PENILAIAN OTOMATIS PADA MEDIA PEMBELAJARAN PEMROGRAMAN *ONLINE* DENGAN PENDEKATAN *CLONE CODE DETECTION*

Bayu Priyambadha[1)], **Fajar Pradana**[2)]

[1, 2)] *Fakultas Ilmu Komputer, Universitas Brawijaya Malang*
Email : bayu_priyambadha@ub.ac.id[1)], fajar.p@ub.ac.id[2)]

Abstrak

Salah satu kompetensi utama yang harus dimiliki oleh lulusan dari jurusan atau program studi bidang keilmuan komputer adalah kemampuan programming (membuat program). Berbagi informasi untuk meningkatkan kualitas pembelajaran telah dilakukan di banyak kampus di Indonesia. Salah satunya adalah penerapan media pembelajaran online atau disebut juga sebagai Learning Management System (LMS). Permasalahan yang sering muncul dalam LMS adalah proses penilaian. Penilaian pemrograman merupakan tantangan tersendiri dalam pembelajaran pemrograman online berbasis LMS. Metode deteksi klon kode memiliki prinsip yang sama dengan proses penilaian kode program. Penerapa metode tersebut menghasilkan hasil yang sangat memuaskan dilihat dari waktu yang dibutuhkan untuk melakukan penilaian pada kode yang beragam.

Kata kunci: *Penilaian otomatis, Learning Management System, deteksi klon, klon semantik.*

1. Pendahuluan

Bidang keilmuan komputer adalah bidang yang berkembang sangat pesat. Sentuhan keilmuan komputer sangat dibutuhkan di berbagai bidang. Bidang komputer tidak lagi menjadi disiplin ilmu tunggal, melainkan sudah menjadi satu rumpun keilmuan. Bidang keilmuan komputer terbagi menjadi lima sub disiplin, yaitu, *Computer Engineering*, *Computer Science*, *Information Systems*, *Information Technology* dan *Software Engineering* [1]. Dimana, terdapat tiga disiplin ilmu yang syarat dengan pembangunan software, yaitu *Computer Engineering*, *Computer Science* dan *Software Engineering* [1]. Saat ini sudah banyak dibuka jurusan atau program studi yang terkait dengan keilmuan komputer di Indonesia.

Salah satu kompetensi utama yang harus dimiliki oleh lulusan dari jurusan atau program studi bidang keilmuan komputer adalah kemampuan *programming* (membuat program). Membuat program dilakukan dengan menuliskan kode program lalu melakukan kompilasi dan menjalankan program. Siswa atau mahasiswa dituntut untuk tidak sekedar memahami sintak program melainkan harus juga memahami logika atau alur program. Bagi siswa atau mahasiswa pemula, untuk menguasai kompetensi *Programming* ini harus sering berlatih dalam menuliskan kode program. Kendala terbesar adalah siswa atau mahasiswa sering kali malas dalam berlatih membuat kode program. Sehingga, terdapat siswa atau mahasiswa yang kurang handal dalam *programming* saat lulus dari jurusan atau program studinya. Berdasarkan sebuah survei yang dilakukan secara acak pada mahasiswa tingkat awal di FILKOM UB, 76% dari 100 mahasiswa tertarik mempelajari materi kuliah silain pemrograman dasar.

Teknologi terkini memberikan banyak keuntungan untuk banyak orang, khususnya mahasiswa. Berbagai macam informasi dapat disebarkan dengan mudah dengan menggunakan teknologi. Di lingkungan kampus, mahasiwa dapat dengan mudah berbagi informasi tentang perkuliahan dengan temannya. Tidak hanya teman, pada kondisi saat ini, banyak dosen yang memanfaatkan perkembangan teknologi untuk berkomunikasi dengan mahasiswanya. Media grup diskusi atau obrolan dapat digunakan untuk berbagi informasi antar mahasiswa dan dosen pada suatu mata kuliah. Kegiatan berbagi informasi seperti ini mempengaruhi perilaku belajar di lingkungan kampus [2]. Berbagi informasi untuk meningkatkan kualitas pembelajaran telah dilakukan di banyak kampus di Indonesia. Salah satunya adalah penerapan media pembelajaran online atau disebut juga sebagai *Learning Management System* (LMS). LMS terus dikembangkan seiring dengan perkembangan teknologi perangkat bergerak. Pembelajaran tidak hanya bisa dilakukan pada komputer desktop saja, namun dapat juga dilakukan menggunakan perangkat bergerak (*mobile*)[3]. Pembelajaran pemrograman juga dapat dilakukan menggunakan sistem LMS [4], [5].

Permasalahan yang sering muncul dalam LMS adalah proses penilaian. Penilaian pemrograman merupakan tantangan tersendiri dalam pembelajaran pemrograman online berbasis LMS. Kesulitan akan meningkat ketika ujian pemrograman diikuti oleh banyak mahasiswa dalam waktu yang sama. Proses penilaian dengan jumlah yang banyak akan memakan waktu dan tenaga yang tidak sedikit [4]. Teknik penilaian otomatis terhadap ujian pemrograman telah berkembang pesat. Terdapat dua jenis pendekatan dalam melakukan penilaian kode program secara otomatis, yaitu, analisis statis dan analisis dinamis. Statis analisis adalah proses analisis

kode yang dilakukan tidak dengan menjalankan program, sementara analisis dinamis dilakukan dengan melakukan eksekusi program [5].

Proses penilaian kode program dalam sistem LMS memiliki kemiripan dengan proses pendeteksian kode klon (*Clone Code Detection*) dalam kode sumber program. Proses pendeteksian klon dan penilaian kode dilakukan dengan proses yang sama yaitu dengan membandingkan dengan kode pembanding (kunci jawaban). Teknik yang dapat dilakukan juga sama dengan melihat sintaknya (statis) [6] atau melihat hasilnya (dinamis) [7], [8]. Dalam proses pendeteksian klon pada kode sumber, terdapat beberapa tipe klon, antara lain, klon tipe 1 (*exact clones*), tipe 2 (*renamed/ paramerized clones*), tipe 3 (*near miss clones*), dan tipe 4 (*semantic clones*) [9]. Pada tipe 1, 2 dan 3 pendeteksian dilihat dari struktur sintaknya. Sedangkan tipe 4 dilihat dari makna kode (semantik). Kesamaan kode secara makna dapat dilihat dari fungsi kode tersebut [8][10]. Kesamaan fungsi dapat dilihat dari bagaimana suatu fragmen kode memberikan hasil dari suatu input yang diberikan. Dengan metode ini, pendeteksian sangat efektif dalam menangkap klon. Tidak hanya klon tipe 4 saja, tetapi juga tipe 1, 2 dan 3 [8]. Sehingga, metode ini dapat diterapkan untuk mendeteksi klon tipe 1,2,3 dan 4.

Penilaian kode dalam proses pembelajaran online berbasis LMS membutuhkan penilaian yang sifatnya fleksibel. Penilaian tidak hanya melihat berdasarkan kesamaan sintak saja. Karena, jawaban yang dituliskan oleh mahasiswa peserta pembelajaran online bisa beragam untuk menyelesaikan sebuah kasus yang sama. Jika penilaian hanya berdasarkan sintak saja, maka hal ini dapat merugikan mahasiswa.

Dalam penelitian ini, mengusulkan sebuah metode penilaian otomatis pembelajaran pemrograman yang mengadopsi teknik yang ada pada proses pendeteksian klon tipe 4. Proses penilaian otomatis dilakukan dengan pendekatan analisis dinamis pada kode. Tantangan lain adalah bagaimana mekanisme tersebut bisa disatukan dalam sebuah LMS.

2. Pendeteksian kode klon (*Clone Code Detection*)

Kegiatan penyalinan dan penyisipan fragmen kode dari suatu kode sumber ke kode sumber yang lain, dengan atau tidak disertai modifikasi, disebut sebagai kloning kode [9]. Proses tersebut sering dilakukan oleh pengembang perangkat lunak. Kloning kode sering dilakukan dengan berbagai alasan, antara lain keterbatasan waktu pengembangan perangkat lunak. Baker dalam Rattan et al [9] menyatakan bahwa dalam sebuah kode sumber setidaknya terdapat 20-30% kode yang terkloning. Keberadaan kloning pada kode sumber sebuah sistem membutuhkan perhatian lebih pada proses pengembangan perangkat lunak. Hal itu disebabkan karena dampak yang mungkin ditimbulkan dari keberadaan kloning.

Beberapa metode telah digunakan dalam proses penemuan kloning kode. Metode-metode tersebut dikelompokkan dalam dua kelompok, yaitu metode yang dijalankan berdasarkan kesamaan sintak (sintaksis) dan kesamaan makna (semantik). Elva dan Leavens [10], menyebutkan dengan istilah lain yaitu bentuk kesamaan representasional dan fungsional. Kesamaan sintak dilakukan dengan melihat bagaimana kode sumber itu disajikan atau dituliskan. Model pendekatan sintaksis hanya mempertimbangkan kesamaan tulisan dan minim memberikan informasi semantik [11]. Sedangkan, pada kesamaan semantik, proses pencarian kloning kode dilakukan dengan melihat sisi maknanya. Makna dari sebuah fragmen kode dapat dilihat dari fungsinya [10], [12], dan hubungan relasi dari elemen-elemen kode sumber tersebut [11], [13].

Keivanloo dan Rilling [11], mengusulkan sebuah pendekatan yang menekankan pada hubungan semantik pada setiap token. Mereka beranggapan bahwa, apabila hanya berdasarkan pada teks atau sintak akan sangat sulit untuk melakukan pengukuran kesamaan berdasarkan makna (semantik). Keivanloo dan Rilling melakukan perbaikan pada metode pendeteksian kloning berbasis sintak dengan cara menggunakan teknologi semantik web. Semantik web digunakan sebagai alat untuk menghimpun informasi relasi pewarisan pada kode sumber berbasis objek. Hubungan semantik pada token dilihat dari relasi warisan yang dipunyai.

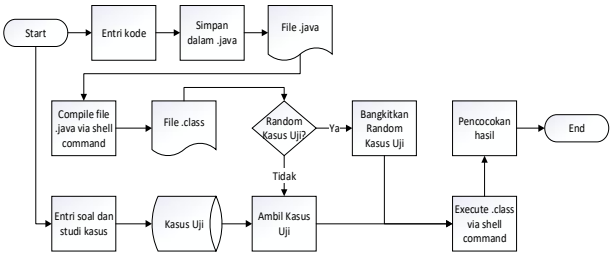
Jiang dan Su [12] menggunakan pendekatan yang berbeda. Dalam penelitiannya, Jiang dan Su mengartikan kesamaan semantik sebagai sebuah kesamaan fungsi (fungsional). Mereka berpendapat bahwa jika dua fragmen kode (pasangan kode) menghasilkan keluaran yang sama ketika mendapat masukan yang sama maka bisa disimpulkan bahwa kedua fragmen kode tersebut adalah sama secara fungsional. Dalam pendekatan ini, terdapat fase pembangkitan masukan yang berfungsi sebagai mendefinisikan masukan untuk pasangan fragmen kode secara otomatis.

Elva dan Leavens [10] memiliki pandangan yang sejalan dengan Jiang dan Su [12]. Mereka memfokuskan penelitian pada metode atau fungsi pada bahasa pemrograman Java. Elva dan Leavens juga percaya bahwa fungsi dari sebuah metode dapat dilihat dari input dan outputnya [8]. Elva dan Leavens melakukan penambahan parameter yang dipertimbangkan dalam mendeteksi kesamaan fungsi antara metode. Parameter tersebut adalah effect (efek) sehingga mereka menamai metodenya dengan *IOE-Behaviour* (*Input, Output, Effect - Behaviour*). Input dilihat dari parameter masukan dari sebuah metode. Output dilihat dari tipe keluaran yang dihasilkan oleh sebuah metode. Sedangkan efek (effect), dilihat dari kelas atau atribut apa yang mungkin dipengaruhi oleh metode tersebut ketika dijalankan.

Priyambadha dan Rochimah [8] melakukan pengembangan dari apa yang sudah dilakukan oleh Elva dan Leavens. Penelitian tersebut bertujuan untuk melakukan deteksi kloning berdasarkan perilaku dari

method dalam kelas. Perilaku dilihat dari input, output dan efek. Pada input atau output, tipe-tipe parameter akan dilihat lebih detail, baik itu berupa kelas, tipe primitif, atau hubungan turunan antar kelas. Sedangkan pada efek, akan dianalisis lebih mendalam. Pendefinisian input dan output pada metode void akan digali, sehingga *method* void tidak diabaikan. Dengan melakukan pencarian kloning kode pada seluruh metode yang ada pada sistem, maka diharapkan kloning dapat ditemukan pada seluruh bagian dari kode sumber. Proses pencocokan kode sumber dilakukan dengan cara dinamis, yaitu dengan cara menjalankan method yang sedang dianalisis.

3. Metodologi



Gambar 1. *Alur Penilaian Otomatis dengan Pendekatan Clone Code Detection*

Pada gambar 1 dijelaskan alur detail dari proses penilaian otomatis menggunakan pendekatan pendeteksian klon pada kode sumber. Tahapan pertama dalam proses penilaian otomatis adalah proses entri kode. Proses entri kode adalah proses yang dilakukan oleh mahasiswa ketika mahasiswa menjawab permasalahan yang ada pada LMS. Kode sumber yang dientri akan di simpan dalam bentuk file yang berekstensi .java. File .java tersebut diberikan penamaan sesuai dengan menggunakan sebuah penamaan yang unik. Karena, mahasiswa yang online dalam LMS dan mengerjakan soal bisa jadi lebih dari satu dalam satu waktu. Kemudian, file .java akan dikompilasi dengan mengeksekusi perintah kompilasi melalui *shell command*. Sistem LMS akan memicu atau mengirimkan perintah untuk dieksekusi dalam *shell command*.

Di sisi lain, admin LMS menyediakan soal beserta kasus uji untuk melakukan penilaian pada kode sumber yang dientrikan oleh mahasiswa. Entri soal dilakukan oleh admin dengan memasukkan permasalahan, kode kunci jawaban dan kasus uji. Kasus uji yang dibuat adalah lima kasus uji. Menurut Elva dan Leavens [8], [10], eksekusi sebuah kode untuk diketahui perilakunya paling tidak dilakukan sebanyak lima kali percobaan. Karena, percobaan diatas lima kali akan menunjukkan hasil yang sama dengan percobaan lima kali. Dalam penelitian ini, uji coba juga akan dilakukan lima kali.

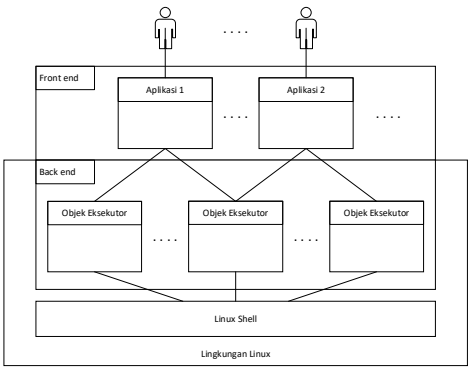
Proses selanjutnya adalah uji coba kode. Uji coba kode dilakukan dengan menjalankan atau mengeksekusi file .class hasil kompilasi dari .java. Dalam menjalankan file tersebut disertakan data input sesuai dengan kode yang dibuat. Data yang akan diinputkan sebagai parameter dari sebuah method dapat dipilih, apakah menggunakan kasus uji yang sudah didefinisikan atau memanfaatkan fitur *random data input* yang disediakan oleh LMS. Random data dapat dilakukan apabila parameter input dari sebuah kode memiliki tipe primitif. Sehingga data secara random dapat dengan mudah dibangkitkan. Untuk data input yang bukan bertipe primitif dapat didefinisikan terlebih dahulu pada saat melakukan entri soal.

Dari lima kali percobaan menjalankan kode, ditangkap nilai keluaran dari kode tersebut. Nilai input dan output dari setiap percobaan akan disimpan. Data ini akan digunakan untuk proses pencocokan apakah kode tersebut memiliki kesamaan secara funsional dengan kunci jawaban. Jika dari kelima percobaan yang sudah dilakukan nilai input dan ouput sesuai antara kode dari mahasiswa dan kode kunci jawaban, maka dapat disimpulkan bahwa kedua kode memiliki fungsionalitas yang sama dalam menyelesaikan masalah. Kode tersebut dianggap benar. Jika dari kelima percobaan tersebut satu saja tidak benar maka kode dianggap salah.

4. Implementasi dan Uji Coba

Implementasi mekanisme penilaian otomatis pada proses pembelajaran pemrograman Java berbasis LMS diwijudkan dalam sistem berbasis web. Sistem tersebut dibangun dengan menggunakan bahasa pemrograman PHP. LMS tersebut dibangun dengan sentuhan gamifikasi di dalamnya dan diberi nama *Code Maniac* (CoMa). *Code Maniac* diimplementasikan di lingkungan FILKOM UB.

Pada gambar 2 dijelaskan gambaran arsitektur sistem penilaian otomatis dengan pendekatan deteksi klon kode. Terdapat dua layer dalam sistem tersebut, yaitu *front end* dan *back end*. *Front end* adalah sisi aplikasi dimana aktor atau pengguna menggunakan aplikasi CoMa. *Back end* adalah sisi server dimana aplikasi yang dibangun dengan PHP ini ditanam pada *webserver* Apache di lingkungan sistem operasi Linux.



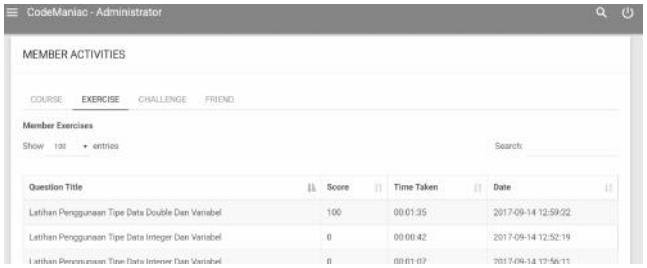
Gambar 2. *Arsitektur Sistem Penilaian Otomatis*

Pada *back end* terdapat beberapa objek eksekutor yang memiliki tugas untuk melakukan eksekusi kode. Eksekusi yang dilakukan objek tersebut adalah kegiatan kompilasi dan eksekusi file .class yang dilakukan di lingkungan *Linux shell*. Hasil dari proses eksekusi akan dikirimkan ke aplikasi untuk dicatat sebagai log sebelum dicocokkan hasilnya dengan hasil dari eksekusi kode kunci.

Uji coba sistem penilaian otomatis ini dilakukan untuk pertama kali di lingkungan SMK Informatika yang ada di kota Malang. Dalam uji coba tersebut melibatkan kurang lebih 300 siswa yang dibagi menjadi dua kelompok. Jadi dalam satu sesi percobaan kurang lebih terdapat 150 siswa online pada sistem CoMa. 150 siswa ini melakukan uji coba dengan cara menjawab persoalan yang ada pada sistem CoMa. Kode yang dimasukkan oleh siswa akan dikompilasi dan dijalankan untuk diketahui apakah kode tersebut sesuai dengan kunci

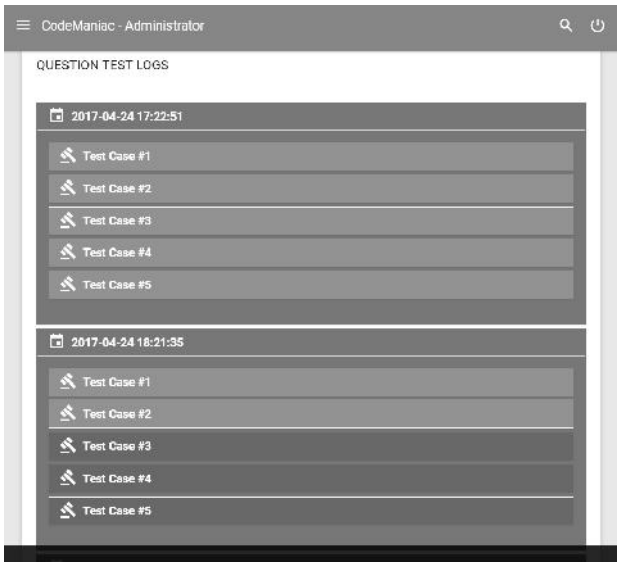
jawaban atau tidak. Jika sesuai maka siswa akan mendapat poin sebagai *reward* dari hasil mengerjakan atau menyelesaikan masalah tersebut.

Gambar 3 menunjukkan hasil pencatatan log pengerjaan latihan di sistem CoMa. Pada gambar tersebut tampak materi latihan Penggunaan Tipe data Double dan Variable dilakukan tiga kali. Pada dua percobaan awal, mendapatkan nilai 0 karena kode yang diinputkan sebagai jawaban atas soal tersebut dinyatakan salah. Untuk percobaan ketiga, nilai yang didapat adalah 100.



Gambar 3. *Tampilan Rekap Log Pengerjaan Latihan di CoMa*

Untuk dapat memutuskan kode dinyatakan benar maka perlu dilakukan uji coba. Seperti yang sudah dijelaskan sebelumnya, uji coba kode dilakukan dengan cara menjalankan kode tersebut (setelah dikompilasi) dengan menggunakan kasus uji. Gambar 4 menunjukkan tampilan log uji coba kode yang disesuaikan dengan skenario uji coba yang dinyatakan oleh Elva dan Leavens. Dalam tampilan tersebut tampak dua kelompok uji coba. Kasus uji yang berwarna hijau menandakan untuk kasus uji tersebut antara input dan output yang dihasilkan oleh kode yang dinilai dan kode kunci adalah sama. Sebaliknya, warna merah menunjukkan hasil yang tidak sama anatar kode yang dinilai dengan kode kuncinya.



Gambar 4. *Log Uji Coba Kode*

5. Hasil dan Pembahasan

Hasil yang diterima setelah uji coba dengan total pengguna sebanyak 300 siswa menunjukkan bahwa proses penilaian kode secara otomatis sangatlah bisa dihandalkan. Proses penilaian dapat dilakukan dengan sangat cepat dan hasil langsung bisa diterima oleh siswa. Dalam sistem CoMa ini, pengguna yang memiliki poin banyak akan ditampilkan di depan aplikasi sebagai jawara kode saat itu.

Pemanfaatan metode deteksi klon kode secara semantik untuk melakukan penilaian pada sistem LMS khususnya untuk pembelajaran pemrograman sangatlah membantu bagi guru atau dosen. Dengan menggunakan metode deteksi klon, klon tipe 1,2,3 dan 4 semua dapat dideteksi. Sehingga dengan metode ini, sistem CoMa tidak menghambat kreatifitas siswa dalam menjawab atau menyelesaikan masalah. Sebagai contoh, dalam sebuah kasus sederhana, siswa diberikan tantangan untuk membuat program dengan input dua buah angka dan mengeluarkan hasil perkalian antara dua angka tersebut. Kunci jawaban dari kasus tersebut adalah seperti pada gambar 5 berikut ini.

```
public int hitung(int a, int b)
{
    int hasil = a * b;
    return hasil;
}
```

Gambar 5. *Kunci Jawaban*

Namun, siswa tidak harus menjawab dengan jawaban yang sama persis dengan kunci untuk mendapatkan poin sempurna. Permasalah tersebut dapat diselesaikan dengan beberapa cara. Seperti contoh jawaban yang ada pada gambar 6. Dengan kode sedikit berbeda, namun secara fungsional kode pada gambar 6 memiliki fungsi yang sama dengan kode pada gambar 5. Dengan

pendekatan penilaian otomatis seperti yang diusulkan, jawaban pada gambar 6 dinyatakan benar dan dapat mendapatkan poin sempurna. Hal seperti ini dirasa penting untuk siswa dapat menggali lagi variasi solusi yang mungkin ada untuk lebih meningkatkan pemahaman terhadap masalah yang sedang dihadapi.

```
public static int hitung (int a, int b)
{
    int hasil = 0;
    for (int i=0; i<b; i++)
    {
        hasil +=a;
    }
    return hasil;
}
```

Gambar 6. *Jawaban*

Kendala didapati tidak terkait dengan proses penilaian melainkan terkait dengan arsitektur sistem *back end*. Sehingga terlihat beberapa kali dalam proses uji coba sistem mengalami perlambatan layanan karena antrian proses kompilasi ataupun proses eksekusi.

6. Kesimpulan dan Saran

Implementasi metode pendeteksian klon kode pada sistem penilaian otomatis di LMS menunjukkan sebuah hasil yang sangat memuaskan. Hal ini terlihat dari uji coba yang dilakukan dari kurang lebih 300 siswa membuahkan hasil penilaian dapat berjalan sesuai dengan yang diharapkan. Kelebihan lain dengan menggunakan metode ini adalah tidak terbatas-nya kreatifitas siswa dalam menyelesaikan permasalahan membuat kode dengan bahasa Java.
Pada waktu mendatang perlu digali lebih dalam tentang mekanisme kompilasi dan eksekusi kode sehingga proses bisa dilakukan lebih cepat dan stabil. Untuk sistem CoMa akan dikembangkan menjadi aplikasi *mobile* sehingga pengguna dapat mengakses kapanpun dan dimanapu.

Daftar Pustaka

[1] C. D. Martin, *Computing curricula 2001*, vol. 35, no. 2. 2003.
[2] K. Iskandar, D. Thedy, J. Alfred, and Yonathan, "Evaluating a Learning Management System for BINUS International School Serpong," *Procedia Comput. Sci.*, vol. 59, pp. 205–213, Jan. 2015.
[3] I. Han and W. S. Shin, "The use of a mobile learning management system and academic achievement of online students," *Comput. Educ.*, vol. 102, pp. 79–89, Nov. 2016.
[4] K. Danutama and I. Liem, "Scalable Autograder and LMS Integration," *Procedia Technol.*, vol. 11, pp. 388–395, 2013.
[5] S. V. Yulianto and I. Liem, "Automatic Grader for Programming Assignment Using Source Code Analyzer," 2014, pp. 0–3.
[6] C. Kustanto and I. Liem, "Automatic Source Code Plagiarism Detection," in *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, 2009, pp. 481–486.
[7] D. Kong, X. Su, S. Wu, and T. Wang, "Detect Functionally Equivalent Code Fragments via K-nearest," *Adv. Comput. Intell. (ICACI), 2012 IEEE Fifth Int. Conf.*, pp. 3–7, 2012.
[8] B. Priyambadha and S. Rochimah, "Case Study on Semantic Clone Detection Based On Code Behavior," *Data Softw. Eng.*, pp. 1–6, 2014.
[9] D. Rattan, R. Bhatia, and M. Singh, *Software clone detection: A systematic review*, vol. 55, no. 7. Elsevier B.V., 2013.
[10] R. Elva and G. Leavens, "Semantic clone detection using method IOE-behavior," *2012 6th Int. Work. Softw. Clones*, pp. 80–81, 2012.
[11] I. Keivanloo and J. Rilling, "Semantic-Enabled Clone Detection," *2013 IEEE 37th Annu. Comput. Softw. Appl. Conf.*, pp. 393–398, Jul. 2013.
[12] L. Jiang and Z. Su, "Automatic mining of functionally equivalent code fragments via random testing," *Proc. eighteenth Int. Symp. Softw. Test. Anal. - ISSTA '09*, p. 81, 2009.
[13] M. Gabel, L. Jiang, and Z. Su, "Scalable detection of semantic clones," *Proc. 13th Int. Conf. Softw. Eng. - ICSE '08*, p. 321, 2008.

Biodata Penulis

Bayu Priyambadha, memperoleh gelar Sarjana Komputer (S.Kom), Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember, lulus tahun 2007. Memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Magister Teknik Informatika Institut Teknologi Sepuluh Nopember, lulus tahun 2015. Saat ini menjadi Dosen di FILKOM Universitas Brawijaya Malang.

Fajar Pradana, memperoleh gelar Sarjana Sains Terapan (S.ST), Jurusan Teknik Informatika PENS ITS Surabaya, lulus tahun 2009. Memperoleh gelar Master of Engineering (M.Eng) Program Pasca Sarjana Magister Teknik Informatika Universitas Gajah Mada Yogyakarta, lulus tahun 2011.Saat ini menjadi Dosen di Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya Malang.

ISSN : 2302-3805