JURNAL TEKNIK INFORMATIKA DAN MULTIMEDIA Vol 3, No.1, Mei 2023, pp. 98 - 103
p-ISSN : 2808-909X (print)
e-ISSN : 2798-2505 (online)
http://journal.politeknik-pratama.ac.id/index.php/JTIM · page 98

PENGARUH PERALIHAN BAHASA KOMPUTER PADA REPLIKASI EYE TRACKING

Yosep Aditya Wicaksono[1], Bagus Sudirman[2], Edy Siswanto[3]

[1] Program Studi Teknik Informatika Universitas Sains dan Teknologi Komputer
Jl. Majapahit 605 Semarang, telp : (024)-6723456, e-mail: yosepadityaw@stekom.ac.id
[2] Program Studi Teknik Informatika Universitas Sains dan Teknologi Komputer
Jl. Majapahit 605 Semarang, telp : (024)-6723456, e-mail: bagus@stekom.ac.id
[3] Program Studi Komputerisasi Akuntansi Universitas Sains dan Teknologi Komputer
Jl. Majapahit 605 Semarang, telp : (024)-6723456, e-mail: edy@stekom.ac.id

ARTICLE INFO

Article history:

Received 10 April 2023
Received in revised form 24 April 2023
Accepted 8 Mei 2023
Available online 15 Mei 2023

ABSTRACT

The use of multiple programming languages during software development is a common practice in modern software development. However, the use of different languages that can affect developer productivity is not widely known. The study presented in this paper replicates a randomized control trial that investigates the use of multiple languages in the context of a database programming task. Participants in this study were given coding assignments written in Java and one of three SQL-like embedded languages: plain SQL on strings, Java methods only, a hybrid embedded language more akin to Java. Apart from recording the responses to the online questionnaire and the participants' solutions to the task, the participants' eye movements were also recorded using an eye tracker. Eye Tracker as a method for the study of software development has developed in recent years and allows for more in-depth information about how developers accomplish programming tasks. This eye tracking method is used as a data collection method in this study. Eye tracking data was collected from 31 participants (from academia and industry) for each of six different programming tasks. In order to compare the effect of inter-group variables and professional experience and in-group task variables on the dependent variable Time in completion, this study used a mixed model ANOVA. The results of this study indicate that, a significant effect on productivity was not found, this is different from the initial research because of the language used. However, the same effect was found from the participants' experience on programming productivity indicating that more experienced programmers were able to complete polyglot programming tasks in a more efficient way. In addition it was found that all participants viewed the

Received April 10, 2023; Accepted Mei 8, 2022; Available online Mei 15, 2023

p-ISSN : 2808-909X e-ISSN : 2798-2505

sample code with the same percentage of time for a given task regardless of experience or language variant provided. Top-level navigation behavior also remains largely unchanged across experiences or language variants. Overall, it can be concluded that the programming level of the polyglot does not have as significant an effect as the task itself. The high-level strategies that participants used appeared to be similar regardless of the language variant presented to them. As a suggestion for future research, the effects of different types of polyglot programming languages should be studied in depth for the conclusions reached to remain correct across various polyglot programming contexts.

Keywords: Eye Tracking, Programming Language, Polyglot Programming, Computer Language

Abstrak

Penggunaan beberapa bahasa pemrograman selama perangkat lunak berkembang adalah praktik umum dalam pengembangan perangkat lunak modern. Akan tetapi, penggunaan bahasa yang berbeda yang dapat memengaruhi pengembang produktifitas justru belum banyak diketahui. Studi yang disajikan dalam penelitian ini mereplikasi kontrol acak percobaan yang menyelidiki penggunaan beberapa bahasa dalam konteks pemrograman basis data tugas. Partisipan dalam penelitian ini diberi tugas pengkodean yang ditulis dalam Java dan salah satu dari tiga bahasa tertanam mirip SQL: SQL biasa dalam string, hanya metode Java, bahasa tertanam hibrid yang lebih mirip dengan Java. Selain merekam tanggapan kuesioner online dan solusi peserta terhadap tugas, gerakan mata peserta juga direkam menggunakan eye tracker. Eye Tracker sebagai metode untuk studi pengembangan perangkat lunak telah berkembang dalam beberapa tahun terakhir dan memungkinkan untuk mendapatkan informasi yang lebih mendalam tentang bagaimana pengembang menyelesaikan tugas pemrograman. Metode Eye tracking ini digunakan sebagai metode pengumpulan data dalam penlitian ini. Data pelacakan mata dikumpulkan dari 31 peserta (dari akademisi dan industri) untuk masing-masing dari enam tugas pemrograman yang berbeda. Untuk membandingkan efek variabel antar kelompok kelompok dan pengalaman profesional dan variabel tugas dalam kelompok pada variabel terikat Waktu dala peneyelesaiinya, penelitian ini menggunakan ANOVA model campuran. Hasil penelitian ini menunjukkan bahwa, efek signifikan pada produktivitas tidak ditemukan, ini berbeda dengan penlitian awal karena bahasa yang digunakan. Namun, efek yang sama ditemukan dari pengalaman peserta terhadap produktivitas pemrograman menunjukkan bahwa programmer yang lebih berpengalaman mampu menyelesaikan tugas pemrograman poliglot dengan cara yang lebih efisien. Selain itu ditemukan bahwa semua peserta melihat kode sampel dengan persentase waktu yang sama untuk tugas yang diberikan terlepas dari pengalaman atau varian bahasa yang diberikan. Perilaku navigasi tingkat atas juga sebagian besar tetap tidak berubah di seluruh pengalaman atau varian bahasa. Secara keseluruhan, dapat disimpulkan bahwa tingkat pemrograman poliglot tidak memiliki efek yang signifikan seperti tugas itu sendiri. Tingkat tinggi strategi yang digunakan peserta tampak serupa terlepas dari varian bahasa yang diberikan kepada mereka. Sebagai saran untuk penelitian dimasa depan, efek dari berbagai jenis pemrograman polyglot bahasa harus dipelajari secara mendalam untuk kesimpulan yang dicapai tetap benar di berbagai konteks pemrograman poliglot.

Keywords: Eye Tracking, Programming Language, Polyglot Programming, Computer Language

1. PENDAHULUAN

Dalam industri pengembangan perangkat lunak pemrograman Polyglot sangatlah umum. Tomassetti dan Torchiano (2014) menemukan bahwa sembilan puluh tujuh persen proyek open source menggunakan lebih dari dua bahasa pemrograman komputer, dan rata-rata, ada lima bahasa pemrograman yang digunakan dalam proyek open source (Tomassetti&Torchiano, (2014); Mayer&Bauer, (2015)). Selain itu, pengembang mengklaim "tahu" sepuluh bahasa komputer yang berbeda dalam tanggapan survei (Meyerovich&Rabkin, (2013)). Namun, penelitian terbaru mengamati bahwa mempelajari bahasa komputer baru memiliki tantangan yang signifikan bahkan untuk pengembang berpengalaman. Pemrograman Polyglot memiliki beberapa tingkat peralihan bahasa yang berbeda diamati, diantaranya adalah :

1) level proyek
2) level fille
3) level tersemat.

p-ISSN : 2808-909X e-ISSN : 2798-2505

Tingkat proyek dan file peralihan tersirat dengan namanya menggunakan pemrograman poliglot tingkat proyek proyek ditulis dengan bahasa yang berbeda tetapi setiap proyek individu menggunakan satu bahasa. Demikian pula, peralihan tingkat file memiliki setiap file yang ditulis dalam satu bahasa, dan di seluruh proyek bahasa tersebut terkadang berbeda. Peralihan bahasa tertanam dapat mengambil banyak bentuk, tetapi dalam konteks ini mengacu pada satu bahasa, bahasa yang disematkan, dan bahasa yang tertanam di bagian lain, yakni bahasa host. Sementara studi tentang pemrograman polyglot dalam pengembangan perangkat lunak telah lebih umum dalam beberapa tahun terakhir, "sebagian besar studi berfokus pada efek dari pemrograman polyglot pada kualitas kode" (Morisio et al., (2012); Tomassetti&Torchiano, (2014)). Sementara beberapa penelitian berfokus pada perbedaan budaya mengadopsi dan menggunakan beberapa bahasa pemrograman, dan terdapat dampak dan efek pemrograman polyglot pada perilaku pengembang yang sebagian besar belum dijelajahi. Baru-baru ini, sebuah penelitian dilakukan oleh Stefik et al. (2013) yang mengamati "efek penggunaan alih bahasa tersemat studi berbasis kuesioner online".

Tabel 1Studi A adalah studi online direplikasi dalam Studi B (kontribusi utama dari penelitian ini).

Study Name	Study Description	Methods Used
Study A	Online-Only Study	Online Questionnaire
Study B	Eye-Tracking Replication	Online Questionnaire, Eye Tracking

Sebuah versi baru dari penelitian ini diterbitkan pada penelitian ini dan dilalukan 2x. Peserta menyelesaikan enam tugas menggunakan API dengan variasi tingkat peralihan bahasa tertanam (Study A). Studi A hanya merekam dan menganalisis solusi peserta untuk tugas dan data interaksi mereka di lingkungan belajar online. Selanjutnya, studi replikasi (Study B) dari Studi A disajikan dengan tambahan metode pengumpulan data yaitu menggunakan peralatan pelacakan mata. Perbedaan antara Studi A dan Studi B tercantum dalam Tabel 1. Perhatikan bahwa semua data yang dikumpulkan dalam Studi A juga dikumpulkan dalam Studi B tetapi selain di Studi B, data pelacakan mata juga dikumpulkan di samping semua data situs web. Pelacakan mata sebagai metode untuk rekayasa perangkat lunak studi telah berkembang dalam beberapa tahun terakhir. "Gerakan mata telah terbukti menangkap informasi yang jauh lebih tidak jelas selama tugas dibandingkan dengan interaksi data atau studi berpikir keras saja" (Fritz e6t al., (2015)). Pelacakan mata telah digunakan untuk mendapatkan wawasan tentang bagaimana pengembang membaca (Stelovsky et al., (1990); Rodeghero&McMillan, (2015)), meninjau (Matsumoto et al., (2006); Maletic et al., (2012); Tamm et al., (2015); Fritz e6t al., (2015)) dan meringkas sumber kode (D'Mello et al., (2014)). Pergerakan mata memungkinkan untuk menangkap sekilas antar elemen pada layar dan pergeseran perhatian bahwa peserta secara sadar menyadari diri mereka sendiri.

Pekerjaan yang berhubungan

Produktivitas programmer dipelajari dalam berbagai aspek pemrograman. Studi berkisar dari fitur bahasa pemrograman seperti sintaks (Stefik&Siebert, (2013)) dan sistem tipe (Hoppe&Hanenberg, (2013)) desain API (Stylos&Myers, (2008)) dan efek kesalahan (Becker, (2016)) untuk studi mencoba untuk menyelidiki proses kognitif yang terlibat (Brechmann et al., (2017); Tamm et al., (2015)).

Pelacakan Mata dalam Pemahaman Program

Banyak penelitian dilakukan dalam domain rekayasa perangkat lunak yang berfokus pada sub-bidang pemahaman program (Brooks, (1983)). Sejumlah besar studi penelitian di bidang pemahaman program telah memanfaatkan pelacakan mata untuk memiliki pemahaman yang lebih baik bagaimana programmer memahami program, dan telah memberikan wawasan dan informasi rinci tentang proses kognitif strategi pemahaman program. Alasan lain mengapa pelacakan mata terus berkembang adalah karena itu dapat memberi lebih banyak informasi daripada data interaksi sederhana atau studi berpikir keras. Fritz e6t al., (2015) menemukan bahwa data pelacakan mata lebih berbutir halus daripada data interaksi biasa dan dapat memberikan wawasan tentang cara programmer membaca kode. Tamm et al., (2015) melakukan penelitian untuk melihat perbedaan pada cara mata seseorang membaca kode versus cara mata seseorang dalam membaca kata-kata, selain itu, perbedaan antara programmer ahli dan programmer pemula membaca kode juga dibandingkan. Mereka membawa empat belas pemula dan sembilan insinyur perangkat lunak profesional. Mereka melakukan percobaan dengan melacak peserta saat mereka membaca kode Java. Mereka menemukan bahwa para pemula melihat kode dengan cara linier yang sama seperti membaca 80% dari waktu. Mereka menemukan bahwa para pemula membaca kode dengan cara yang mirip

p-ISSN : 2808-909X e-ISSN : 2798-2505

dengan cara mereka membaca kata kata sementara para ahli mengguankan cara yang berbeda ketika membaca kode (Tamm et al., (2015)).

Penelitian yang dilakukan oleh D'Mello et al., (2014) meringkas kode tugas pada sistem open source Java besar dan menemukan bahwa pengembang cenderung melihat panggilan paling banyak dibandingkan dengan tanda tangan metode. Ini menunjukkan bahwa pengembang berperilaku berbeda saat diberi tugas dengan kode yang realistis. Pelacakan mata telah digunakan untuk mengevaluasi dan mempelajari banyak aspek rekayasa perangkat lunak. Beberapa penelitian juga menunjukkan bahwa pengembang menggunakan situs web eksternal sebagai Stack Over dan alat bantu dalam memahami program menggunakan eye tracking sebagai metodologi. Data pelacakan mata yang dikumpulkan selama tugas rekayasa perangkat lunak juga telah dianalisis menggunakan visualisasi. Visualisasi ini membantu menjelajahi mata yang padat melacak data dan mengungkapkan tren yang dapat diverifikasi dengan data kuantitatif. Dalam penelitian lain, perbedaan gerakan mata antara metode besar dan pendek juga diamati, dan ditemukan bahwa pengembang menggunakan metode yang lebih kecil untuk waktu yang lebih singkat daripada metode yang lebih besar, tetapi memiliki durasi xasi yang lebih lama per baris dalam metode tersebut. Perbedaan antar metode ini didasarkan pada ukuran menunjukkan bahwa temuan penting dapat ditemukan menggunakan metode kecil atau besar, sehingga berbagai metode harus digunakan untuk memastikan bahwa temuan digeneralisasi ke berbagai metode yang lebih besar. Perilaku membaca programmer pemula dan non-pemula yang diminta untuk menjawab pertanyaan pemahaman untuk program C++. Gerakan mata dianalisis dengan membagi program menjadi potongan kode yang logis . Hasil menunjukkan bahwa ketika peserta membaca metode yang lebih kecil, potongan yang paling banyak dihabiskan peserta untuk xating memiliki tingkat yang sama dan perbedaan antara pemula dan non-pemula hanya muncul ketika metode yang lebih besar dibaca.

Pemrograman Poliglot

Dalam literatur ilmiah, manfaat dan kerugian pemrograman poliglot pada tingkat faktor manusia kurang dieksplorasi (Visser et al., (2000)). Penggunaan bahasa yang lebih tepat untuk suatu tugas mengarah pada produktivitas yang lebih baik dan pemeliharaannya jauh lebih mudah dengan mengurangi baris kode proyek (Fjeldberg, (2008)). Pandangan terakhir ini, bahwa bahasa tambahan menyebabkan ketegangan, dan menunjukkan bahwa ketidakcocokan antara bahasa pemrograman justru menjadi penghalang dengan perbedaan yang bervariasi di seluruh pasangan bahasa (mis., Java ke Kotlin vs. pasangan lainnya). Dalam penelitian ini, metodologi yang berbeda digunakan yakni, uji coba terkontrol secara acak pada variasi yang lebih besar tingkat pengalaman dibandingkan dengan studi metode campuran hanya pada profesional, hasilnya saling melengkapi. Studi sebelumnya telah menunjukkan bahwa pemrograman poliglot sangat luas dan banyak digunakan dalam pengembangan perangkat lunak (Tomassetti&Torchiano, (2014)). Mayer et al., (2015) mempelajari frekuensi pemrograman polyglot dalam proyek industri dan bagaimana pengembang memandang proyek menggunakan beberapa bahasa pemrograman dengan melakukan survei dengan peserta industri (Le et al., (2017)). Mayer et al., (2015) menemukan bahwa pengembang melaporkan bahasa tertentu lebih cocok untuk tugas-tugas tertentu dan menggunakan beberapa bahasa memungkinkan persyaratan proyek untuk diterjemahkan ke dalam kode lebih mudah. Namun, pengembang juga merasa menggunakan beberapa bahasa pemrograman dalam proyek perangkat lunak bermasalah untuk dipahami proyek dan perubahan sistem. Mereka juga menemukan bahwa kebanyakan hubungan antara dua bahasa terjadi antara bahasa tujuan umum seperti Java dan bahasa khusus domain seperti SQL.

Masalah yang paling umum adalah pengembang melaporkan tautan lintas bahasa ini adalah bug yang dibuat saat perubahan muncul, ragu dalam memilih pengidentifikasi yang digunakan dalam kedua bahasa karena takut melanggar proyek, dan kesulitan dalam pemahaman program. Baru-baru ini anti-pola untuk proyek pemrograman polyglot dilaporkan, dokumentasi pengembang dan laporan bug yang berisi kata kunci umum yang berhubungan dengan polyglot programming dan website terpercaya yang digunakan developer seperti Stack Over flow, masalah GitHub, Bugzilla, Pengembang IBM, dan pengembang android untuk menemukan praktek pemrograman polyglot mulai dianalisis. Sementara studi yang berfokus pada efek pemrograman polyglot pada kualitas kode (Morisio et al., (2012); Tomassetti&Torchiano, (2014)) memiliki sedikit literatur yang berisi pengukuran efek pemrograman polyglot pada produktivitas programmer. Satu studi baru-baru ini yang mengeksplorasi dampak pemrograman polyglot pada developer dilakukan oleh Stefik&Siebert, (2013). Stefik&Siebert, (2013) melakukan studi percontohan menggunakan online-kuesioner yang terdiri dari enam tugas pemrograman database dan membagi peserta menjadi tiga kelompok bahasa yang berbeda dengan tingkat peralihan bahasa yang berbeda. Ini adalah uji coba kontrol acak pertama pada pemrograman poliglot dan berfungsi sebagai pilot untuk Studi A. Sementara beberapa

p-ISSN : 2808-909X e-ISSN : 2798-2505

peneliti telah mengusulkan penggunaan pelacakan mata sebagai metodologi untuk mempelajari tugas pemrograman polyglot (Konopka, (2015)), itu tetap belum dijelajahi dalam percobaan studi.

Ekstensi Chrome iTrace - Infrastruktur Pelacakan Mata

iTrace adalah infrastruktur komunitas (http://www.i-trace.org) untuk melakukan studi pelacakan mata dalam berbagai lingkungan pengembangan terintegrasi (IDE). Saat ini dukungan untuk Visual Studio dan Eclipse disediakan oleh tim iTrace. Selain kedua IDE ini, karena pengembangan tidak hanya terbatas pada IDE, pelacakan mata di dalam Chrome direncanakan dan mendukung publik di masa depan infrastruktur dengan dukungan untuk situs web seperti Stack Overflow, Bugzilla, dan GitHub. Pelacakan mata sebagai metode untuk memahami pengembangan perangkat lunak telah mendapatkan popularitas di komunitas rekayasa perangkat lunak sejak tahun 2006. Namun, studi dilakukan menggunakan pelacakan mata sebelum iTrace secara realistis hanya dapat dilakukan pada potongan kode pendek. Ini karena keterbatasan perangkat lunak pelacakan mata sebelumnya perlu membatasi stimulus ke satu layar statis atau mengizinkan konten dinamis dengan pemrosesan pos yang membosankan dan memakan waktu untuk memetakan pandangan ke elemen secara manual dari kode sumber. Penggunaan banyak file, pelipatan kode, dan pengguliran harus dilakukan dan ditangani sepenuhnya secara manual menggunakan pendekatan ini. iTrace memungkinkan penggunaan pelacakan mata dalam lingkungan pengkodean yang realistis dengan melacak elemen kode sumber di layar dan memetakan koordinat pandangan ke elemen secara otomatis. Pendekatan otomatis ini memungkinkan peneliti untuk melakukan studi dengan banyak file dan konteks beralih, lebih hemat waktu untuk memetakan pandangan secara manual.

Prototipe iTrace pertama dikembangkan pada tahun 2012 sebagai plugin Eclipse. Pada tahun 2018, sebuah versi baru dan refactored dari iTrace telah dibuat. Aplikasi ini dibagi menjadi beberapa aplikasi yang lebih kecil yang berinteraksi dan berkomunikasi satu sama lain dengan satu aplikasi inti pusat yang bertanggung jawab atas data dan manajemen sesi, berinteraksi dengan pelacak mata, dan berkomunikasi dengan plugin melalui soket TCP. Plugin dikembangkan sebagai ekstensi untuk IDE yang berkomunikasi dengan aplikasi inti, menerima data tatapan, dan memetakan elemen tatapan ke baris dan kolom sumber elemen kode. Pemetaan tambahan untuk elemen seperti tombol atau kotak dialog bisa diimplementasikan, tetapi tidak tersedia dalam rilis publik produk. Refactoring dilakukan untuk memungkinkan pengembangan IDE atau platform tambahan yang lebih mudah dimasa depan seperti Visual Studio Code, Atom, atau Intell. Pada tahun 2019, iTrace versi publik pertama dirilis dengan fitur tambahan dari refactoring awal. Aplikasi inti, iTrace-Core, diberikan kemampuan untuk berkomunikasi dengan beberapa klien plugin secara bersamaan. iTrace-Core bisa berkomunikasi dengan plugin menggunakan soket TCP atau koneksi WebSocket dengan pengaturan untuk mengizinkan komunikasi soket untuk menggunakan port yang dapat dikonfigurasi pengguna untuk menghindari port conflict.

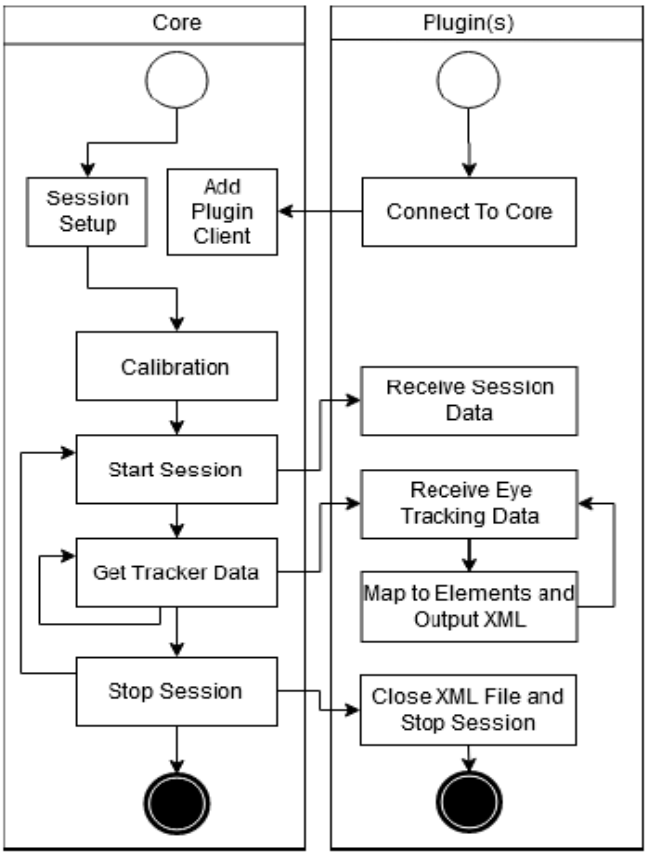
Ini memungkinkan plugin dikembangkan untuk platform tambahan seperti Google Chrome dan Kode Visual Studio. Selain itu, iTrace-Core dapat berinteraksi dengan dan menerima data pelacakan mata dari pelacak seri GazePoint GP3 dan pelacak Tobii Pro. Tambahan dukungan pelacak dapat diterapkan di versi iTrace-Core yang akan datang. Keduanya Plugin Eclipse dan Visual Studio bersumber terbuka dan tersedia untuk umum. Pemrosesan pasca fitur infrastruktur sangat ditingkatkan dalam rilis ini dengan tambahan untuk semua data tatapan dan analisis untuk direkam dalam satu database SQLite. Database ini tidak hanya berisi sesi satu peserta tetapi dapat berisi sesi lengkap paket studi yang memungkinkan peneliti membuat artefak untuk makalah yang diterima dan bertukar data antar kolaborator. Karena database SQLite standar digunakan untuk analisis data, perintah SQL standar apa pun dapat digunakan dalam penelitian analisis, dan data dapat diekspor untuk analisis di dalam aplikasi eksternal. Saat ini, beberapa batasan masih ada di dalam iTrace. Keterbatasan utama adalah ketidakmampuan untuk secara akurat memetakan data pandangan ke kode sumber yang sedang diedit selama sesi pelacakan mata. Ini membatasi jenis tugas yang dapat direkam menggunakan iTrace karena tugas bug xing dan tugas refactoring memerlukan kode sumber untuk diedit dan tidak dapat dipetakan secara akurat menggunakan sistem iTrace saat ini. Tambahan, iTrace memiliki dukungan terbatas untuk validasi dan koreksi data, dan fitur sedang dikembangkan dirancang untuk menambahkan lebih banyak dukungan di area ini.

Arsitektur iTrace

Arsitektur iTrace dibagi menjadi aplikasi inti dan berbagai aplikasi plugin. Aplikasi inti menangani manajemen sesi, interfacing dengan pelacak mata, dan berkomunikasi dengan plugin saat plugin berkomunikasi dengan aplikasi inti dan memetakan pandangan ke garis dan kolom kode sumber dan elemen IDE lainnya (Gambar 1). Ketika sebuah plugin terhubung ke aplikasi inti, plugin tersebut

p-ISSN : 2808-909X e-ISSN : 2798-2505

ditambahkan ke daftarklien plugin. Saat sesi di aplikasi inti dimulai, sebuah paket dikirim ke setiap klien dengan data sesi dan plugin membuat XML yang sesuai file dan bersiap untuk menerima data tatapan dari aplikasi inti. Jika sebuah plugin terhubung ke aplikasi inti saat sesi sedang berlangsung maka paket sesi awal ini akan dikirim segera setelah koneksi antara core dan plugin dibuat. Untuk setiap tatapan yang diterima inti maka koordinat tatapan akan dicatat pada file XML dengan real-time tiemstamp dan informasi tatapan tambahan. Jika tatapan itu tepat, maka core akan mengirimkan setiap klien koordinat pandangan dan stempel waktu pandangan. Setiap plugin akan menerima tampilan dan memetakannya ke elemen di dalam aplikasi menggunakan API khusus aplikasi dan akan mencatat stempel waktu pemetaan dan pandangan ke file plugin XML. Baik iTrace-Eclipse dan iTrace-VisualStudio menerima tatapan tersebut ke buffer dengan satu entri. Jika tatapan lain datang sebelum tatapan sebelumnya mulai diproses, tatapan baru akan menimpa tatapan lama di buffer. Ini dilakukan untuk memastikan bahwa penundaan apa pun yang disebabkan oleh API khusus aplikasi tidak menyebabkan tatapan berikutnya akan dipetakan secara tidak benar. Saat pengguna mengakhiri sesi, maka paket dikirim ke setiap klien. Baik klien maupun plugin akan menulis sesi akhir dari file XML dan menutup file XML.



Gambar 1: Arsitektur Inti dan Plugin iTrace

Implementasi Plugin Chrome

Ada beberapa detail implementasi khusus plugin chrome yang membedakan dari detail implementasi plugin umum, implementasi khusus ini terjadi karena plugin chrome dibuat sebagai chrome ekstensi dan tidak memiliki izin yang sama untuk sumber daya di komputer host. Detail implementasi khusus chrome-plugin pertama yang berbeda dari plugin sebelumnya adalah bahwa koneksi WebSocket digunakan sebagai lawan dari soket TCP koneksi yang digunakan oleh iTrace-Eclipse dan iTrace-VisualStudio. Ini diperlukan sebagai Ekstensi Chrome tidak dapat membuat soket tetapi sebaliknya harus bergantung pada WebSockets untuk membuat koneksi ke iTrace-Core. Selain itu, file XML tidak terus-menerus ditulis saat data tampilan diterima dari inti karena akses file asli tidak diperbolehkan di ekstensi Chrome. Sekali sesi telah berakhir, maka pengguna harus mengklik tombol secara manual untuk mengekspor file XML dan file tersebut tidak akan disimpan di direktori yang sama dengan file XML int data atau data plugin lainnya karena satu-satunya cara untuk menyimpan file ke direktori file asli adalah menyimpan file sebagai unduhan Chrome.

p-ISSN : 2808-909X e-ISSN : 2798-2505

Berbeda dengan plugin lainnya, iTrace-Chrome akan terputus dari Chrome setelah sesi berakhir. Setelah sesi berakhir dan pengguna mengekspor file XML, koneksi antara plugin Chrome dan aplikasi inti akan berakhir. Pengguna harus menyambungkan kembali plugin ke aplikasi inti jika ingin merekam mata gerakan dari tugas tambahan. Untuk memetakan koordinat tatapan yang diterima dari plugin chrome aplikasi inti ke elemen situs, satu set pemetaan yang berbeda digunakan untuk masing-masing elemen lokasi. Javascript digunakan untuk mendapatkan daftar DOM elemen pada koordinat tatapan dan kombinasi kelas, id, dan nama tag diperiksa untuk menentukan jenis elemen apa yang sedang dilihat. Untuk mengetahui baris apa yang sedang dilihat, setiap baris teks di dalam bagian lingkungan yang digunakan dalam studi dibungkus di dalam div untuk memungkinkan setiap baris dipilih secara individual. Terakhir, untuk lingkungan studi yang digunakan, waktu yang digunakan untuk menyinkronkan data aplikasi inti dan data plugin juga ditulis ke elemen div tersembunyi setiap kali data baru diproses dan setiap 5 detik situs web akan mencatat waktu acara ini bersama dengan inputnya k ebagian penyelesaian.

Tantangan yang Dihadapi

Ada beberapa tantangan yang dihadapi dengan plugin Chrome. Pertama, tidak ada kemampuan untuk menulis ke sistem file asli. Semua penulisan file harus dilakukan dalam mesin JavaScript browser, dan API pengunduhan file harus digunakan untuk menulis data XML ke sistem file komputer. Tantangan lain yang dihadapi adalah plugin Chrome umum tidak layak karena sangat bergantung pada antarmuka yang digunakan untuk setiap studi. Untuk setiap situs yang diperlukan untuk studi, logika tambahan harus ditambahkan untuk memiliki kemampuan mengklasifikasikan dan memetakan pandangan ke elemen yang penting secara semantik. Mungkin ada plugin Chrome umum untuk Stack Over flow dan laporan bug situs web serta GitHub yang mengikuti beberapa struktur dalam Model Objek Dokumen (DOM), dengan konten dinamis, banyak logika perlu ditangani untuk memetakan elemen yang benar diperlukan. Selain itu, blok teks yang besar memiliki kesulitan untuk memetakan baris dan kolom informasi mengenai kata karena informasi baris dan kolom dapat berubah sewaktu-waktu karena perubahan ukuran jendela dan simpul teks diperlakukan sebagai elemen tunggal dalam DOM. Informasi baris dan kolom harus diperoleh melalui kombinasi baris dan kolom yang dihitung berdasarkan ukuran font teks pada layar dan menambahkan pembungkus HTML di sekitar sekelompok teks.

Terakhir, penanganan data yang berubah merupakan masalah yang sulit bagi semua plugin. Data dari plugin saat ini tidak dapat digunakan untuk menganalisis data pelacakan mata pada kode sumber yang tidak statis. Informasi baris dan kolom kode sumber yang telah diedit tidak dapat digunakan untuk mendapatkan informasi semantik tentang kode sumber yang sedang dilihat. Pengeditan adalah masalah yang sulit untuk ditangani dan diperhitungkan dalam analisis pelacakan mata. Saat ini, beberapa solusi sedang dieksplorasi oleh tim iTrace tetapi belum ada solusi yang diimplementasikan ke plugin mana pun. Untuk menangani pengeditan untuk studi ini, lingkungan studi secara otomatis menyimpan teks untuk solusi setiap 5 detik untuk mengurangi masalah ini.

EKSPERIMEN

Study A

Banyak detail pengaturan eksperimen tetap konsisten di antara Study A dan Study B. Lingkungan online yang sama digunakan bersama dengan tugas yang sama persis. Perbedaan utama adalah penambahan eye tracking sebagai metode pengumpulan data. Karena metode tambahan ini, seorang moderator diharuskan hadir pada penelitian untuk memastikan bahwa data pelacakan mata yang akurat dikumpulkan. Studi A dilakukan sepenuhnya secara online tanpa pengawasan peserta, Studi B dilakukan sepenuhnya secara pribadi dengan moderator yang hadir dalam studi tersebut.

Desain Percobaan

Replikasi ini adalah desain tindakan berulang di mana peserta secara acak ditugaskan ke salah satu dari tiga kelompok eksperimen. Setiap kelompok diberi varian bahasa yang berbeda untuk menyelesaikan tugas. 6 tugas yang sama diberikan kepada semua peserta terlepas dari kelompok mereka. Satu-satunya perbedaan adalah varian bahasa yang digunakan untuk menyelesaikan tugas.

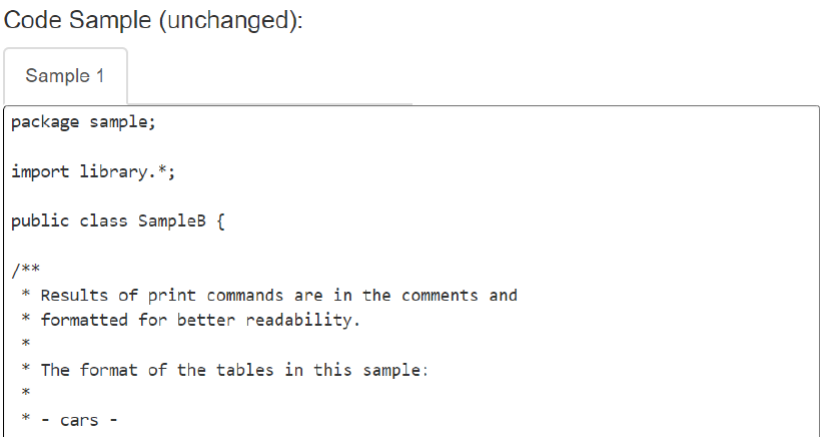
Tabel 2: Jumlah Peserta dan Distribusi Antar Kelompok

p-ISSN : 2808-909X e-ISSN : 2798-2505

	Hybrid	String-Based	Object-Oriented
Students	5	5	6
Professional	4	6	5

Pengaturan Studi

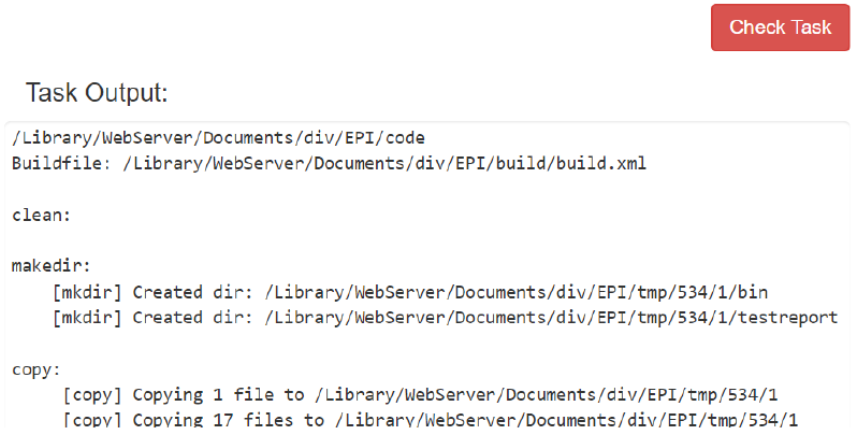
Replikasi dilakukan menggunakan platform online asli yang digunakan untuk belajar. Platform ini memberi tahu para peserta tentang hak dan persetujuan mereka. Peserta mengisi pra-kuesioner yang mengklasifikasikan masing-masing peserta menjadi salah satu kelompok pengalaman (Mahasiswa sarjana, mahasiswa pascasarjana, pasca sarjana, non-gelar, atau profesional). Kuesioner juga untuk informasi tambahan seperti jumlah total pengalaman pemrograman, jumlah pengalaman kerja pemrograman, bahasa asli mereka, dan usia juga diberikan. Ketika pra-kuesioner selesai peserta disajikan dengan layar yang merinci prosedur untuk sisanya studi. Seorang moderator hadir selama penelitian untuk memastikan bahwa data pelacakan mata benar dan peserta dilacak dengan benar. Sebelum mengerjakan setiap tugas, eye tracker dikalibrasi untuk memastikan data berkualitas tinggi. Sebelum tugas pertama, peserta memiliki waktu hingga 5 menit untuk membaca kode sampel yang memberikan contoh yang mirip dengan tugas mereka dan ditulis dalam varian bahasa yang ditugaskan. Kode sampel ini tersedia bagi para peserta untuk dibaca selama sisa waktu belajar. Kode sampel ada di sisi kiri halaman web sementara area solusi peserta mengetik jawaban mereka berada di sisi kanan di bawah waktu tersisa untuk menyelesaikan tugas seperti yang ditunjukkan pada Gambar 2 dan 3.



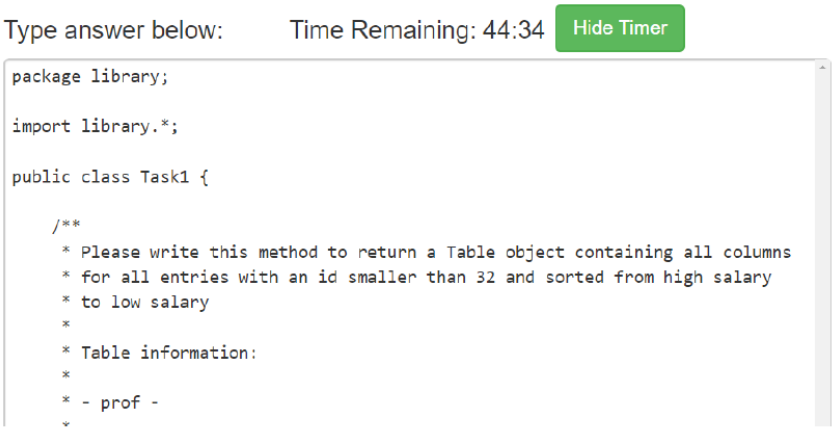
Gambar 2: Lingkungan Studi: Contoh Kode

Setiap tugas terdiri satu tugas database (memilih dan menyortir tabel yang ada, mengedit baris tertentu, menambahkan baris tambahan, dan memilih dan menggabungkan beberapa tabel). Jika seorang peserta tidak dapat menyelesaikan tugas di 45 menit, maka tukgas akan otomatis berakhir dan peserta akan beralih ke tugas berikutnya setelah kalibasi ulang pelacak mata. Batas waktu tugas ini telah dibuat untuk mencegah percobaan mengambil terlalu banyak waktu dan membatasi jumlah waktu maksimum peserta menjadi empat setengah jam seperti yang dijanjikan kepada peserta. Selam titik tugas, pesrta dapat memeriksa keakuratan jawahan mereka dengan waktu yang masih tersisa sebelum berpindah ke waktu berikutnya menggunakan tombol p”periksa tugas” di bawah area solusi. Solusi yang mereka usulkan kemudian dikirim ke server tempat dikompilasi bersama dengan kelas tambahan yang diperlukan dan dijalankan terhadap sejumlah unit- tes. Jika solusi peserta lulus unit-test, maka output tugas dicetak bahwa tes berhasil, dan munculan ditampilkan kepada pengguna yang meminta mereka untuk beralih ke tugas berikutnya. Jika solusi peserta gagal untuk dikompilasi atau gagal unit-test, output tugas akan menampilkan kesalahan kompilasi atau test-case yang gagal Peserta dapat mengedit dan menguji solusi mereka sebanyak yang mereka inginkan sampai mereka berhasil menyelesaikan tugas atau melebihi batas waktu tugas 45 menit.

p-ISSN : 2808-909X e-ISSN : 2798-2505



Gambar 3: Lingkungan Studi: Output Tugas Gambar 4.2: Lingkungan Studi: Output Tugas



Gambar 4: Lingkungan Studi: Area Solusi, Timer

Task	Name	Description
Task 1	Simple Select	Select query with a single conditional and sort
Task 2	Moderate Select	Select query with a composite conditional
Task 3	Difficult Select	Select query with several composite conditionals
Task 4	Update	Update a single entry in an existing table
Task 5	Insert	Insert a single entry into an existing table
Task 6	Join Select	Select query requiring a join between two tables

Tabel 3 Daftar Tugas yang Digunakan

Intervensi

Tiga kelompok berbeda dirancang dengan berbagai tingkat peralihan bahasa. API yang dirancang untuk eksperimen ini memungkinkan kueri basis data dan desain setiap varian API dipusatkan di sekitar berbagai ide pendekatan kueri dan membutuhkan jumlah peralihan bahasa yang berbeda yang diperlukan untuk menyelesaikan tugas. Kode Java digunakan untuk berbagai panggilan API dengan berbagai tingkat pernyataan SQL tertanam untuk berinteraksi dengan server basis data. Grup pertama (dalam daftar 4 pada listing 1) menggunakan pendekatan string yang mengharuskan pengguna API untuk mengetahui sintaks yang tepat dari kueri SQL yang ingin mereka tulis dan tidak memiliki dukungan pengecekan tipe untuk kueri SQL. Satu-satunya pemeriksaan kesalahan dalam pendekatan ini terjadi di database, dan programmer harus mengandalkan umpan balik dari database untuk menemukan kesalahan apa pun dalam kueri mereka. Selain itu, kueri berbasis string harus sama persis dengan varian SQL yang digunakan di server basis data. Namun, setiap pengguna dengan pengalaman SQL yang memadai diberikan lebih banyak fleksibilitas dengan memungkinkan mereka untuk menggunakan keseluruhan bahasa SQL.

p-ISSN : 2808-909X e-ISSN : 2798-2505

Listing 1: Contoh Desain Berbasis String

```
1 public Table query(Table table) throws Exception {
2     Query query = new Query();
3     query.Prepare("SELECT Field1, Field2 "
4         +"FROM table WHERE Field1 < 234 AND Field2 > 42 "
5         +"ORDER BY Field3 DESC");
6     Table result = table.Search(query);
7     return result;
8 }
```

Kelompok kedua menggunakan pendekatan metode yang membutuhkan pengguna API untuk menggunakan sejumlah pemanggilan metode untuk membuat kueri. Varian ini juga hanya menggunakan satu bahasa pemrograman dan diklasifikasikan sebagai API monoglot, dan varian ini menghilangkan kebutuhan untuk beralih antar bahasa pemrograman untuk menulis kueri. Ini mungkin berdampak pada produktivitas karena menghindari biaya peralihan. Kelompok ketiga menggunakan pendekatan hibrida dari sebelumnya ua varian bahasa. Proses pembuatan kueri dipisahkan menjadi pemanggilan metode yang berbeda, tetapi setiap langkah proses pembuatan kueri menggunakan string untuk menyusun elemen di dalam langkah tersebut. Sintaks khusus ini berada di antara peralihan bahasa biasa dari SQL ke Java karena sintaksnya lebih dekat ke bahasa host. Spektrum poliglot yang terlihat dalam percobaan ini dapat dilihat sebagai spektrum keputusan desain bahasa. Dalam API berbasis string, SQL langsung disematkan ke dalam Java dan tidak ada koneksi langsung antara bahasa sementara di API Berorientasi Objek, metode dalam Java dibuat untuk menyelesaikan operasi seperti SQL.

Listing 2: Contoh Desain Berorientasi Objek

```
1 public Table query(Table table) throws Exception {
2   Query query = new Query();
3   query.AddField("Field1")
4     .AddField("Field2");
5   query.Filter(q.Where("Field1").LessThan(234).And("Field2").
      ↪ GreaterThan(42));
6   query.SortHighToLow("Field3");
7   Table result = table.Search(query);
8 }
```

Listing 3: Contoh Desain Hybrid

```
1 public Table query(Table table) throws Exception {
2     Query query = new Query();
3     query.AddFields("Field1, Field2");
4     query.Filter("Field1 < 234 AND Field2 > 42");
5     query.SortHighToLow("Field3");
6     Table result = charts.Search(query);
7     return result;
8 }
```

Variabel

Beberapa variabel dependen digunakan dalam studi replikasi. Variabel dependen pertama adalah waktu untuk solusi yang benar yang diukur dalam studi asli. Jika peserta gagal menyelesaikan tugas, waktu penyelesaian yang benar diatur ke total waktu yang dihabiskan untuk mencoba menyelesaikan tugas yang akan menjadi 45 menit karena batas waktu tugas. Variabel dependen kedua adalah total waktu xation yang dihabiskan di dalam area of interest (AOI) di website tempat mereka mengambil studi. Ada enam AOI tingkat atas yang ada di setiap tugas: Tombol Periksa Tugas, Timer, Info Tugas, Area Solusi, Contoh Kode, dan Keluaran Tugas.

Randomisasi

Setelah peserta memasuki tahun kuliah atau status profesional, mereka ditugaskan ke kelompok pengalaman berdasarkan tanggapan mereka. Distribusi peserta di setiap kelompok pengalaman di antara ketiga perlakuan eksperimental ini dipantau di lingkungan studi. Jika distribusi ini tidak sama di antara peserta sebelumnya dalam kelompok pengalaman peserta saat ini, peserta Para peserta ditugaskan ke salah satu kelompok yang kurang terwakili. Setelah ada distribusi perlakuan eksperimental yang sama dalam kelompok pengalaman, maka semua kelompok bebas untuk ditugaskan sampai setiap kelompok eksperimen diisi lagi. Hal itu dilakukan agar distribusi kelompok tetap merata.

p-ISSN : 2808-909X e-ISSN : 2798-2505

Blinding

Sementara Studi A dilakukan dengan setting double blind, replikasi ini dilakukan dengan setting single blind. Para peserta tidak tahu ke kelompok mana mereka ditugaskan, tetapi seorang moderator diperlukan untuk memastikan bahwa data pelacakan mata dikumpulkan dengan benar. Moderator ini tidak sengaja memberikan kelompok yang ditugaskan kepada peserta, tetapi moderator dapat melihat tugas yang diberikan kepada peserta dan menyimpulkan kelompok yang ditugaskan kepada peserta tersebut. Namun, untuk membatasi bias, moderator diinstruksikan untuk tidak mengungkapkan informasi apa pun terkait tugas atau kelompok yang ditugaskan kepada peserta. Para peserta tidak diberi informasi tentang kelompok tempat mereka ditugaskan atau hipotesis penelitian. Mereka hanya mengetahui informasi yang disajikan kepada mereka di lingkungan belajar. Selama persyaratan, peserta diberitahu bahwa mereka akan berpartisipasi dalam studi yang berkaitan dengan efek bahasa pemrograman pada programmer tetapi sifat studi yang tepat dan kelompok tempat mereka ditugaskan tidak disediakan.

ANALISIS DAN HASIL

Pra-Pemrosesan

Karena peserta harus menulis solusi mereka sendiri, cuplikan kode direkam dan dikirim ke server setiap 5 detik. Selain cuplikan kode, waktu peristiwa dari plugin iTrace-Chrome disimpan pada waktu yang sama untuk menyinkronkan data pelacakan mata dengan cuplikan ini. Setiap sampel tatapan dikaitkan dengan cuplikan kode menggunakan rekaman acara. Cuplikan kode yang terkait dengan pandangan pertama dalam xasi digunakan untuk mengasosiasikan setiap xation dengan cuplikan kode. Untuk menghitung xasi, filter IVT-xasi digunakan dengan ambang batas kecepatan 30 = detik dan jarak maksimum 75 ms. Kesenjangan yang lebih kecil diisi dengan interpolasi linier antara dua titik ujung celah.

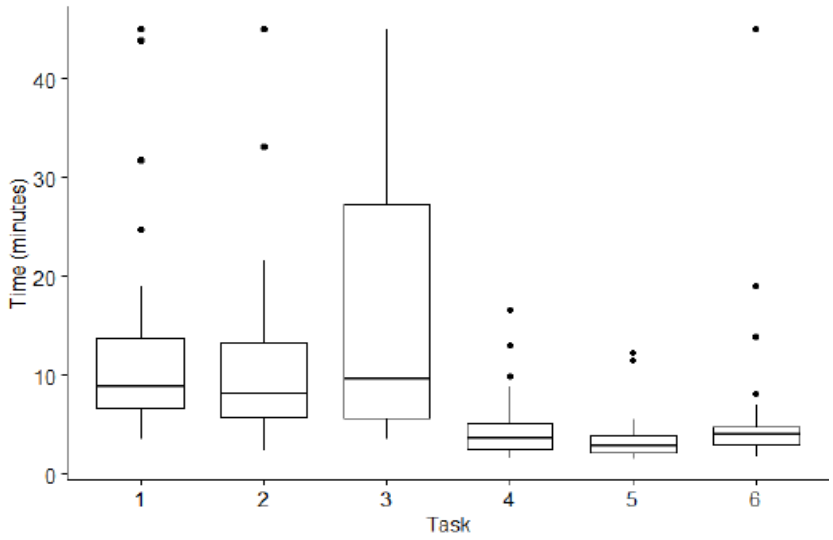
Hasil Produktivitas

Ada beberapa cara untuk mengukur produktivitas suatu program. Dalam konteks khusus ini, produktivitas diukur dengan waktu untuk menyelesaikan tugas. Perhatikan bahwa peserta tidak dapat melanjutkan ke tugas berikutnya sampai mereka menyelesaikannya dengan benar atau sampai 45 menit berlalu. Menjelajahi hasil untuk variabel dependen pertama (Waktu untuk Menyelesaikan), dan diketahui bahwa rata-rata peserta membutuhkan waktu 550,81 detik untuk menyelesaikan tugas dengan standar deviasi 614,93 detik dan waktu untuk menyelesaikan tugas berkisar antara 92,0 detik hingga 2702,0 detik yaitu lebih dari batas waktu 45 menit. ANOVA model campuran dijalan untuk membandingkan efek variabel antar kelompok kelompok dan pengalaman profesional dan variabel tugas dalam kelompok pada variabel terikat Waktu untuk Menyelesaikan. Kebulatan ditemukan dilanggar menurut Tes Mauchly. Jadi koreksi Greenhouse-Geisser standar dalam penelitian ini akan digunakan bila perlu.

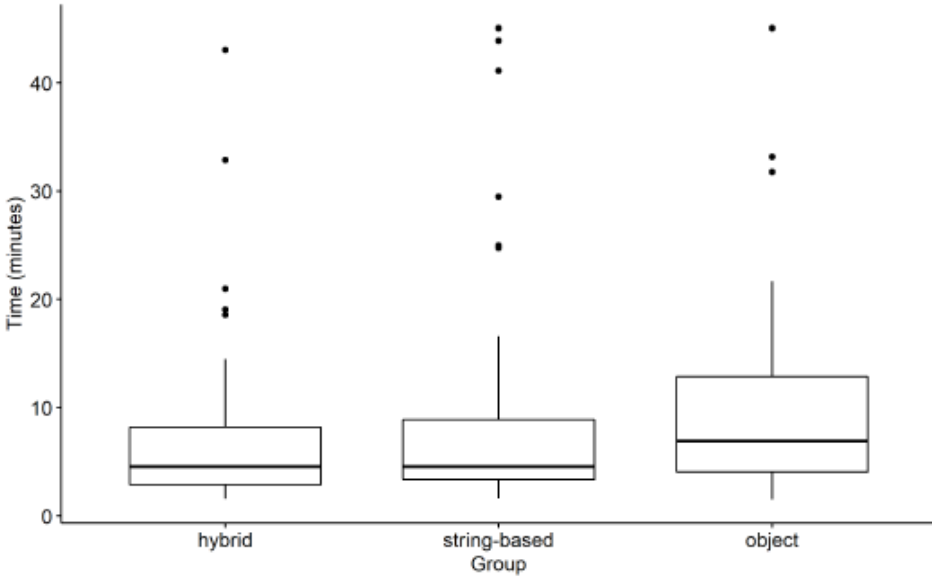
	Task 1	Task 2	Task 3	Task 4	Task 5
Task 2	1				
Task 3	1	0.295			
Task 4	0.295	0.0733	< 0.001		
Task 5	0.0126	0.0152	< 0.001	1	
Task 6	0.0733	0.411	< 0.001	1	1

Tabel 4: Hasil Uji-t Berpasangan Untuk Pengaruh Tugas terhadap Waktu Selesai

p-ISSN : 2808-909X e-ISSN : 2798-2505



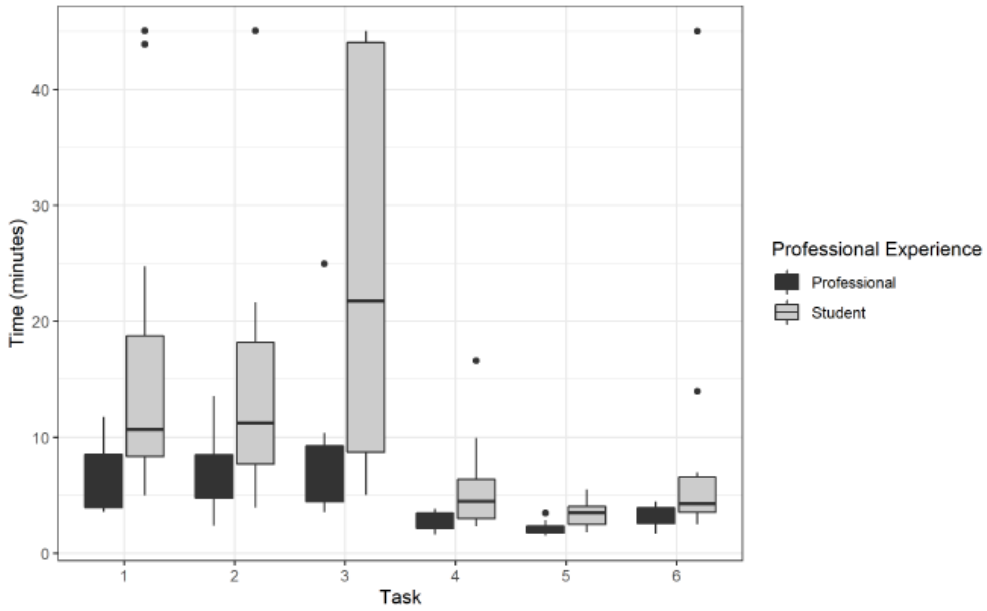
Gambar 5 Waktu Penyelesaian Untuk Setiap Tugas



Gambar 6 Waktu Penyelesaian Untuk Setiap Kelompok

Disini ditemukan bahwa tugas memiliki pengaruh yang signifikan terhadap waktu penyelesaian (p < 0:001), F(5; 95) = 16:2555. Karena percobaan tidak diimbangi, perbedaan waktu penyelesaian berdasarkan tugas ini bisa disebabkan oleh pemahaman jenis tugas yang lebih baik atau kesulitan tugas. Dalam upaya untuk memahami perbedaan antara tugas, analisis post-hoc digunakan untuk uji-t berpasangan dengan koreksi Bonferroni. Hasil uji-t berpasangan ini tercantum pada Tabel 4 dan menunjukkan bahwa tiga tugas pertama tidak memiliki perbedaan yang signifikan di antara mereka. Tiga tugas terakhir juga tidak memiliki perbedaan yang signifikan di antara mereka. Namun, sebagian besar dari tiga tugas pertama masing-masing membutuhkan waktu lebih lama untuk diselesaikan peserta daripada masing-masing dari tiga tugas terakhir. Perbedaan yang signifikan ini tidak ditemukan pada tiga perbandingan. Tugas 1 dibandingkan dengan tugas 4 serta tugas 6 dibandingkan dengan tugas 1 dan tugas 2. Perbedaan tersebut terlihat jelas pada Gambar 5. Akan tetapi, ketika efek untuk variabel antara subjek kelompok dicari justru ditemukan bahwa efek yang signifikan pada waktu penyelesaian (p = 0:2843); F(2; 19) = 1:3449 pada peserta kelompok yang ditugaskan tidak ditemukan. Meskipun Gambar 6 menunjukkan bahwa rata-rata kelompok hibrida membutuhkan waktu lebih lama, perbedaan ini tidak signifikan secara statistik.

p-ISSN : 2808-909X e-ISSN : 2798-2505



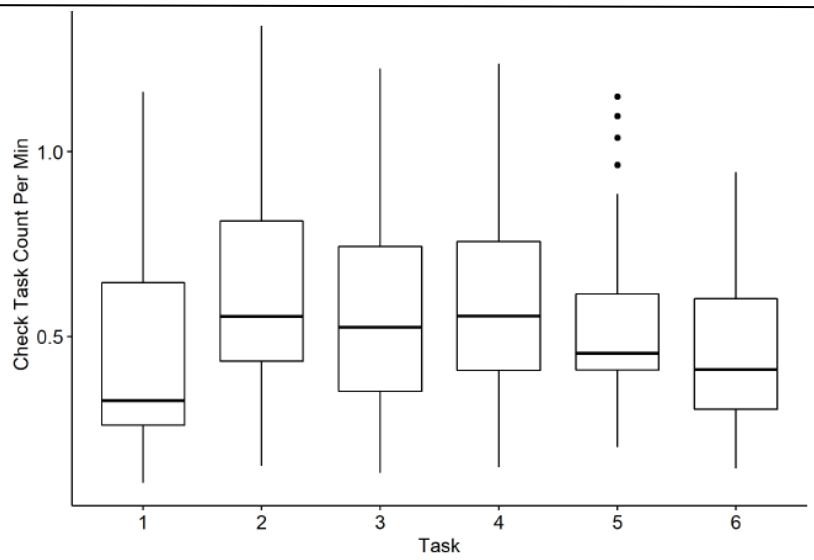
Gambar 7 Waktu Penyelesaian Untuk Setiap Tugas dan Tingkat Pengalaman Kerja

Hasil Pengalaman

Menjelajahi hasil model ANOVA lebih lanjut, efek profesional pengalaman tepat waktu untuk diselesaikan dapat dilihat. Ditemukan bahwa, peserta yang menggambarkan diri mereka sendiri sebagai program profesional membutuhkan waktu rata-rata 4,93 menit untuk menyelesaikan setiap tugas, dan non-profesional membutuhkan waktu rata-rata 12,1 menit untuk menyelesaikan setiap tugas. Selain itu, interaksi pengalaman profesional dan tugas memang memiliki pengaruh yang signifikan (p = 0:00435), F(5; 95) = 4:4964 yang menunjukkan bahwa tugas tertentu diselesaikan oleh peserta dengan tingkat pengalaman kerja yang tinggi. Mereka melakukan tugas rata-rata lebih cepat, tetapi tidak ada tugas khusus yang dibantu oleh pengalaman untuk diselesaikan lebih cepat daripada yang dijelaskan oleh tugas itu sendiri atau pengalaman kerja peserta. Metrik lain yang dapat di gunakan untuk membandingkan peserta adalah tingkat di mana mereka memeriksa solusi mereka dengan tombol periksa tugas. Jumlah kali peserta menjalankan solusi mereka dinormalisasi sesuai dengan waktu tugas sebagai tugas yang diambil lebih lama lebih mungkin untuk memeriksa solusi mereka beberapa kali. Model ANOVA campuran dijalankan untuk membandingkan efek antarkelompok variabel kelompok dan pengalaman profesional dan variabel dalam kelompok tugas pada variabel dependen Periksa Tingkat Tugas. Kebulatan ditemukan dilanggar menurut Tes Mauchly.

Perbedaan yang signifikan berdasarkan tugas yang diberikan kepada peserta F(5; 125) = 3:7094 (p = 0:00512) ditemukan, perbedaan ini dapat dilihat pada Gambar 8. Tugas 1 memiliki tingkat pengecekan tugas terendah yang kemungkinan besar karena peserta membutuhkan waktu lebih lama untuk menyesuaikan diri dengan lingkungan belajar. Namun, pengalaman profesional ditemukan tidak memiliki pengaruh yang signifikan terhadap tingkat tugas pemeriksaan F(1; 25) = 2:5274 (p = 0:1245). Sementara peserta yang ditugaskan ke varian API hybrid tampaknya memiliki tingkat pemeriksaan tugas yang sedikit lebih rendah, ini adalah perbedaan yang relatif kecil dan tidak signifikan. Interaksi antara pengalaman dan tugas F(5; 125) = 0:3000 (0:8969) dan pengalaman dan kelompok F(2; 25) = 0:4863 (0:6206) ternyata tidak signifikan. Dengan kesimpulan ini, dapat dilihat bahwa peserta dengan dan tanpa pengalaman profesional memeriksa tugas mereka dengan kecepatan yang kira-kira sama dalam tugas yang diberikan terlepas dari tugas yang mereka selesaikan atau varian API yang mereka gunakan.

p-ISSN : 2808-909X e-ISSN : 2798-2505



Gambar 8 Memeriksa Tingkat Tugas Berdasarkan Tugas

Gambar 9 AOI Tingkat Atas: Kode Contoh, Timer, Area Solusi, Tombol Periksa Tugas, dan Keluaran Tugas Ditampilkan

Hasil Perilaku Pandangan

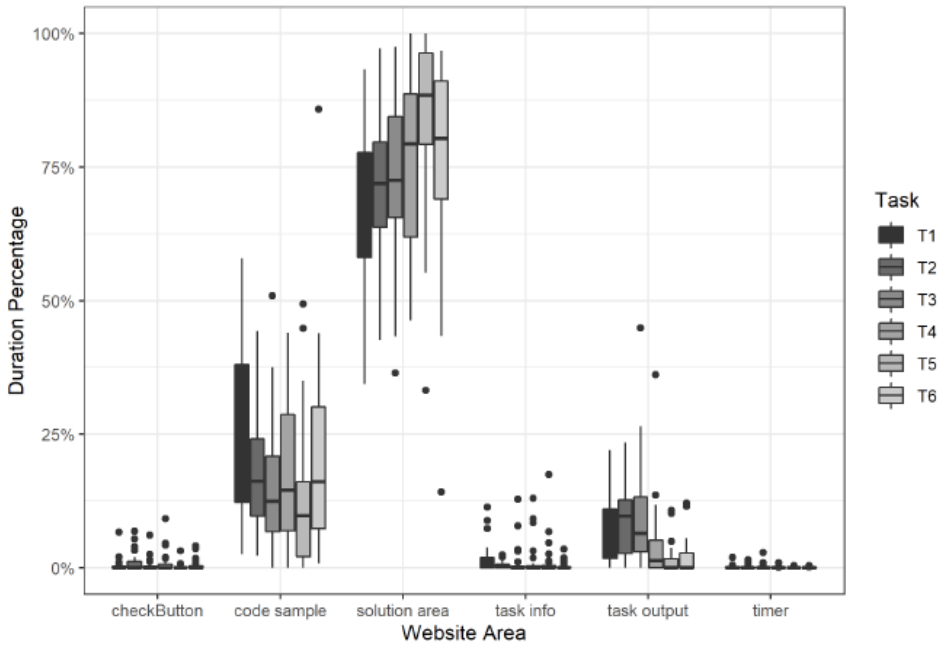
Tinjau Area Minat Tingkat Atas (AOI) di Situs Web Studi

Ada enam Area Minat tingkat atas, AOI, yang dilacak selama sesi pelacakan mata. Semua kecuali Info Tugas, yang menyediakan pesan tingkat tinggi untuk semua tugas, digambarkan dalam Gambar 9. Distribusi rata-rata keseluruhan xasi ditunjukkan pada Tabel 5. Ini menunjukkan bahwa sebagian besar waktu, peserta melakukan xating pada area solusi yang berisi kode yang mereka jalankan yang merupakan 74,76% dari durasi xation tugas. Kode sampel dan keluaran tugas adalah AOI lain yang memiliki durasi xasi dalam jumlah besar yang masing-masing menyumbang 18,30% dan 5,32% dari durasi xasi. AOI lain masing-masing menyumbang kurang dari 1% dari durasi xation kemungkinan besar karena sifat tambahan dari AOI ini. Untuk mengeksplorasi lebih lanjut perbedaan dalam distribusi AOI tingkat atas ini, dua ANOVA metode campuran dijalankan sehubungan dengan persentase durasi xation dan persentase jumlah fiksasi menggunakan variabel dalam subjek dari AOI tingkat atas, dan Tugas bersama dengan variabel antar-subjek dari kelompok.

Website Area	Duration (Visits)
Check Task Button	0.55%(0.52%)
Sample Code	18.3%(18.66%)
Solution Area	74.76%(74.15%)
Task Info	0.97%(0.95%)
Task Output	5.32%(5.63%)
Timer	0.1%(0.1%)

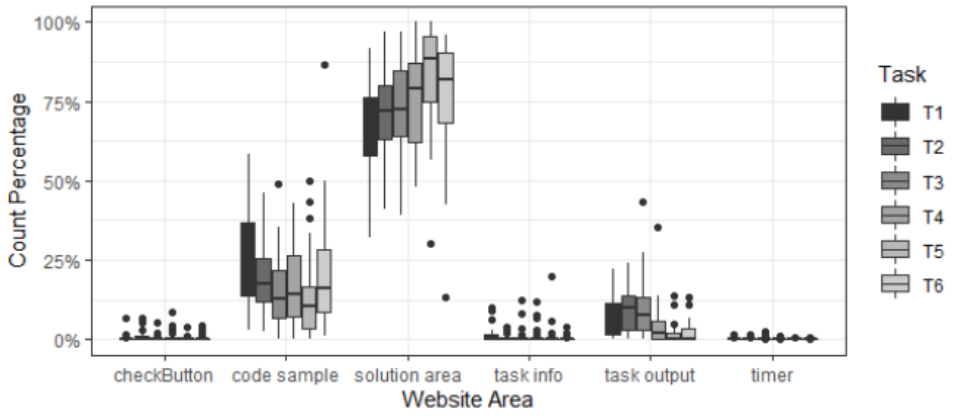
Tabel 5 Distribusi Rata-Rata Durasi Fiksasi Selama Semua Tugas di AOI Tingkat Atas, Distribusi Jumlah Fiksasi dalam Tanda Kurung

Pertama, tipe AOI berpengaruh signifikan terhadap persentase durasi F(5; 140) =22:85426 (p < 0:001). Ini berarti durasi waktu xating tidak dihabiskan secara merata di antara AOI level atas. Mengingat bahwa beberapa AOI menghabiskan sebagian besar waktu yang dihabiskan untuk xating sementara yang lain memiliki kurang dari satu persen waktu yang dihabiskan untuk xation, ini adalah efek langsung yang dapat dilihat. Saat melihat efek interaksi tipe AOI pada variabel lain, hasil yang lebih menarik dapat dilihat. Untuk interaksi tipe AOI dan Tugas, ditemukan efek signifikan F(25; 700) =7:1555 (p = 0:00137). Hal ini menunjukkan bahwa distribusi durasi xation tidak sama untuk setiap tugas. Seperti yang dapat dilihat pada Gambar 10, persentase durasi AOI tertentu tampaknya memiliki perubahan yang signifikan berdasarkan tugas.



Gambar 10 Persentase Durasi AOI Tingkat Atas Untuk Setiap Tugas

p-ISSN : 2808-909X e-ISSN : 2798-2505



Gambar 11 Hitung Persentase AOI Tingkat Atas Untuk Setiap Tugas

Untuk interaksi tipe AOI dan Grup, ditemukan bahwa tidak ada pengaruh yang signifikan F(10; 140) = 0:1102 (p = 0:8913). Hal ini menunjukkan bahwa peserta tidak melihat AOI secara berbeda berdasarkan varian API yang mereka gunakan. Selain itu, untuk interaksi tipe AOI, Grup, dan Tugas, efek yang signifikan F(50; 700) = 0:6312 (p = 0:6123) tidak ditemukan. Jadi, meskipun varian API tidak menyebabkan efek yang menyeluruh, hasil ini tampaknya mengindikasikan bahwa tugas tertentu berdampak pada distribusi durasi xation pada AOI ini. Melihat persentase hitungan, hasil yang mirip dilihat dengan persentase durasi tetapi dengan sedikit perbedaan. Pertama, tipe AOI berpengaruh signifikan terhadap persentase durasi F(5; 140) = 26:750 (p < 0:001). Ini berarti bahwa jumlah xasi tidak merata di antara AOI tingkat atas. Untuk interaksi jenis dan Tugas AOI, efek signifikan F(25; 700) = 7:672 (p < 0:001) ditemukan. Hal ini menunjukkan bahwa distribusi jumlah xasi tidak sama untuk setiap tugas.

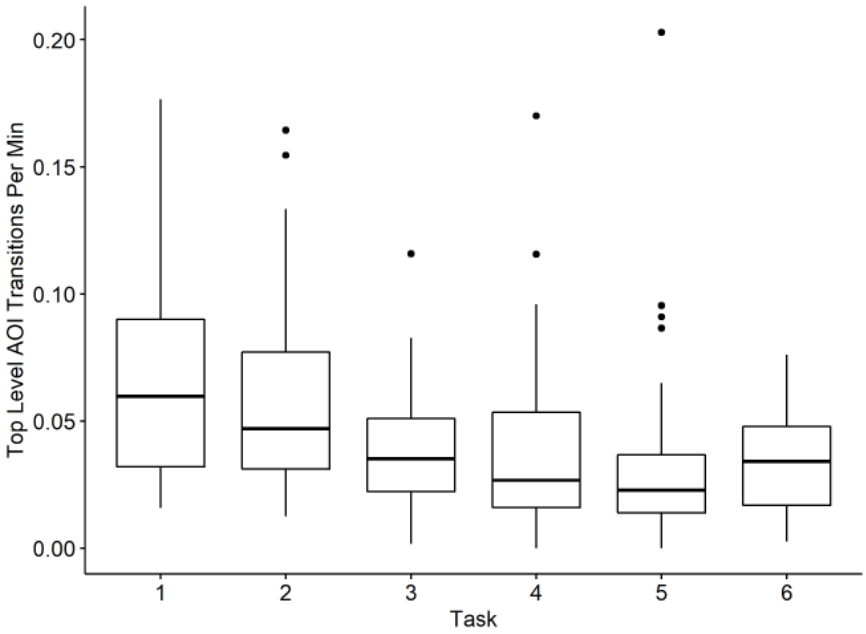
Persentase penghitungan AOI tertentu (gambar 11) tampaknya memiliki perubahan yang signifikan berdasarkan tugas dan mengikuti pola yang mirip dengan distribusi persentase durasi pada model sebelumnya. Untuk interaksi tipe AOI dan Grup, ditemukan bahwa tidak ada pengaruh yang signifikan F(10; 140) = 0:1492 (p = 0:8913). Hal ini menunjukkan bahwa peserta tidak melihat AOI secara berbeda berdasarkan varian API yang mereka gunakan. Selain itu, pada interaksi tipe AOI, Group, dan Task, pengaruh yang signifikan F(50; 700) = 0:7160 (p = 0:6123) tidak ditemukan meskipun pengaruh yang signifikan untuk persentase durasi ditemukan. Serupa dengan hasil sebelumnya, ini bisa menunjukkan bahwa tidak ada daya yang cukup dalam tes ini atau rata-rata durasi xation dipengaruhi oleh interaksi ini.

Menatap Transisi Antar Bidang Minat di Situs Web Studi

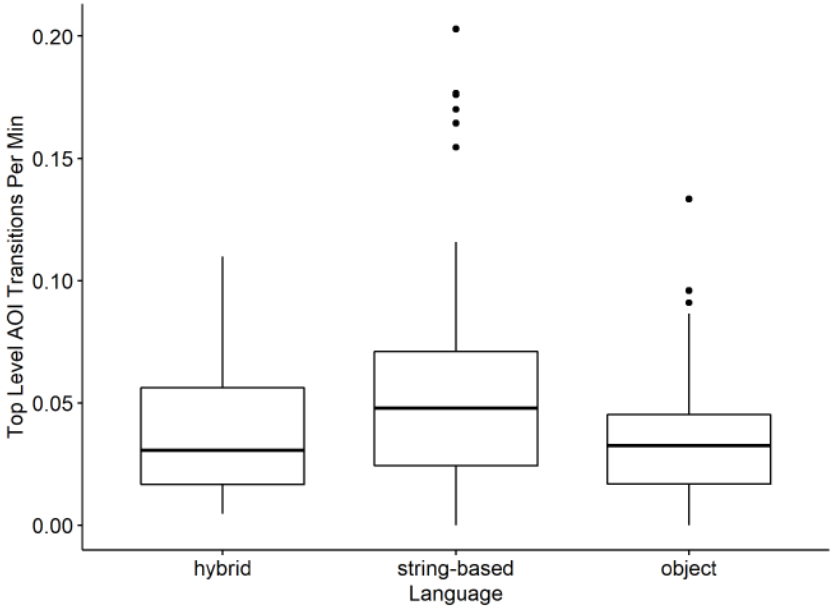
Untuk memahami bagaimana peserta menavigasi lingkungan belajar, transisi yang dilakukan peserta antara AOI tingkat atas diselidiki dalam penelitian ini (gambar 12). Karena waktu penyelesaian terbukti dipengaruhi oleh tugas, sehingga tingkat transisi disesuaikan cara dengan membaginya dengan waktu yang diperlukan untuk menyelesaikan tugas. Ini menghasilkan Transisi Per Menit AOI tingkat atas yang akan digunakan dalam analisis di bagian ini. Proses normalisasi data ini sangat penting untuk mendapatkan hasil yang benar. Untuk menjelajahi perbedaan dalam tingkat transisi AOI tingkat atas ini, ANOVA metode campuran dijalankan sehubungan dengan transisi AOI tingkat atas per menit menggunakan variabel Tugas dalam mata pelajaran bersama dengan variabel antara mata pelajaran kelompok dan pengalaman profesional.

Ditemukan bahwa tugas memiliki efek yang signifikan pada laju transisi AOI tingkat atas F(5; 125) = 9:1451 (p < 0:001). Tingkat transisi tertinggi dari AOI tingkat atas adalah untuk Tugas 1, kueri pemilihan sederhana. Tingkat transisi secara keseluruhan menurun saat peserta berpindah melalui tugas yang tersisa. Tingkat transisi AOI tingkat atas dapat dilihat untuk setiap tugas pada Gambar 12. Namun, variabel independen dan interaksi yang tersisa gagal untuk memiliki perbedaan yang signifikan. Pengalaman profesional F(1; 25) = 0:1517 (p = 0:7002), dan kelompok F(2; 25) = 0:8016 (p = 0:4598) tidak memiliki pengaruh yang signifikan terhadap tingkat transisi. Untuk grup, ditemukan bahwa peserta yang menggunakan API berbasis string melakukan lebih banyak transisi AOI tingkat atas dibandingkan dengan peserta yang menggunakan varian API hybrid atau berorientasi objek. Untuk perbedaan antara pengalaman profesional, perbedaan yang besar tidka terlihat, tetapi profesional tampaknya melakukan lebih banyak transisi antara AOI tingkat atas daripada nonprofesional. Namun, kedua perbedaan ini tidak signifikan.

p-ISSN : 2808-909X e-ISSN : 2798-2505



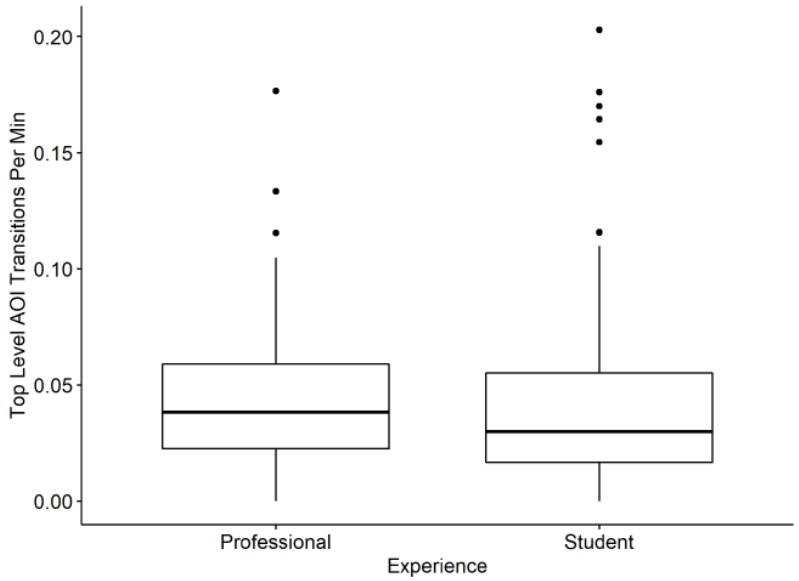
Gambar 12 Laju Transisi Tingkat Atas Berdasarkan Tugas



Gambar 13 Transisi Tingkat Atas Berdasarkan Grup

Seperti yang dapat dilihat pada Tabel 6 dan 7, transisi terbesar adalah dari area solusi ke kode sampel dan dari kode sampel ke area solusi. Tingkat transisi ini cukup mirip ketika membandingkan antara pergi ke dan dari area solusi yang menunjukkan bahwa sebagian besar transisi dari area solusi ke kode sampel memiliki transisi yang sesuai dari kode sampel ke area solusi. Tidak ditemukan banyak perbedaan antara perilaku navigasi tingkat atas berdasarkan pengalaman profesional yang didukung oleh hasil kuantitatif uji ANOVA yang dilakukan sebelumnya.

p-ISSN : 2808-909X e-ISSN : 2798-2505



Gambar 14 Tingkat Transisi Tingkat Atas Berdasarkan Pengalaman Profesional

From \ To	checkButton	code sample	solution area	task info	task output	timer
checkButton	0.00000	0.00010	0.00165	0.00000	0.00051	0.00002
code sample	0.00008	0.00000	0.02227	0.00022	0.00112	0.00002
solution area	0.00083	0.02255	0.00000	0.00125	0.00829	0.00039
task info	0.00000	0.00021	0.00073	0.00000	0.00000	0.00006
task output	0.00115	0.00116	0.00815	0.00001	0.00000	0.00000
timer	0.00000	0.00003	0.00027	0.00014	0.00001	0.00000

Tabel 6 Matriks Tingkat Transisi AOI Profesional Tingkat Atas

From \ To	checkButton	code sample	solution area	task info	task output	timer
checkButton	0.00000	0.00015	0.00131	0.00002	0.00030	0.00000
code sample	0.00008	0.00000	0.02159	0.00018	0.00103	0.00001
solution area	0.00098	0.02138	0.00000	0.00158	0.00642	0.00037
task info	0.00002	0.00012	0.00104	0.00000	0.00024	0.00008
task output	0.00057	0.00118	0.00625	0.00007	0.00000	0.00002
timer	0.00000	0.00003	0.00029	0.00013	0.00001	0.00000

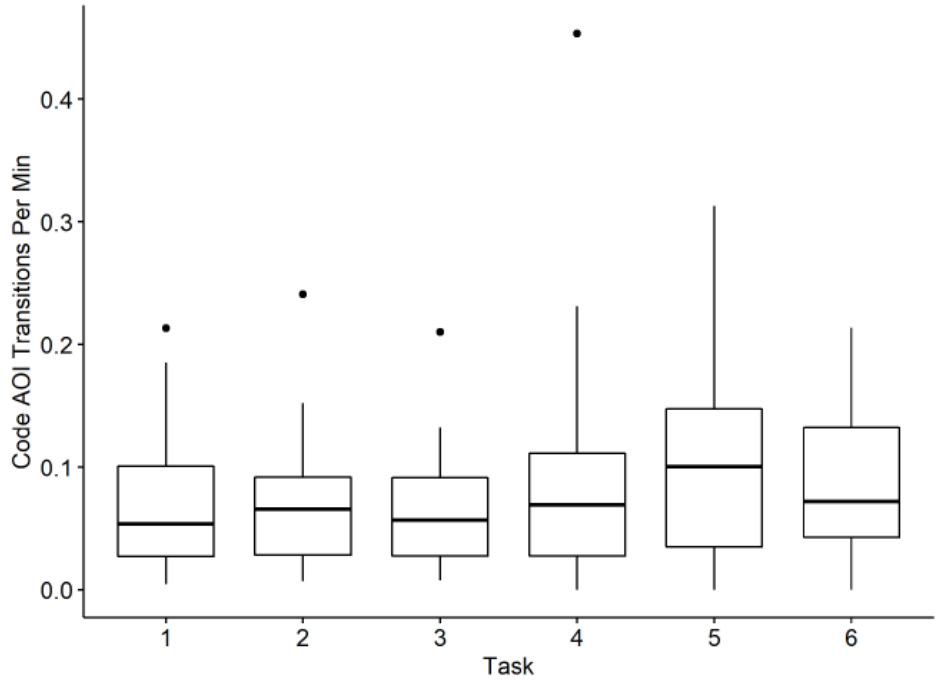
Tabel 7 Matriks Laju Transisi AOI Siswa Tingkat Atas

Transisi Tingkat Token

Selanjutnya, tingkat transisi antara elemen tingkat token dalam kode sumber yang terdapat dalam kotak teks di situs web dieksplorasi, dan tingkat alih kode ditentukan menjadi jumlah transisi antara elemen kode Java: pemanggilan metode, variabel deklarasi, deklarasi kelas, dll., dan elemen string di dalam kode Java. Untuk API berbasis string, transisi ini merepresentasikan pengalihan bahasa penuh dari Java ke SQL sedangkan untuk API berorientasi objek, ini hanya merepresentasikan peralihan dari kode Java ke parameter string metode. ANOVA metode campuran dijalankan sehubungan dengan transisi alih kode per menit menggunakan variabel Tugas dalam mata pelajaran bersama dengan variabel antara mata pelajaran kelompok dan pengalaman profesional. Ditemukan bahwa pengalaman profesional memiliki pengaruh yang signifikan F(1; 25) = 6:666 (p = 0:0161). Dalam Gambar 17 peserta melakukan lebih banyak transisi antara elemen kode Java dan parameter pemanggilan metode berbasis string. Hal ini menunjukkan bahwa peserta menavigasi kode berbeda dari pemula. Namun, penelitian ini gagal menemukan perbedaan yang signifikan antara tingkat alih kode berdasarkan tugas F(5; 125) = 2:350 (p = 0:0716). Pada Gambar 15 dapat dilihat bahwa meskipun tugas yang rata-rata membutuhkan waktu paling lama untuk diselesaikan peserta, tampaknya memiliki tingkat transisi tertinggi antara elemen-elemen ini dalam kode, sebagian besar tugas lainnya memiliki tingkat transisi yang sama antara satu sama lain.

p-ISSN : 2808-909X e-ISSN : 2798-2505

Hasil paling menarik yang ditemukan adalah grup memiliki sedikit pengaruh pada tingkat transisi antara token ini F(2; 25) = 0:2949 (p = 07471). Meskipun varian API dirancang untuk mensimulasikan tingkat peralihan bahasa yang berbeda, peralihan sebenarnya antara token bahasa host dan bahasa yang disematkan di dalam API yang diwakili oleh parameter menunjukkan sedikit variasi antara peserta yang menggunakan grup API yang berbeda. Kurangnya perbedaan (gambar 15) mungkin karena varian API tidak mewakili tingkat peralihan bahasa yang berbeda atau karena perbandingan parameter berbasis string dan kode Java mengukur perilaku yang berbeda antara varian API yang berbeda.

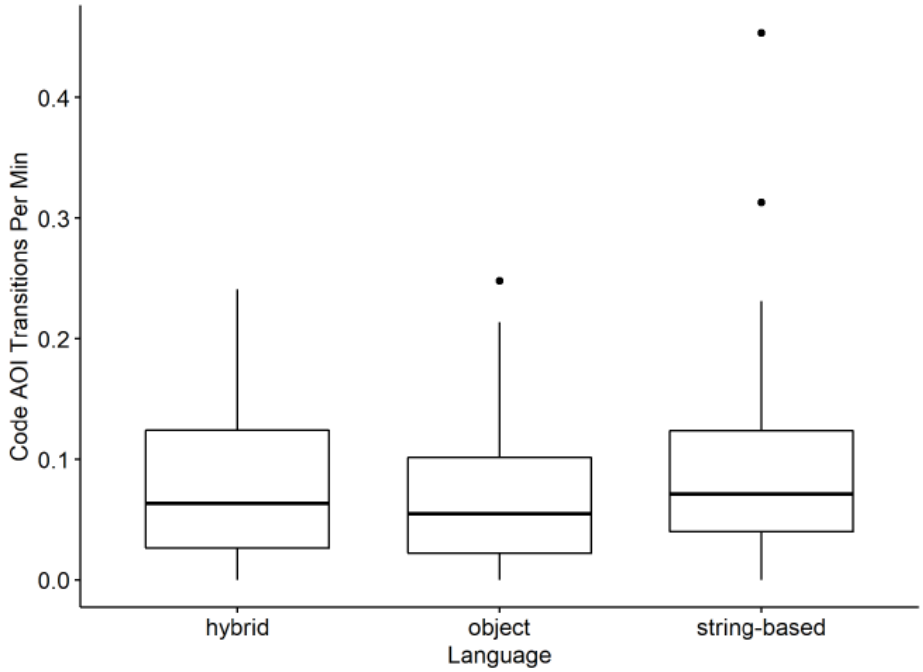


Gambar 15 Tingkat Alih Kode Berdasarkan Tugas

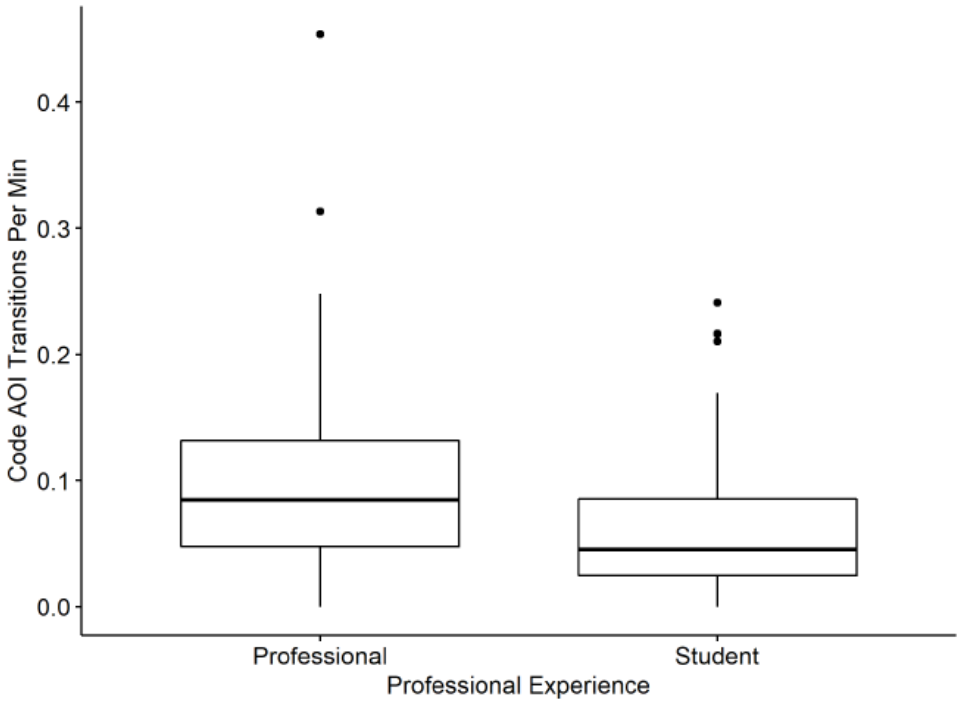
Ancaman terhadap Validitas

Dengan setiap studi empiris muncul beberapa potensi ancaman terhadap validitas. Ancaman terhadap validitas internal termasuk ancaman yang berpotensi mempengaruhi hubungan sebab-akibat yang coba diungkapkan oleh penelitian. Untuk dua populasi yang direkrut, lingkungan belajar yang berbeda digunakan. Peserta yang direkrut dari universitas mengikuti studi di dalam lab sementara peserta direkrut dari perusahaan lokal berpartisipasi dalam studi di dalam kantor kosong yang diubah menjadi lingkungan belajar. Perbedaan yang ditemukan antara kedua populasi ini dapat disebabkan oleh lingkungan belajar yang berbeda. Namun, ancaman ini dimitigasi dengan mengendalikan lingkungan studi di kedua lokasi dengan melakukan belajar di dalam ruangan tertutup, meminimalkan kebisingan di luar, dan menggunakan peralatan yang sama untuk melacak gerakan mata peserta dan melakukan penelitian.

p-ISSN : 2808-909X e-ISSN : 2798-2505



Gambar 16 Tingkat Alih Kode Berdasarkan Kelompok



Gambar 17 Tingkat Alih Kode Menurut Pengalaman Profesional

Ancaman terhadap validitas eksternal adalah sejauh mana hasil penelitian dapat digeneralisasikan ke dunia nyata. Tugas yang digunakan dalam penelitian ini relatif kecil, dan hasil yang di temukan mungkin tidak berlaku untuk tugas dunia nyata yang melibatkan banyak file dan persyaratan yang lebih kompleks. Selain itu, peserta ysng firekrut hanya dari dua populasi terpisah yang mungkin membatasi sejauh mana hasil dapat meluas ke populasi programmer secara umum. Terakhir, hasil yang ditemukan mengenai dampak peralihan bahasa pada perilaku pengembang dan produktivitas mungkin tidak meluas ke semua bentuk pemrograman poliglot seperti pemrograman poliglot tingkat proyek atau file. Bentuk lain dari pengalihan bahasa tertanam mungkin juga tidak mengikuti tren yang diamati pada peserta penelitian ini. Bahasa pemrograman Java dan SQL digunakan sebagai bahasa poliglot, dan hasil yang ditemukan mungkin

p-ISSN : 2808-909X e-ISSN : 2798-2505

tidak berlaku jika bahasa pemrograman yang digunakan memiliki tingkat kesamaan atau perbedaan yang berbeda. Misalnya, efek pemrograman poliglot menggunakan Java dan C yang merupakan bahasa serupa mungkin berbeda dari efek pemrograman poliglot menggunakan Java dan Haskell.

Ancaman terhadap validitas konstruk adalah sejauh mana suatu tes mengukur apa yang diklaim, atau dimaksudkan, untuk diukur. Ancaman pertama untuk membangun validitas adalah asumsi bahwa waktu untuk menyelesaikan tugas adalah cara yang akurat untuk mengukur efisiensi programmer. Meskipun solusi tercepat dapat dianggap efisien, solusi tersebut mungkin tidak sekuat atau dapat diperluas seperti solusi yang membutuhkan waktu lebih lama untuk dibuat. Mengingat sifat kecil dari tugas yang digunakan dalam penelitian ini, waktu untuk menyelesaikan tugas adalah ukuran yang tepat untuk memperkirakan efisiensi programmer. Sementara varian API yang digunakan dalam penelitian ini dimaksudkan untuk mengubah tingkat peralihan bahasa dan mensimulasikan berbagai tingkat pemrograman poliglot tersemat, perbedaan kinerja pengembang antar API mungkin terkait dengan desain spesifik dari API yang digunakan dalam penelitian ini. Ancaman ini dimitigasi dengan memiliki API terbatas dengan hanya sejumlah kecil fungsi dan metode yang berarti bahwa lebih sedikit desain API yang harus diubah di antara setiap varian. Ancaman terhadap validitas kesimpulan adalah sejauh mana kesimpulan yang dicapai tentang hubungan dalam data yang masuk akal. Untuk mengurangi ancaman ini, maka pengujian statistik yang sesuai digunakan dan koreksi yang diperlukan dilakukan untuk memastikan bahwa statistik inferensial penelitian ini benar.

DISKUSI DAN IMPLIKASI

Implikasi pada Produktivitas

Tidak seperti Studi A, peserta kelompok yang ditempatkan tidak memiliki pengaruh yang signifikan terhadap efisiensi pemrogram. Perbedaan antara waktu untuk menyelesaikan tugas berdasarkan varian API yang digunakan programmer mengikuti pola yang sama yang diamati dalam Studi A menunjukkan bahwa ukuran sampel replikasi tidak cukup besar untuk mengklasifikasikan perbedaan itu ada di antara grup API, tetapi perbedaan mendasar mungkin masih ada. Beberapa perbedaan besar ditemukana diantara antara tugas-tugas yang diselesaikan programmer, dan perbedaan ini memiliki efek yang lebih besar pada waktu yang dibutuhkan peserta untuk berhasil menyelesaikan tugas daripada varian API yang digunakan peserta. Hal ini menunjukkan bahwa kerumitan dan kesulitan tugas yang coba diselesaikan oleh seorang programmer akan berdampak lebih besar pada produktivitas programmer daripada tingkat pemrograman poliglot yang digunakan pemrogram.

Implikasi pada Pengalaman

Perbadaan juga ditemukan dalam pengaruh programmer profesional atau peserta biasa terhadap waktu yang dibutuhkan untuk menyelesaikan tugas. Akan tetapi perbedaan ini tidak mengherankan mengingat programmer dengan pengalaman lebih banyak harus dapat menyelesaikan tugas lebih cepat daripada programmer dengan pengalaman lebih sedikit. Hasil ini setuju dengan hasil yang ditemukan dalam Studi A. Sementara pengalaman umum tampaknya berpengaruh pada produktivitas dalam tugas pemrograman polyglot, penelitian ini tidak dapat menemukan perbedaan dalam produktivitas profesional dan programmer non-profesional berdasarkan varian API yang mereka gunakan. Ini menyiratkan bahwa tingkat pengalaman programmer lebih penting daripada tingkat pemrograman polyglot yang digunakan. Programmer yang lebih berpengalaman akan lebih produktif terlepas dari tingkat peralihan bahasa yang digunakan dalam tugas pemrograman.

Implikasi pada Tatapan Perilaku

Tidak ditemukan perbedaan yang signifikan dalam perilaku navigasi tatapan tingkat atas peserta berdasarkan kelompok tempat mereka ditugaskan atau pengalaman mereka. Hal ini menunjukkan bahwa peserta mendekati tugas dengan strategi tingkat tinggi yang sama terlepas dari tingkat peralihan bahasa yang harus mereka lakukan. Perilaku navigasi tingkat atas ditemukan berubah berdasarkan tugas dengan tugas di awal penelitian memiliki tingkat transisi tingkat atas yang lebih besar daripada tugas menjelang akhir penelitian. Hal ini dapat menunjukkan bahwa peserta yang lebih terbiasa dengan tugas melakukan navigasi tingkat atas yang lebih sedikit atau bisa jadi karena perbedaan dalam tugas itu sendiri. Ditemukan juga bahwa tugas tersebut sendiri memiliki perbedaan yang signifikan berdasarkan tugas mana yang digunakan peserta. Namun karena tugas disajikan dalam urutan yang sama, ada kemungkinan bahwa ini kurang berkaitan dengan strategi tingkat tinggi yang digunakan peserta untuk setiap tugas yang diberikan dan lebih berkaitan dengan penempatan tugas dalam penelitian. Perbedaan yang signifikan juga ditemukan

p-ISSN : 2808-909X e-ISSN : 2798-2505

dalam perilaku transisi level token berdasarkan pengalaman. Hal ini menunjukkan bahwa para ahli dan pemula menggunakan strategi yang berbeda untuk membaca kode selama tugas pemrograman polyglot. Tidak diketahui apakah perbedaan ini disebabkan oleh pemrograman poliglot itu sendiri atau perbedaan yang sama yang ada selama tugas yang melibatkan satu bahasa pemrograman.

KESIMPULAN DAN PEKERJAAN SELANJUTNYA

Dalam penelitian ini, hasil studi replikasi yang mengukur pengaruh pemrograman polyglot pada perilaku programmer disajikan. Ini adalah salah satu studi pertama yang menggunakan pelacakan mata sebagai metodologi dalam tugas pemrograman poliglot. Para peserta dibagi menjadi tiga kelompok: kelompok polyglot berbasis string, sebuah bjectoriented kelompok monoglot, dan kelompok hibrid. Hasil menunjukkan bahwa peserta dengan pengalaman profesional mampu menyelesaikan tugas pemrograman poliglot lebih efisien daripada rekan siswa mereka terlepas dari kelompok tempat mereka ditempatkan. Berbeda dengan penelitian asli, replikasi ini tidak dapat menemukan hasil yang signifikan efek kelompok tempat peserta ditempatkan untuk efisiensi programmer menyelesaikan tugas. Hasil juga menunjukkan perilaku navigasi programmer sebagian besar tetap tidak berubah dibandingkan dengan kelompok tempat mereka ditempatkan atau pengalaman yang dimiliki peserta. Sebagai pekerjaan di masa depan, efek dari berbagai jenis pemrograman polyglot bahasa harus dipelajari lebih mendalam lagi untuk melihat apakah kesimpulan yang dicapai dalam makalah ini tetap benar di berbagai konteks pemrograman poliglot. Studi lebih lanjut dalam tatapan perilaku juga perlu dilakukan untuk menentukan dampak yang tepat dari kesulitan tugas dan perilaku navigasi pemrogram.

DAFTAR PUSTAKA

A. Stefik and S. Siebert. An empirical investigation into programming language syntax. Trans. Comput. Educ., 13(4):19:1{19:40, Nov. 2013.

A. Van Deursen, P. Klint, and J. Visser. Domain-specific languages: An annotated bibliography. ACM Sigplan Notices, 35(6):26-36, 2000.

A. Vetro, F. Tomassetti, M. Torchiano, and M. Morisio. Language interaction and quality issues: an exploratory study. In Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, pages 319-322. ACM, 2012.

B. A. Becker. A new metric to quantify repeated compiler errors for novice programmers. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, pages 296{301. ACM, 2016.

B. Sharif, M. Falcone, and J. I. Maletic. An eye-tracking study on the role of scan time in finding source code defects. In Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA '12, pages 381{384, New York, NY, USA, 2012. ACM.

F. Tomassetti and M. Torchiano. An empirical assessment of polyglot-ism in github. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, page 17. ACM, 2014.

H. Uwano, M. Nakamura, A. Monden, and K. ichi Matsumoto. Analyzing individual performance of source code review using reviewers' eye movement. In Proceedings of the 2006 Symposium on Eye Tracking Research &Amp; Applica- tions, ETRA '06, pages 133{140, New York, NY, USA, 2006. ACM.

H.-C. Fjeldberg. Polyglot programming. a business perspective. PhD thesis, Master thesis, Norwegian University of Science and Technology, 2008.

J. Siegmund, N. Peitek, C. Parnin, S. Apel, J. Hofmeister, C. K • astner, A. Begel, A. Bethmann, and A. Brechmann. Measuring neural efficiency of program comprehension. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pages 140{150. ACM, 2017.

J. Stylos and B. A. Myers. The implications of method placement on api learnability. In Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, pages 105{112. ACM, 2008.

K. Kevic, B. M. Walters, T. R. Sha er, B. Sharif, D. C. Shepherd, and T. Fritz. Tracing software developers' eyes and interactions for change tasks. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pages 202{213. ACM, 2015.

p-ISSN : 2808-909X e-ISSN : 2798-2505

L. A. Meyerovich and A. S. Rabkin. Empirical analysis of programming language adoption. SIGPLAN Not., 48(10):1{18, Oct. 2013.

M. E. Crosby and J. Stelovsky. How do we read algorithms? a case study. Computer, 23(1):25{35, 1990.

M. Hoppe and S. Hanenberg. Do developers benefit from generic types?: an empirical comparison of generic and raw types in java. In Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, OOPSLA 2013, part of SPLASH 2013, Indianapolis, IN, USA, October 26-31, 2013, pages 457{474. ACM, 2013.

M. Konopka. Combining eye tracking with navigation paths for identi cation of cross-language code dependencies. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, page 1057{1059, New York, NY, USA, 2015. Association for Computing Machinery.

P. Mayer and A. Bauer. An empirical analysis of the utilization of multiple programming languages in open source projects. In Proceedings of the 19th In- ternational Conference on Evaluation and Assessment in Software Engineering, page 4. ACM, 2015.

P. Mayer, M. Kirsch, and M. A. Le. On multi-language software development, cross-language links and accompanying tools: a survey of professional software developers. Journal of Software Engineering Research and Development, 5(1):1, 2017.

P. Rodeghero and C. McMillan. An empirical study on the patterns of eye movement during summarization tasks. In 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), volume 00, pages 1{10, Oct. 2015.

P. Rodeghero, C. McMillan, P. W. McBurney, N. Bosch, and S. D'Mello. Improving automated source code summarization via an eye-tracking study of programmers. In Proceedings of the 36th international conference on Software en- gineering, pages 390{401, 2014.

R. E. Brooks. Towards a theory of the comprehension of computer programs. Intl J. of Man-Machine Studies, 18(6):543{554, 1983.

T. Busjahn, R. Bednarik, A. Begel, M. Crosby, J. H. Paterson, C. Schulte, B. Sharif, and S. Tamm. Eye movements in code reading: Relaxing the linear order. In 2015 IEEE 23rd International Conference on Program Comprehension, pages 255{265, May 2015.