JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 8
No. 2
Agustus
2022

Evaluasi Metodologi CI/CD untuk Pengembangan Perangkat Lunak dalam Perkuliahan

Hapnes Toba*[1], Tjatur Kandaga Gautama[#2], Julio Narabel[*3], Andreas Widjaja[*4], Sendy Ferdian Sujadi[+5]

[#]Program Sarjana Teknik Informatika, [+]Program Sarjana Sistem Informasi, [*]Program Magister Ilmu Komputer
Fakultas Teknologi Informasi, Universitas Kristen Maranatha
Jl. Suria Sumantri No. 6,5 Bandung 40164
[1]hapnestoba@it.maranatha.edu
[2]tjatur.kandaga@it.maranatha.edu
[3]1879003@it.maranatha.edu
[4]andreas.widjaja@it.maranatha.edu
[5]sendy.fs@it.maranatha.edu

***Abstrak*— Saat ini sistem *Continuous Integration* (CI)/ *Continuous Delivery* (CD) merupakan standar baru dalam pengembangan perangkat lunak di industri. Sistem CI/CD merupakan langkah otomatisasi dari sebagian proses dalam pengembangan perangkat lunak. Ketika suatu sistem CI/CD digunakan oleh tim pengembang perangkat lunak maka akan diperoleh banyak data pemrosesan dan data hasil akhir dari proses CI/CD tersebut. Penelitian ini berupaya untuk mengevaluasi data yang terhimpun dalam sebuah sistem CI/CD dan diharapkan akan menemukan informasi yang bermanfaat sebagai umpan balik terhadap potensi sistem CI/CD dalam perkuliahan. Evaluasi riset dilakukan dengan metode survei pada kelas pilihan di semester ganjil tahun akademik 2021/22. Survei dimulai sejak masa ujian tengah semester sampai dengan akhir semester, yaitu pada saat mahasiswa peserta kelas mulai membuat sistem/ aplikasi guna memenuhi kelengkapan tugas besar mata kuliah. Adapun kelas yang dipilih tersebut adalah mata kuliah rekayasa perangkat lunak di program studi S-1 Teknik Informatika. Hasil survei menunjukkan bahwa mayoritas mahasiswa sangat antusias dan merasa penting untuk mendalami konsep CI/CD sebagai salah satu metode mutakhir pengembangan perangkat lunak.**

***Kata kunci*— *continuous integration, continuous delivery*, manajemen proyek, pengukuran kualitas, rekayasa perangkat lunak**

I. PENDAHULUAN

Berbagai teknologi pendukung pengembangan rekayasa perangkat lunak (RPL) yang banyak digunakan di industri saat ini berusaha memaksimalkan proses otomatisasi. Proses otomatisasi ini dikenal dengan sebutan *Continuous Integration (CI)/Continuous Delivery (CD)*. Seorang *programmer* akan menulis kode program kemudian menyimpannya (mem-*push*) ke dalam sistem pengelolaan kode sumber (*version control system*), seperti: CVS, Subversion, Git, dan sejenisnya.

Proses penyimpanan kode sumber ini akan memicu (men-*trigger*) rangkaian proses CI/CD yang meliputi proses pengecekan ketersediaan pustaka (*library*) yang dibutuhkan oleh program sehingga dapat diunduh secara otomatis jika belum tersedia, persiapan basis data, kompilasi kode sumber, pengujian perangkat lunak yang meliputi *unit test*, *integration test*, dan *user interface test*, dan terakhir proses instalasi program (*deployment*) ke dalam lingkungan penggunaan (*production environment*). Seluruh proses CI/CD ini akan dijalankan secara otomatis menggunakan perangkat khusus, sehingga pekerjaan *programmer* menjadi lebih efisien [1]-[3].

Proses otomatisasi dengan menggunakan sistem CI/CD ini akan menghasilkan data pemrosesan dan data hasil akhir. Dalam perjalanan sebuah proyek, tim pengembang perangkat lunak pada umumnya hanya akan memperhatikan data hasil akhir dan mengabaikan data pemrosesan [4]. Semua data tersebut, baik data hasil akhir maupun data pemrosesan, sebenarnya dapat dianalisis dan diolah untuk menghasilkan gambaran kualitas proses pengembangan perangkat lunak [5].

Berdasarkan pada latar belakang tersebut, maka dalam penelitian ini diangkatlah rumusan masalah sebagai berikut bagaimana persepsi mahasiswa terhadap pemanfaatan perangkat CI/CD dalam sesi perkuliahan di perguruan tinggi. Dari rumusan ini diharapkan dapat dirumuskan titik berat materi CI/CD dalam perkuliahan rekayasa perangkat lunak sehingga dapat menjadi bekal bagi mahasiswa dalam upaya peningkatan kualitas proyek perangkat lunak kelak pada saat bekerja [4].

II. METODE

Dalam bagian ini diuraikan pendekatan dan metode kerja yang mendukung penelitian.

A. Konteks Penerapan Continuous Integration (CI)/ Continuous Delivery (CD)

Continuous Integration (CI) adalah praktik pengembangan perangkat lunak yang mengharuskan pengembang perangkat lunak untuk menggabungkan kode sumber ke dalam *shared repository* beberapa kali setiap harinya. Setiap penggabungan kode sumber baru akan diverifikasi dengan proses pembangunan otomatis, sehingga tim pengembang dapat mendeteksi masalah yang timbul sedini mungkin. Dengan penggabungan secara rutin, kesalahan akan dapat diketahui dengan cepat dan penyebabnya dapat diketahui dengan mudah [1].

Continuous Delivery (CD) telah menjadi bagian yang tidak terpisahkan dalam setiap proses pengembangan perangkat lunak. Proses *Continuous Delivery* akan menolong tim pengembang perangkat lunak untuk mengetahui jawaban terhadap pertanyaan-pertanyaan yang seringkali berpengaruh dalam perjalanan proyek, yaitu [2]:

- Apakah semua komponen perangkat lunak dapat bekerja dan saling mendukung seperti seharusnya?
- Apakah kode sumber terlalu kompleks untuk dapat digabungkan?
- Apakah kode sumber sudah mengikuti standar pengkodean yang telah ditetapkan?
- Berapa banyak kode yang tercakup oleh pengujian otomatis?
- Apakah semua pengujian berhasil dilewati dengan baik setelah penambahan kode sumber terbaru?
- Apakah terjalin komunikasi timbal balik antara anggota tim kerja proyek dan target pengguna perangkat lunak yang sedang dikembangkan?

Pertanyaan-pertanyaan di atas menjadi penuntun yang digunakan dalam riset ini sebagai penjabaran dari rumusan masalah yang telah dituliskan dalam bagian terdahulu. Berpijak pada pertanyaan-pertanyaan itu pula, diusulkan sebuah kajian yang menghubungkan antara data dari sebuah sistem CI/CD dengan aspek pengelolaan proyek sebagaimana dirumuskan dalam *Project Management Body of Knowledge* (PMBOK) [4].

B. *Perangkat Pengolah CI/CD*

Sistem *Continuous Integration* (CI) pada umumnya merupakan pengatur orkestrasi yang melibatkan banyak *tools* dan menggabungkannya untuk mencapai tujuan otomatisasi proses pengembangan perangkat lunak. Berbagai kelompok *tools* beserta contoh-contohnya diantaranya adalah:

- *Source code control/version control*, merupakan *tools* untuk mengatur penggabungan kode sumber secara terpusat dari semua programmer yang terlibat. Juga berfungsi sebagai backup kode sumber, pengatur ketika terjadi bentrokan antar kode yang dikirim programmer yang berbeda, dan pengaturan versi kode sumber secara otomatis. Contoh *tools version control*: CVS, SubVersion, Git, Mercurial dll.
- *Server* atau *cloud* penyedia layanan *version control*, merupakan layanan di internet yang menyediakan *server* atau lingkungan *cloud* untuk pengaturan kode sumber. Contohnya adalah: Github, Gitlab, dan Bitbucket.
- Perangkat pembangun kode sumber menjadi kode tujuan sesuai bahasa program dan *framework* yang digunakan. Contohnya: Java *compiler*, C *compiler*, *C++ compiler*, C# *compiler*, Python *interpreter*, PHP *interpreter*, Make, Ant, Gradle, dan Maven.
- Perangkat untuk pengujian perangkat lunak, yang dapat dibagi lagi menjadi pengujian sebuah fungsi atomik (*unit testing*), pengujian penggabungan berbagai komponen perangkat lunak (*integration test*), dan pengujian antarmuka perangkat lunak (*user interface/user acceptance test*). Contohnya adalah: JUnit, NUnit, TestNG, Selenium, dan Mockito.
- Perangkat untuk melakukan *deployment*/instalasi perangkat lunak.

Perangkat untuk sistem CI sendiri cukup banyak, contohnya: TeamCity, Jenkins, Circle CI, Gitlab CI, Team Foundation Server, Travis CI, Go CD, Bamboo, CodeShip, Buddy Build, dan AWS Code Pipeline. Orkestrasi sistem CI pada umumnya mengikuti model *pipeline*, yaitu *output* dari sebuah *tools* akan menjadi *input* untuk *tools* berikutnya [3]. Oleh karena itu diperlukan adanya riset khusus untuk mengkombinasikan *tools* yang cocok berdasarkan metrik pengukuran yang diperlukan.

Penggunaan sistem CI/CD sudah merupakan *best practice* dalam industri perangkat lunak modern, dan hal ini membuka peluang untuk memaksimalkan pemanfaatannya. Secara umum sebuah sistem CI/CD yang digunakan oleh suatu tim pengembang perangkat lunak akan menghasilkan banyak data dari pemrosesannya [1], [5].

C. *Tahap Penelitian*

Penelitian ini merupakan bagian dari serangkaian tahapan riset dengan pembagian tahap yang dapat dilihat dalam Gambar 1. Secara khusus, dalam makalah ini disampaikan pembahasan untuk tahap keempat dan kelima, sedangkan untuk tahap pertama sampai ketiga diuraikan dalam makalah yang berbeda sebagaimana disampaikan dalam bagian III.C selanjutnya.

Gambar. 1 Tahapan riset

1. Menganalisis fitur dan instalasi berbagai *tools* CI/CD

Tahap pertama akan dilakukan survei dan analisis fitur dan kemampuan berbagai *tools* CI/CD yang tersedia. Setelah itu akan dipilih dua atau tiga *tools* yang memiliki fitur yang cukup lengkap dan dapat digunakan secara

bebas (*free & open source*) atau dapat dicoba terlebih dahulu (*free trial*) atau memiliki biaya lisensi yang terjangkau. Salah satu kriteria penting pemilihan, selain biaya juga harus memiliki arsitektur yang terbuka sehingga datanya dapat diambil untuk dilakukan analisis lanjutan.

2. Mempelajari cara mengambil data pemrosesan dari setiap *tools*, buat aplikasinya dan lakukan uji coba.

Mempelajari secara detail setiap *tools* dan bagaimana data pemrosesan dari *tools* tersebut dapat diambil. Setelah itu dibuatlah aplikasi awal yang akan mengambil data dari setiap *tool* dan menyimpannya ke dalam basis data untuk dipelajari. Lakukan pengujian pada aplikasi yang sudah dibuat.

3. Menganalisis dan pemrosesan data

Lakukan analisis dan evaluasi terhadap data yang diperoleh, kemudian tentukan data mana saja yang akan digunakan, yaitu data yang relevan dengan tujuan dan disesuaikan dengan metrik pengelolaan proyek yang terstandarisasi dalam PMBOK [4]. Kemudian dikembangkan aplikasi dengan modul untuk koleksi dan pemrosesan data.

4. Penyajian data

Data dengan skala dan domain yang berbeda akan sulit untuk bisa disajikan bersama-sama sehingga perlu dilakukan proses normalisasi terhadap data, khususnya untuk ditampilkan sebagai sebuah *dashboard* yang terintegrasi. Pada aplikasi tambahkan modul untuk penyajian data berupa tabel dan grafik sehingga lebih komunikatif dan mudah dibaca. Setelah itu lakukan kembali pengujian terhadap aplikasi tersebut.

5. Pembuatan laporan dan dokumentasi penelitian

Luaran penelitian ini berupa umpan balik melalui survei yang memberikan gambaran terhadap potensi dan kualitas proses pengembangan perangkat lunak dengan CI/CD yang dilakukan oleh tim pengembang, khususnya dalam konteks proyek akademik (tugas perkuliahan).

III. HASIL DAN PEMBAHASAN

A. *Eksplorasi Perangkat*

Analisis fitur dan instalasi berbagai *tools* CI/CD yang diperlukan untuk pengembangan perangkat lunak, dilaksanakan untuk perangkat berikut ini:

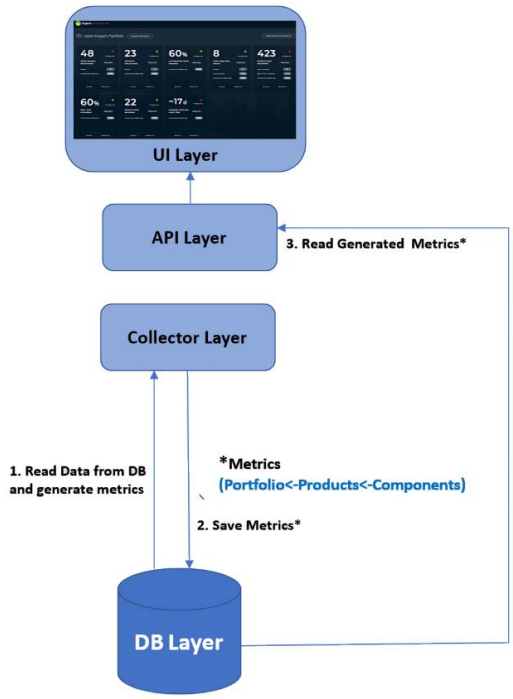
a. TeamCity: *server* otomasi pengembangan perangkat lunak CI/CD berbasis Java (lisensi oleh JetBrains) [5].
b. Docker: *server* berbasis virtualisasi sistem operasi untuk menyediakan layanan SAAS dan PAAS yang dapat digunakan untuk pengembangan perangkat lunak dalam konsep *container* (lisensi Docker, Inc.) [6].
c. Jenkins: *server* otomasi pengembangan perangkat lunak CI/CD berbasis Java (*open source*)
d. Github: sistem untuk kontrol *versioning* dan pengelolaan kode sumber (saat ini merupakan anak perusahaan Microsoft) [7].
e. Hygieia: sebuah sistem agregator untuk *pipeline* CI/CD yang memungkinkan pembuatan *dashboard* terhadap performa pengembangan perangkat lunak (*open source*, dikelola oleh Capital One) [8].

B. *Perancangan Interaksi dan Dashboard*

Orkestrasi sistem untuk *dashboard* CI/CD pada umumnya mengikuti model *pipeline* yang terdiri dari kombinasi berbagai sub-sub sistem. Dalam riset ini dirancang aliran data yang diimplementasikan sebagai *dashboard*. *Dashboard* tersebut memuat karakteristik pengelolaan proyek perangkat lunak, sehingga metrik yang diusulkan dalam *dashboard* pun akan menyesuaikan kebutuhan pengelolaan proyek dalam konteks akademik pendidikan tinggi, yang dibatasi pada aspek (4 dari 10 area dalam PMBOK), yaitu:

a. Ruang lingkup (*scope*)
b. Penjadwalan (*schedule*)
c. Komunikasi (*communication*)
d. Kualitas (*quality*)

Arsitektur sistem *dashboard* yang telah dirancang pada saat ini dapat dilihat dalam Gambar 2. Arsitektur ini diadaptasi dari sistem utilisasi *dashboard* Hygieia, dengan deskripsi sebagai berikut:



Gambar. 2 Arsitektur *dashboard* untuk pengelolaan proyek perangkat lunak

- *DB Layer*: digunakan untuk penyimpanan dan akses data. Setiap tipe metrik yang digunakan di dalam *dashboard* disimpan dalam bentuk koleksi data pada bagian *Collector Layer*, dapat dibangkitkan dan dituliskan di dalam basis data. Untuk mempermudah pengelolaan data, dalam riset ini akan digunakan basis data berkonsep ‘dokumen’ (format JSON), yang tidak bergantung pada relasi antar entitas, namun berbasis ‘layanan’ (*services*), sehingga lebih mudah

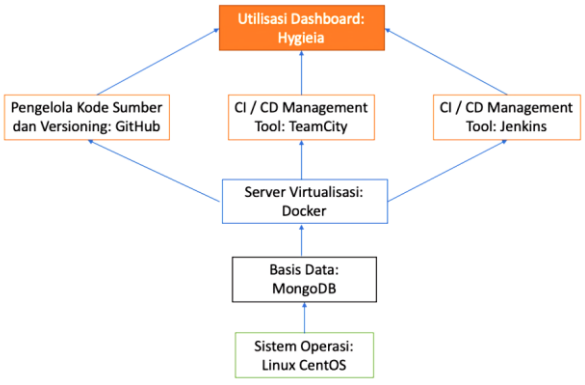
untuk aksesibilitas lintas *platform* (*cross-platform*), misalnya dengan MongoDB.

- *Collector Layer*: berisi mekanisme pembangkitan dan penyimpanan data dalam bentuk koleksi. Sebuah koleksi dapat menyediakan data-data mentah yang diperlukan untuk membangkitkan metrik pengukuran CI /CD. Akses data utama yang perlu dikelola adalah *time series* dan *summary*. *Time series* memberikan daftar kegiatan (aktivitas proyek) dalam rentang waktu tertentu (misalnya 90 hari), dan *summary* adalah nilai agregasi dari rentang waktu tersebut. Metrik dapat dibangkitkan untuk setiap anggota tim kerja (*portfolio*) dalam sebuah proyek (*product*) untuk tugas-tugas yang spesifik (*component*)
- *API Layer*: menyediakan antarmuka (*interface*) untuk menampilkan data melalui tampilan pengguna yang spesifik (UI).
- *UI Layer*: berisi tampilan-tampilan yang menyatakan metrik perkembangan proyek perangkat lunak dalam bentuk *dashboard*. Data yang ditampilkan menggunakan API yang mengedepankan Nama Anggota Tim, Proyek/Produk, Tipe Metrik, dll dalam bentuk yang intuitif.

C. Pipeline CI/CD dan Skema Basis Data

Dalam kaitan dengan metrik yang perlu ditampilkan untuk sebuah proyek perangkat lunak, perlu disusun kebutuhan data yang berasal dari *pipeline* (*tools*) yang digunakan. Secara garis besar tatanan *pipeline* dapat dilihat dalam Gambar 3. Proses pembangunan sistem CI/CD yang digunakan pada riset ini terdiri dari fase eksperimen, eksplorasi, dan pemanfaatannya. Metode pengujian sistem CI/CD yang dilakukan pada saat pengembangan tersebut adalah dengan menggunakan metode "*pipelining*" melalui proses "*build*" untuk melihat kebergunaannya dan kemudahannya [9].

Sistem CI/CD secara teknologi telah mendukung pula konsep *High Performance Computing* (HPC) dengan mengimplementasikan *Graphical Processing Units* (GPU) [10]. Definisi data yang untuk implementasi sistem CI/CD beserta kaitannya dengan aspek manajemen proyek (PMBOK) dapat dilihat dalam Tabel 1.



Gambar. 3 *Pipeline* CI/CD

TABEL I
DEFINISI DATA DALAM PERANCANGAN *DASHBOARD PIPELINE CI/CD* (ADAPTASI DARI [8] DENGAN PENAMBAHAN ASPEK PMBOK)

Aspek PMBOK	**Koleksi Data *Fields***	**Deskripsi**	***Tools* Penyedia**
Communication [11]	*apitoken*	Informasi hasil proses otentifikasi. Digunakan untuk mengakses repositori *pipeline tools* CI/CD.	GitHub
Communication	*authentication*	Model penyimpanan data privat untuk login	Lokal (*custom* Hygieia)
Communication	*cmdb*	Konfigurasi pengelolaan data dan layanan (*services*)	(opsional, misalnya *HP Service Manager*)
Communication	*dashboards*	Koleksi *widgets*, *collectors* dan komponen aplikasi yang merepresentasikan pengembangan sebuah proyek perangkat lunak ataupun yang telah siap digunakan.	Lokal (*custom* Hygieia)
Communication	*environment_status*	Informasi status (online atau offline) untuk sebuah *server* serta komponen dan lingkungannya yang spesifik.	Lokal (*custom* Hygieia)
Communication	*pipelines*	Informasi tentang detil *collector* untuk setiap anggota tim proyek.	Lokal (*custom* Hygieia)
Communication	*scope-owner*	Berisi anggota tim yang mengerjakan fitur-fitur dalam sebuah proyek perangkat lunak.	Jira, GitHub atau GitLab (*feature history*)
Communication	*services*	Produk atau layanan yang disediakan oleh sebuah *tool* yang terkoneksi dengan *dashboard*.	Lokal (*custom* Hygieia)
Communication	*team*	Anggota tim yang bertanggung jawab terhadap sebuah fitur dalam kurun waktu tertentu.	Jira, GitHub atau GitLab (*feature history*)
Quality [12]	*artifacts*	Hasil kompilasi (*binary object*), dan tersimpan dalam repositori.	*Compiler* dalam Jenkins atau

Aspek PMBOK	Koleksi Data *Fields*	Deskripsi	*Tools* Penyedia
			TeamCity
Quality	*builds*	Hasil eksekusi proses build (*binary object*).	Jenkins atau TeamCity
Quality	*code_quality*	Menunjukkan kualitas kode dalam kurun waktu yang spesifik. Termasuk hasil *unit test, functional test, manual acceptance tests* atau laporan *bugs*.	Jenkins atau TeamCity
Quality	*performance*	Informasi terkait dengan performa aplikasi/produk yang sedang dikembangkan.	Jenkins atau TeamCity
Quality	*test_results*	Merepresentasikan sebuah koleksi skenario pengujian yang telah dilakukan dalam proyek. Melibatkan misalnya: *unit test run, security scan, static analysis, functional tests, manual acceptance tests,* or laporan *bugs*.	Jenkins atau TeamCity
Scope & Schedule [13]	*collector_items*	Memberikan representasi koleksi unik dari sebuah sistem eksternal, misalnya Job dalam sebuah CI *tool*. Penyimpanan *key-value* untuk akses informasi bagi *dashboard*. Idealnya setiap koleksi memiliki deskripsi yang unik.	Semua *tools*
Scope & Schedule	*collectors*	Informasi *collector* yang telah terdaftar untuk diambil datanya oleh *dashboard*.	Semua *tools*
Scope & Schedule	*commits*	Hasil commit kode program untuk sebuah versi dalam repositori.	GitHub
Scope & Schedule	*components*	Bagian kode program yang pada akhirnya disatukan untuk sebuah proyek. Setiap komponen aplikasi memiliki perbedaan pada: repository sumber, *build job, deploy job,* dll.	Lokal (*custom* Hygieia)
Scope & Schedule	*environment_ components*	Informasi komponen yang dapat dikembangkan dalam sebuah lingkungan pemrograman.	Lokal (*custom* Hygieia)
Scope & Schedule	*feature*	Memberikan deskripsi fitur untuk sebuah komponen.	Jira, GitHub atau GitLab (*feature history*)
Scope & Schedule	*git_requests*	Memberikan informasi untuk repositori Git (*commits, issues, pull-request*, dll)	GitHub
Scope & Schedule	*incident*	Memberikan alasan munculnya insiden pada saat proses mengumpulkan data dengan layanan (API).	(opsional, misalnya *HP Service Manager*)
Scope & Schedule	*scope*	Berisi daftar fitur yang menjadi scope sebuah proyek perangkat lunak.	Jira, GitHub atau GitLab (*feature history*)
Scope & Schedule	*templates*	Sebuah koleksi template yang merepresentasikan sebuah proyek perangkat lunak, baik yang sedang dikembangkan atau yang telah dalam produksi.	Lokal (*custom* Hygieia)
Scope & Schedule	*feature_history*	Memberikan data historis untuk fitur yang sedang ramai (*trend*) dikembangkan dalam sebuah *content management system*.	Jira, GitHub atau GitLab (*feature history*)

D. Evaluasi Potensi Pemanfaatan CI/CD

Untuk mengevaluasi potensi pemanfaatan CI/CD dan *dasbboard*-nya dilakukan survei dalam kelas mata kuliah RPL Dasar di semester ganjil 2021/22. Peserta survei adalah para mahasiswa semester III sejumlah 39 orang mahasiswa. Tabel II memberikan daftar pertanyaan yang

diajukan untuk mengevaluasi potensi pemanfaatan perangkat CI/CD dari sudut pandang mahasiswa.

Survei dilakukan dengan memberikan pendapat berupa skor yang bernilai 1 sampai dengan 5, dengan: nilai 1 = sangat tidak setuju, 2 = tidak setuju, 3 = cukup setuju, 4 = setuju, dan nilai 5 = sangat setuju. Terdapat 10 pertanyaan, sebagaimana ditampilkan pada Tabel II, yang perlu diisi oleh mahasiswa selama periode survei.

Pertanyaan-pertanyaan survei dikaitkan pula dengan aspek pengelolaan proyek dalam PMBOK, yaitu: *communication*, *quality*, *scope*, dan *schedule*. Masing-masing ada lima pertanyaan yang mengarah pada aspek gabungan antara: *communication & quality* (terkait isu non-teknis), serta *scope & schedule* (terkait isu teknis).

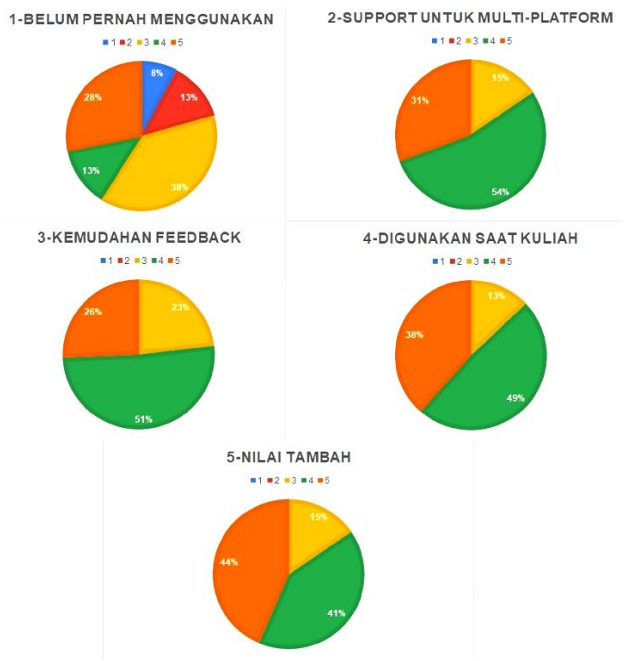
TABEL II
PERTANYAAN SURVEI POTENSI PEMANFAATAN CI/CD

No.	Pertanyaan	Aspek PMBOK
1.	Saya belum pernah menggunakan sistem CI/CD sebelumnya.	*Communication & Quality*
2.	Sistem CI/CD dapat menangani dengan baik berbagai *platform* aplikasi (*console/desktop/web*) [14].	*Communication & Quality*
3.	*Feedback* dari sistem CI/CD membantu dan mudah dipahami.	*Communication & Quality*
4.	Saya setuju/mendukung jika sistem /*environment* CI/CD diterapkan dalam kuliah Pemrograman dan RPL.	*Communication & Quality*
5.	Kemampuan menggunakan/ mempersiapkan sistem CI/CD akan menjadi nilai tambah saya dalam pekerjaan nanti.	*Communication & Quality*
6.	Menghubungkan *repository* kode sumber ke sistem CI/CD cukup mudah dilakukan [15].	*Scope & Schedule*
7.	Proses *build* aplikasi di dalam sistem CI/CD dapat berjalan dengan lancar [16].	*Scope & Schedule*
8.	Pembuatan dan proses eksekusi *unit testing* di dalam sistem CI/CD dapat berjalan dengan lancar [17].	*Scope & Schedule*
9.	Sistem CI/CD dapat menangani dengan baik berbagai bahasa pemrograman (C/C++/Java/Python/ HTML/PHP) [18].	*Scope & Schedule*
10.	Penggunaan sistem CI/CD akan dapat meningkatkan kinerja saya dalam membangun aplikasi.	*Scope & Schedule*

1) Isu Non-Teknis: kelompok pertanyaan mengenai *communication & quality* difokuskan pada konteks bagaimana sistem CI/CD dapat mempengaruhi pola pikir mahasiswa mengenai proses pengembangan RPL.

Dalam metodologi pengembangan RPL tradisional, seperti misalnya: *waterfall* [19], *Rapid Application Development* (RAD) [20], dan sejenisnya; terdapat pembagian proses yang terkesan kaku mulai dari tahap perencanaan (analisis), organisasi (implementasi), aksi (instalasi), dan kontrol (evaluasi). Setiap tahap harus dinyatakan tuntas, barulah kemudian dapat dilakukan perpindahan ke tahap selanjutnya atau melakukan iterasi di tahap yang sama. Dengan diperkenalkannya sistem CI/CD mahasiswa mulai memperlihatkan pemahaman tentang pentingnya proses penyesuaian secara cepat (*agile*) selama keseluruhan proses pengembangan perangkat lunak tanpa harus dibatasi oleh pengelompokan fase semata.

Hasil survei memperlihatkan superioritas CI/CD dalam hal non-teknis setelah diperkenalkan kepada mahasiswa (lihat rekapitulasi hasil survei dalam Gambar 4). Di awal semester, terdapat 79% mahasiswa yang belum pernah mendengar atau menggunakan sistem CI/CD. Hal ini ditunjukkan melalui gabungan jawaban pilihan 3 = cukup setuju (38%), 4 = setuju (13%), dan nilai 5 = sangat setuju (28%) untuk pertanyaan nomor 1. Namun seiring dengan berjalannya waktu mulai sejak periode ujian tengah semester, mahasiswa semakin fasih dan yakin akan manfaat sistem CI/CD. Hal ini ditunjukkan dengan isian survei untuk pertanyaan nomor 5, dengan semua mahasiswa memilih pilihan 3 = cukup setuju (15%), 4 = setuju (41%), dan nilai 5 = sangat setuju (44%).



Gambar. 4 Hasil survei potensi CI/CD untuk isu non-teknis

Mahasiswa bereksperimen mengerjakan potongan kode dan sistem sederhana dalam berbagai *platform*, yaitu: *console* (C++), *desktop* (Java), dan *web* (PHP). Masing-masing disertai dengan siklus lengkap: *development*, *commit*, *build*, *test*, *staging*, *deploy*, dan *production*. Mahasiswa ditugaskan secara berkelompok untuk mengerjakan serangkaian deskripsi fitur perangkat lunak sebagai bagian tugas besar mata kuliah.

Di awal proyek, setiap kelompok diminta untuk mengalokasikan pekerjaan (*staffing*), yaitu: pembagian *function*, *method* atau *class* yang dikembangkan. Setelah itu, dengan memanfaatkan perangkat CI/CD, antar anggota kelompok dapat memantau kemajuan proyek

melalui Github *code repository*, Jenkins & TeamCity *build*, dan *unit testing*. Dalam setiap tahapan proyek mahasiswa dapat saling berkolaborasi dan melakukan pengawasan versi (*versioning control*) untuk melengkapi potongan kode program hingga sebuah deskripsi tugas dituntaskan. Rangkaian deskripsi tugas yang dilakukan oleh mahasiswa dapat dilihat dalam Tabel III.

Dari hasil eksperimen ini, mahasiswa menyatakan keyakinannya bahwa sistem CI/CD mendukung pengembangan perangkat lunak secara *multi-platform* (pertanyaan nomor 2 dijawab dengan setuju = 54%, dan sangat setuju = 31%). Mahasiswa juga merasakan kemudahan saling memberikan *feedback* antar anggota tim kerja (pertanyaan nomor 3 dijawab dengan setuju = 51%, dan sangat setuju = 26%).

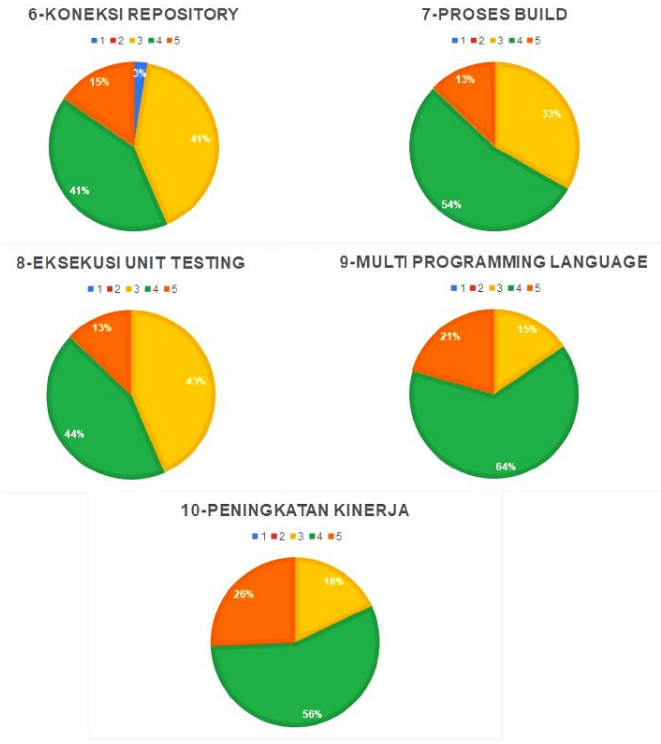
TABEL III
DESKRIPSI FITUR SAAT EKSPERIMEN

No.	Deskrispsi Fitur	Bahasa Pemrograman
1.	*1) Fibonacci Series*	C++ (*console*)
2.	*2) Prime Number Program* *3) Palindrome Program*	Java (*console*)
3.	*Address Book*	Java (*desktop*)
4.	*User administration and system login management: create, read, update, and read*	PHP (*web-based*)
5.	*Create, read, update, and read customer*	PHP (*web-based*)
6.	*Create, read, update, and read sales items*	PHP (*web-based*)
7.	*Create, read, update, and read order*	PHP (*web-based*)

Dengan dirasakannya manfaat di atas, mahasiswa memberikan usulan agar sistem CI/CD digunakan dalam skala yang lebih luas dalam sesi-sesi perkuliahan lainnya yang melakukan *programming* (pertanyaan nomor 4 dijawab dengan setuju = 49%, dan sangat setuju = 38%). Pada akhirnya mahasiswa pun merasa yakin bahwa sistem CI/CD akan dapat memberikan nilai tambah dalam pengembangan karirnya mendatang (pertanyaan nomor 5 dijawab dengan setuju = 41%, dan sangat setuju = 44%).

2) Isu Teknis: kelompok pertanyaan terkait *scope & schedule* difokuskan pada konteks bagaimana sistem CI/CD memiliki kemampuan pengelolaan proyek secara *cross-environment*. Pengembangan sistem dengan CI/CD tidak lagi dibatasi dengan bahasa pemrograman. Misalnya sebuah produk perangkat lunak berbasis web dapat merupakan kombinasi beberapa bahasa pemrograman, misalnya: antara sisi *front-end* (dengan AngularJS), dan sisi *back-end* (dengan PHP). Tanpa perlu disediakan perangkat yang berjalan terpisah, setiap komponen dapat dipantau sebagai satu proyek pada sebuah sistem CI/CD yang sama.

Hasil rekapitulasi isian survei untuk isu teknis ini dapat dilihat dalam Gambar 5. Dari hasil eksperimen, mahasiswa menyatakan bahwa koneksi dengan *repository* yang berbeda-beda tidak mengalami kesulitan (pertanyaan nomor 6 dijawab dengan setuju = 41%, dan sangat setuju = 15%). Meskipun bahasa pemrograman berbeda, sebuah sistem masih dengan relatif mudah diintegrasikan dalam sebuah sistem kompilasi (*build*) yang sama. Mayoritas mahasiswa dengan jawaban setuju = 54%, dan sangat setuju = 13%, mendukung kemudahan proses *build* yang disertai pengelolaan versi (*versioning*) dengan perangkat CI/CD pada pertanyaan nomor 7.



Gambar. 5 Hasil survei potensi CI/CD untuk isu teknis

Pengujian perangkat lunak secara *white-box* dengan memanfaatkan *unit testing* juga didukung dengan baik oleh sistem CI/CD. Mahasiswa merasakan kemudahannya dengan jawaban setuju = 44%, dan sangat setuju = 13% pada pertanyaan nomor 8. Kemampuan CI/CD untuk mengelola proyek dalam *multi-programming language* memberikan keyakinan pada mahasiswa bahwa proyek akan tetap berjalan dengan lancar meskipun memerlukan bahasa pemrograman lebih dari satu (pertanyaan nomor 9 dijawab dengan setuju = 64%, dan sangat setuju = 21%). Pada akhirnya mahasiswa juga merasa sangat yakin bahwa akan dicapai peningkatan kinerja dalam proses pengembangan perangkat lunak dengan dukungan sistem CI/CD (pertanyaan nomor 10 dijawab dengan setuju = 56%, dan sangat setuju = 26%).

IV. KESIMPULAN

Dalam penelitian ini telah berhasil dilakukan perancangan dan mekanisme penyimpanan data sebuah sistem pengelolaan proyek yang dilengkapi dengan sistem CI/CD. Melalui sistem CI/CD ini pengelolaan proyek lintas *platform* dan lintas bahasa pemrograman dapat dilakukan dengan lebih mudah, dan hasilnya dapat

langsung terlihat, tanpa harus menunggu selesainya suatu fase pengembangan perangkat lunak.

Serangkaian eksperimen yang melibatkan para mahasiswa peserta mata kuliah RPL telah membuktikan superioritas pendekatan CI/CD untuk pengelolaan proyek perangkat lunak. Hasil survei dari para mahasiswa juga menunjukkan keyakinan bahwa CI/CD sangat diperlukan sebagai salah satu *hard-skills* utama dalam bidang RPL dewasa ini. Selain sebagai suatu kebutuhan *hard-skills*, penguasaan CI/CD juga akan terkait erat dengan kemampuan pengelolaan proyek (*soft-skills*).

Hal ini terutama terlihat pada hasil survei yang memperlihatkan bahwa sistem CI/CD memberikan kemudahan *feedback*, yang terkait dengan komunikasi dan kualitas pengembangan perangkat lunak. Menjawab rumusan masalah yang diangkat dalam bagian pendahuluan, hasil survei memperlihatkan kebutuhan yang sangat tinggi bagi mahasiswa untuk menguasai cara pengaksesan repositori kode program, melakukan kompilasi (*build*), komunikasi (*feedback*) dan penyampaian (*delivery*) hasil kode melalui pengelolaan versi (*versioning*).

Sebagai saran pengembangan, hasil yang dipaparkan dalam makalah ini dapat diperluas pemanfaatannya untuk dapat digunakan dalam pengelolaan beberapa proyek sekaligus. Koneksi dengan sistem *cloud* [21] atau *container* [22] secara *live* dapat menjadi agenda riset selanjutnya, berikut potensi terkait pengelolaan tim kerja dalam proyek pengembangan perangkat lunak.

UCAPAN TERIMA KASIH / ACKNOWLEDGMENT

Riset ini terlaksana dengan dukungan pendanaan dari hibah LPPM Universitas Kristen Maranatha.

REFERENSI

[1] M. Aggarwal, *TeamCity: continuous integration & DevOps with Java and .NET*, Packt Publishing, 2018.

[2] (2021). Tutorialspoint website. [Online]. Available: https://www.tutorialspoint.com/continuous_integration/index.htm

[3] J. Lee, *Master Jenkins Course For Developers and DevOps*, Packt Publishing, 2017.

[4] P. M. Institute, *A guide to the project management body of knowledge (PMBOK guide)*, Pennsylvania: Project Management Institute, Publisher, 2017.

[5] R. Parashar, "Path to Success with CICD Pipeline Delivery," *International Journal of Research in Engineering, Science and Management*, vol. 4, no. 6, pp. 271-273, 2021.

[6] S. Garg and S. Garg, "Automated cloud infrastructure, continuous integration and continuous delivery using docker with robust container security," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 467-470.

[7] L. Yu, E. Alégroth, P. Chatzipetrou, and T. Gorschek, "Utilising CI environment for efficient and effective testing of NFRs," *Information and Software Technology*, vol. 117, 106199, 2020.

[8] (2021). Hygieia Documentation. [Online] Available: http://hygieia.github.io/hygieia/getting_started.html

[9] S. Ferdian, T. Kandaga, A. Widjaja, H. Toba, R. Joshua, and J. Narabel, "Continuous Integration and Continuous Delivery Platform Development of Software Engineering and Software Project Management in Higher Education," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 7, no. 1, 2021.

[10] A. Widjaja, T. K. Gautama, S. F. Sujadi, and S. R. Harnandy. "High Performance Computing Environment using General Purpose Computations on Graphics Processing Unit." *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 7, no. 2, 2021.

[11] A. Muñoz, A. Farao, J. R. C. Correia, and C. Xenakis, "P2ISE: Preserving Project Integrity in CI/CD Based on Secure Elements," *Information*, vol. 12, no. 9, pp. 357, 2021.

[12] H. Chassidim, D. Almog, and S. Mark, "Quality development (QDev) unit in a software engineering school," *World Transactions on Engineering and Technology Education*, vol. 16, no. 3, pp. 249-253, 2018.

[13] F. Zampetti, V. Carmine, S. Panichella, G. Canfora, H. Gall, and M. Di Penta, "An empirical characterization of bad practices in continuous integration," *Empirical Software Engineering*, vol. 25, no. 2, pp. 1095-1135, 2020.

[14] O. F. Pablo, M. David, D. C. Duma, R. Elisabetta, J. Gomes, and S. Davide, "Software Quality Assurance in INDIGO-DataCloud project: a converging evolution of software engineering practices to support European Research e-Infrastructures," *Journal of Grid Computing*, vol.18, no. 1, pp. 81-98, 2020.

[15] H. Schulz, and A. van Hoorn, "Representative Load Testing in Continuous Software Engineering: Automation and Maintenance Support," *Software Engineering Lecture Notes in Informatics (LNI)*, Gesellschaft für Informatik, Bonn: Springer, 2020, https://doi.org/10.18420/SE2020_46.

[16] C. Feld, M. Geimer, M-A. Hermanns, P. Saviankou, A. Visser, and B. Mohr, "Detecting Disaster Before It Strikes: On the Challenges of Automated Building and Testing in HPC Environments," In *Tools for High Performance Computing 2018/2019*, pp. 3-26. Springer, Cham, 2021.

[17] A. C. Arroyo, M. R. Montes, and J. D. S. Quilis, "A pilot experience with software programming environments as a service for teaching activities," *Applied Sciences*, vol. 11, no. 1, pp. 341, 2021.

[18] S. Eismann, J. Scheuner, E. Van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, "Serverless applications: Why, when, and how?," *IEEE Software*, vol. 38, no. 1, pp. 32-39, 2020.

[19] K. Bhavsar, V. Shah, and S. Gopalan, "Scrumbanfall: an agile integration of scrum and kanban with waterfall in software engineering," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, no. 4, pp. 2075-2084, 2020.

[20] G.A. Dhanush, "Develop a Scalable and Serverless Client-based Application using Agile Methodology," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 11, pp. 5372-5379, 2021.

[21] S. M. Mohammad, "Streamlining DevOps automation for Cloud applications," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 6, no. 4, pp. 955-959, 2018.

[22] A. M. Shama and D. W. Chandra, "Implementasi Static Application Security Testing Menggunakan Jenkins CI/CD Berbasis Docker Container Pada PT. Emporia Digital Raya," *JURNAL ILMIAH INFORMATIKA*, vol. 9, no. 2, pp. 95-99, 2021.