yang terlibat dapat meningkatkan laju *commit*, tetapi jika satu orang saja melakukan kesalahan dalam *commit*, hal tersebut akan berdampak terhadap semua pengembang. Jika pengembang jarang melakukan *commit*, itu berarti bahwa terdapat banyak perubahan dalam kode. Hal tersebut dapat menyebabkan konflik jika dilakukan penggabungan dengan kode pengembang lain. Selain itu, banyaknya perubahan juga berdampak pada waktu yang dibutuhkan untuk pengujian dan pemasangan.

Permasalahan tersebut yang terjadi pada perusahaan, tim pengembang tidak dapat mengikuti CI. Manfaat CI dapat dirasakan oleh tim pengembang dari perusahaan jika tim pengembang mengubah kebiasaan dalam melakukan pengembangan perangkat lunak. CI mengharuskan setiap individu untuk sering melakukan kode, tidak untuk melakukan kode rusak, memperbaiki kerusakan segera, menulis tes otomatis, semua tes tertulis dan inspeksi harus lulus, menjalankan *build* pribadi, dan menghindari kode yang rusak [7].

Bagi perusahaan yang menerapkan integrasi berkelanjutan mengharuskan jika perubahan dalam sistem, maka pengujian otomatis harus dilakukan pada bagian kecil yang ditambahkan ke sistem. Selain tes otomatis, bagian kecil yang ditambahkan ke sistem juga harus dibangun secara otomatis, sehingga tim pengembang yang menulis kode dapat segera mendapatkan umpan balik, sehingga penggunaan integrasi berkelanjutan dapat membantu tim pengembangan untuk mengidentifikasi *bug* [7]. Selain itu, pengguna dapat dengan cepat mencoba versi yang diperbarui, sehingga dapat membangun kepercayaan dari *stakeholder* [4].

Faktor teknis bukanlah faktor yang penting dalam pengembangan perangkat lunak. Dalam pengembangan perangkat lunak, faktor yang paling penting adalah hubungan antar manusia itu sendiri. Faktor manusia penting untuk

TABEL VII
PERBANDINGAN SEBELUM DAN SETELAH MENGGUNAKAN CI

Kriteria	Sebelum CI	Setelah CI
Waktu Pengembangan	Lama karena tim pengembang harus fokus ke pengembangan semua kebutuhan	Singkat karena tim pengembang fokus pada satu kebutuhan
Waktu Pengujian	Lama, karena pengujian dilakukan terhadap keseluruhan sistem	Singkat karena pengujian dilakukan setiap 1 fitur selesai pengkodean
Frekuensi commit	jarang	sering
Jumlah Bugs	Banyak, karena pengujian dilakukan pada keseluruhan sistem	sedikit , karena setiap fitur diuji dan diperbaiki setelah selesai, tidak perlu menunggu keseluruhan sistem selesai
Waktu Pemasangan	Setelah sistem keseluruhan selesai dan pengujian selesai	Setelah menyelesaikan 1 kebutuhan dan mengujinya
Dokumentasi	Dokumentasi dibuat setelah pekerjaan selesai	Setiap kebutuhan didokumentasikan. Dokumentasi menggunakan alat otomatis yang dapat menghasilkan data statistik.
Manajemen perubahan	Perubahan hanya diterima setelah proses dan persetujuan yang panjang. Perubahan diperkenalkan melalui rilis baru	Perubahan diterima setiap saat hanya dengan menambahkan kebutuhan baru oleh klien kepada tim analisis bisnis dan tim pengembangan perangkat lunak
biaya	biaya adalah waktu dan pengerjaan yang dilakukan secara manual dalam pengembangan dan integrasi perangkat lunak	biaya didapat dari server dan alat yang digunakan dalam continuous integration
Ukuran perusahaan	Besar, karena semua pekerjaan dilakukan secara manual	Kecil, karena beberapa pekerjaan dimudahkan oleh alat otomatis

mencapai hasil yang konsisten dan selaras dengan strategi dan nilai-nilai organisasi [10].

Sistem CI membantu para anggota proyek untuk memfokuskan berbagai sumber pada isu-isu kunci, sehingga memperpendek waktu pengembangan dan meningkatkan kualitas perangkat lunak. Hal ini memberikan perusahaan keuntungan yang lebih [14]. Sistem CI juga meningkatkan kerja kolaborasi dalam tim pengembangan, sehingga berdampak pada kemajuan perusahaan [15]. Selain itu, koordinasi dalam perusahaan menjadi lebih terarah dengan membagi tugas-tugas pengembangan perangkat lunak ke dalam kelompok kecil [16]. CI juga mendorong kesadaran tim untuk bekerja sama dalam pengembangan perangkat lunak [17]. Perbandingan sebelum penggunaan CI dan setelah penggunaan CI diperlihatkan pada Tabel VII.

IV. KESIMPULAN

Makalah ini telah mengumpulkan literatur tentang penggunaan CI yang bertujuan untuk mendapatkan jawaban atas RQ: (1) ”Apa dampak menggunakan *Continuous Integration* dalam pengembangan perangkat lunak?” dan (2) ”Apa efek menggunakan *Continuous Integration* pada perusahaan?”. Hasil menunjukkan bahwa ada beberapa literatur yang tidak memiliki jawaban terhadap RQ yang menghasilkan skor rata-rata dari semua hasil penelitian, yaitu 2,93 dari nilai maksimal 3. Ini membuktikan bahwa proses pengumpulan data dan ekstraksi yang dilakukan telah berjalan dengan baik. Penggunaan CI dalam pengembangan perangkat lunak dapat membawa dampak baik dan buruk. Dampak baik timbul jika pengembang dapat mengikuti teknik CI, sedangkat dampak buruk terjadi jika pengembang tidak mengikuti CI sehingga menyebabkan banyak masalah pada perangkat lunak yang dibangun. Begitu juga dampak CI terhadap perusahaan, CI akan bermanfaat jika pengembang mampu mengubah kebiasaan dalam pengembangan perangkat lunak. Analisis dalam makalah ini berkontribusi pada pengetahuan tentang dampak dari CI, sehingga dapat dikembangkan untuk penelitian selanjutnya.

REFERENSI

[1] K. Schwaber dan M. Beedle, *Agile Software Development with Scrum*, New Jersey, USA: Prentice Hall, 2001.

[2] F.A. Abdul dan M.C.S. Fhang, "Implementing Continuous Integration Towards Rapid Application Development," *2012 International Conference on Innovation Management and Technology Research*, 2012, hal. 118-123.

[3] M. Hilton, T. Tunnell, K. Huang, D. Marinov, dan D. Dig, "Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects," *31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, hal. 426-437.

[4] S.-T. Lai dan F.-Y. Leu, "Applying Continuous Integration for Reducing Web Applications Development Risks," *10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, 2015, hal. 386-391.

[5] A. Eck , F. Uebernickel dan W. Brenner, "Fit for Continuous Integration: How Organizations Assimilate an Agile Practice," *20th Americas Conference on Information Systems, AMCIS*, 2014, hal. 1-11.

[6] B. Fitzgerald dan K.-J. Stol, "Continuous Software Engineering: A Roadmap and Agenda," *Journal of Systems and Software*, Vol. 123, hal. 176-189, 2017.

[7] S. Hamdana dan S. Alramouni, "A Quality Framework for Software Continuous Integration," *International Conference on Applied Human Factors and Ergonomics (AHFE 2015)*, 2015, hal. 2019-2025.

[8] B.A. Kitchenham dan S.M. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," Keele University and University of Durham, UK, Joint Tech. Report, 2007.

[9] D. Ståhl dan J. Bosch, "Modeling Continuous Integration Practice Differences In Industry Software Development," *Journal of Systems and Software*, Vol. 87, hal. 48-59, 2014.

[10] M.-L. Sanchez-Gordon, A. d. Amescua, R. V. O'Connor, dan X. Larrucea, "A Standard-Based Framework to Integrate Software Work in Small Settings," *Computer Standards & Interfaces*, Vol. 54 Part 3, hal. 162-175, 2017.

[11] V. Gupta, P. Kapur, dan D. Kumar, "Modeling and Measuring Attributes Influencing Devops Implementation in an Enterprise Using Structural Equation Modeling," *Information and Software Technology*, Vol. 97, hal. 75-91, 2017.

[12] D. Ståhl, T. Mårtensson, dan J. Bosch, "The Continuity of Continuous Integration: Correlations and Consequences," *Journal of Systems and Software*, Vol. 127, hal. 150-167, 2017.

[13] D. Ståhl dan J. Bosch, "Cinders: The Continuous Integration and Delivery Architecture Framework," *Information and Software Technology*, Vol. 83, hal. 76-93, 2017.

[14] W. Yiran, Z. Tongyang, dan G. Yidong, "Design and Implementation of Continuous Integration Scheme Based on Jenkins and Ansible," *International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2018, hal. 245-249.

[15] R. Tim, S. Tanachutiwat, M. Vukadinovic, H.-J. Schlebusch, dan H. Lichter, "Continuous Integration Processes for Modern Client-Side Web Applications," *International Electrical Engineering Congress (iEECON)*, 2017 hal. 1-4.

[16] J. Lu, Z. Yang, dan J. Qian, "Implementation of Continuous Integration and Automated Testing in Software Development of Smart Grid Scheduling Support System," *International Conference on Power System Technology*, 2014, hal. 2441-2446.

[17] M. Brandtner, E. Giger, dan H. Gall, "Supporting Continuous Integration by Mashing-Up Software Quality Information," *Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, 2014, hal. 184-193.