**Jurnal Sistem Informasi dan Ilmu Komputer**
**Volume. 2, Nomor. 4 Tahun 2024**
e-ISSN: 2986-4976; dan p-ISSN: 2986-5158, 202-212
DOI: https://doi.org/10.59581/jusiik-widyakarya.v2i4.4558
*Available online at*: https://journal.widyakarya.ac.id/index.php/jusiik-widyakarya

# Strategi Teknik *Debugging* untuk Sistem Operasi *Windows*

**Herlan Wibowo[1*], Muhamad Faisal Ilham[2], Elkin Rilvani[3]**
[1,2,3]Universitas Pelita Bangsa, Indonesia
*Email:* [1]herlanwib@mhs.pelitabangsa.ac.id, [2]muhamadfaisallilham@gmail.com, [3]elkin.rilvani@pelitabangsa.ac.id

Alamat : Jl. Inspeksi Kalimalang No.9, Cibatu, Cikarang Sel., Kabupaten Bekasi, Jawa Barat 17530
*Korespondensi penulis:* herlanwib@mhs.pelitabangsa.ac.id*

***Abstract****. The debugging process is a critical step in software development, especially for the Windows operating system, which offers various tools and techniques to address software issues. This study explores debugging strategies such as logging, memory analysis, real-time debugging, reverse engineering, and compilation-based techniques to identify and resolve problems such as memory leaks and unresponsive applications. Results indicate that selecting the right debugging method depends on the type and complexity of the problem, with a combination of methods often yielding the best outcomes. This study provides practical guidance for Windows users to enhance debugging effectiveness in resolving system and application issues.*

***Keywords****: Debugging, Windows, Logging, Memory Analysis, Compilation Techniques.*

**Abstrak**. Proses debugging merupakan langkah krusial dalam pengembangan perangkat lunak, terutama untuk sistem operasi Windows yang menyediakan berbagai alat dan teknik untuk mengatasi masalah perangkat lunak. Penelitian ini membahas strategi debugging seperti logging, analisis memori, debugging real-time, reverse engineering, dan teknik berbasis kompilasi, yang digunakan untuk mengidentifikasi dan menyelesaikan masalah seperti kebocoran memori dan aplikasi tidak merespons. Hasil penelitian menunjukkan bahwa pemilihan metode debugging yang tepat bergantung pada jenis dan kompleksitas masalah, dengan kombinasi metode sering kali memberikan hasil terbaik. Studi ini memberikan panduan praktis bagi pengguna Windows untuk meningkatkan efektivitas debugging dalam memperbaiki masalah sistem dan aplikasi.

**Kata kunci**: Debugging, Windows, Logging, Analisis Memori, Teknik Kompilasi.

## 1. LATAR BELAKANG

Debugging adalah proses penting dalam pengembangan perangkat lunak yang bertujuan untuk menemukan dan memperbaiki kesalahan atau bug dalam kode program. Proses ini tidak hanya diperlukan untuk aplikasi yang baru dibuat tetapi juga untuk perangkat lunak yang sudah ada agar tetap berfungsi dengan baik. Sistem operasi Windows menyediakan berbagai fasilitas untuk membantu proses debugging. Windows memiliki tools bawaan dan API seperti Event Viewer, Windows Debugger (WinDbg), dan Visual Studio Debugger yang mendukung proses debug, terutama dalam menangani aplikasi yang mengalami kegagalan atau error. Fasilitas ini memungkinkan pengembang atau pengguna tingkat lanjut untuk melihat pesan kesalahan, melacak penyebab masalah, serta mengidentifikasi solusi yang tepat untuk memulihkan fungsi aplikasi.

Salah satu masalah utama yang sering dihadapi dalam sistem operasi Windows adalah aplikasi yang tiba-tiba tidak berfungsi atau berhenti secara mendadak, yang dikenal sebagai “Application Not Responding” atau “Aplikasi Tidak Berfungsi.” Masalah ini

*Received: Oktober 31, 2024; Revised: November 14, 2024; Accepted: November 28, 2024; Published: November 30, 2024*

sering kali menimbulkan gangguan signifikan bagi pengguna, terutama jika terjadi pada aplikasi kritis. Pada penelitian ini untuk menyelesaikan masalah tersebut maka disusunlah langkah peneletian dalam bentuk pertanyaan bagaimana menyelesaikan permasalahan aplikasi tidak berfungsi dengan teknik debugging yang tepat untuk mengatasi permasalahan dalam system operasi windows.

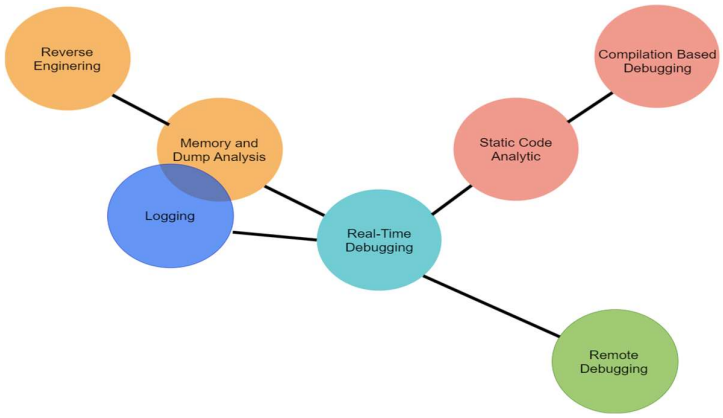
Manfaat dari tools debugging yang disediakan oleh Windows sangatlah besar. Dengan tools ini, pengguna dan pengembang dapat mendeteksi penyebab utama dari masalah aplikasi, menghemat waktu dalam mengatasi kesalahan, serta mengurangi risiko gangguan lebih lanjut pada sistem. Melalui penelitian ini, diharapkan dapat memberikan panduan praktis bagi pengguna Windows dalam menerapkan teknik debugging yang efektif untuk menangani aplikasi yang tidak berfungsi dan memperpanjang umur operasional aplikasi tersebut dalam sistem operasi Windows.

## 2. KAJIAN TEORITIS

### Konsep Dasar Debugging

Definisi Debugging adalah proses deteksi dan perbaikan kesalahan pada perangkat lunak. Peran Debugging pada Sistem Operasi Windows, debugging digunakan baik oleh pengembang aplikasi maupun tim IT untuk memastikan bahwa perangkat lunak berjalan tanpa gangguan.

### Strategi Teknik Debugging untuk Sistem Operasi Windows




**Gambar 1.** Relasi Antar Teknik Debugging

Setiap teknik debugging memiliki kelebihan dan cocok untuk berbagai jenis masalah yang ditemukan di Windows. Berikut adalah beberapa teknik debugging yang umum digunakan, bersama dengan penjelasannya:

Compilation-Based Debugging adalah teknik yang melibatkan penyertaan informasi debugging saat proses kompilasi program. Dengan menambahkan tanda-tanda debug di dalam kode, pengembang dapat melacak sumber kesalahan secara lebih mudah ketika program dijalankan. Teknik ini sangat berguna saat pengembang menggunakan alat pengembangan yang memungkinkan opsi "Debug Build", seperti di Visual Studio, yang mengompilasi kode dengan informasi tambahan yang dapat digunakan dengan debugger. Teknik ini mempermudah pelacakan kesalahan di dalam kode sumber dan mempercepat identifikasi letak masalah, menjadikannya langkah awal yang efektif dalam proses debugging.

Logging adalah metode debugging yang menggunakan pernyataan log untuk mencatat alur dan status eksekusi kode selama runtime. Log biasanya dicatat ke dalam file teks, event viewer, atau database, memungkinkan pengembang untuk melacak bagaimana program berjalan dan bagaimana variabel berubah. Contoh penggunaannya adalah dengan menggunakan OutputDebugString di Windows untuk mengirim pesan ke debugger atau mencatat informasi ke dalam Event Viewer. Teknik ini membantu pengembang dalam melacak alur eksekusi program dan kondisi variabel pada runtime, sangat berguna untuk masalah yang sulit direproduksi di lingkungan pengembangan.

Real - Time Debugging melibatkan penggunaan alat yang memungkinkan pengembang menjalankan program secara step-by-step, menghentikan eksekusi pada breakpoint tertentu, serta memeriksa nilai variabel dan status program pada titik-titik tersebut. Windows menyediakan beberapa debugger seperti WinDbg dan Visual Studio Debugger, yang populer digunakan untuk debugging aplikasi atau sistem berbasis Windows. Teknik ini memungkinkan pengembang menangani masalah secara langsung dalam kode program dan sangat efektif dalam menemukan bug spesifik pada kode yang kompleks, membantu memastikan program berjalan sesuai harapan.

Memory and Dump Analysis adalah teknik yang digunakan untuk memeriksa memori pada saat program mengalami kegagalan atau crash. Dump file, yang merupakan snapshot dari memori sistem pada waktu tertentu, dapat dianalisis untuk menemukan penyebab crash dan potensi masalah dalam program. Contoh implementasinya termasuk penggunaan Windows Error Reporting (WER) yang menghasilkan file dump, yang kemudian bisa dianalisis menggunakan alat seperti WinDbg. Teknik ini sangat berguna dalam menganalisis masalah yang hanya muncul dalam lingkungan runtime, terutama ketika sistem tidak merespons atau mengalami kegagalan total.

Static Code Analysis adalah proses evaluasi kode sumber tanpa menjalankan program. Penggunaan alat seperti linting atau scanning tools membantu pengembang menemukan kesalahan atau potensi masalah dalam kode. Contoh alat yang digunakan termasuk Visual Studio Code Analysis atau SonarQube. Keunggulan utama dari teknik ini adalah kemampuannya untuk menemukan kesalahan sintaksis dan potensi bug sebelum kode dikompilasi atau dieksekusi. Hal ini mempercepat pengembangan dengan memastikan bahwa kode sudah bebas dari kesalahan dasar sebelum tahap testing lebih lanjut.

Reverse Engineering adalah teknik yang bermanfaat ketika kode sumber tidak tersedia, atau ada kebutuhan untuk memahami cara kerja aplikasi pihak ketiga atau program yang mempengaruhi sistem. Dengan menggunakan alat seperti IDA Pro atau x64dbg, pengembang dapat mempelajari bagaimana aplikasi beroperasi pada tingkat assembly. Teknik ini sangat efektif dalam menganalisis aplikasi pihak ketiga atau malware yang tidak memberikan akses ke kode sumber, memungkinkan pengembang mempelajari struktur dan perilaku program tersebut.

Remote Debugging adalah teknik debugging yang digunakan untuk memecahkan masalah aplikasi yang berjalan pada perangkat atau sistem jarak jauh. Remote debugging memungkinkan pengembang untuk mengontrol debugger di komputer lain yang menjalankan aplikasi. Contoh penerapannya adalah Visual Studio yang mendukung remote debugging dengan menghubungkan debugger di satu mesin ke aplikasi di mesin lain. Teknik ini sangat berguna dalam pengembangan perangkat lunak yang perlu diuji di berbagai konfigurasi sistem operasi atau perangkat keras, membantu memastikan program dapat berjalan dengan baik di lingkungan yang berbeda.

**Pemilihan Teknik Debugging Berdasarkan Kebutuhan**

Tidak ada satu teknik yang cocok untuk semua situasi debugging, sehingga pemilihan metode yang tepat sangat bergantung pada berbagai faktor. Salah satu faktor utama adalah jenis dan kompleksitas masalah yang dihadapi. Masalah yang lebih kompleks seringkali memerlukan analisis mendalam seperti debugging real-time atau analisis memori untuk menemukan akar permasalahan secara menyeluruh. Ketersediaan sumber daya dan alat juga mempengaruhi pemilihan teknik; misalnya, dump file dapat dengan mudah dihasilkan saat terjadi crash, tetapi tidak selalu mencakup semua konteks eksekusi program, sehingga mungkin memerlukan alat tambahan untuk analisis lebih lanjut. Selain itu, waktu yang tersedia untuk proses debugging juga menjadi pertimbangan

penting. Teknik logging mungkin lebih cepat dan mudah diimplementasikan untuk solusi langsung, sementara reverse engineering memerlukan lebih banyak waktu dan keterampilan karena kompleksitasnya. Kombinasi faktor-faktor ini menentukan metode debugging yang paling sesuai untuk menyelesaikan masalah tertentu.

**Tantangan dalam Debugging pada Windows**

Kompleksitas sistem pada Windows membuat debugging menjadi tantangan tersendiri, karena sistem operasi ini memiliki banyak proses latar belakang dan aplikasi yang saling bergantung satu sama lain. Oleh karena itu, debugging harus dilakukan dengan hati-hati agar tidak menimbulkan pengaruh negatif pada keseluruhan sistem. Selain itu, masalah hak akses sering kali menjadi kendala, karena pada beberapa kasus, debugger memerlukan hak akses tinggi untuk menjalankan fungsinya. Hal ini berarti bahwa proses debugging di Windows harus dilakukan dengan konfigurasi izin yang tepat untuk memastikan kelancaran dan keamanan proses analisis.

## 3. METODE PENELITIAN

Penelitian ini menggunakan pendekatan deskriptif kualitatif dengan menerapkan berbagai strategi teknik debugging pada aplikasi simulasi yang mengalami permasalahan. Strategi-strategi ini diuji untuk mengevaluasi efektivitasnya dalam menyelesaikan masalah aplikasi pada sistem operasi Windows.

Tahapan Penelitian

a. Studi Literatur
    a. Menganalisis literatur terkait strategi debugging, termasuk dokumentasi teknis, artikel jurnal, dan buku seperti *The Pragmatic Programmer*.
    b. Memfokuskan penelitian pada metode debugging populer: debugging berbasis kompilasi, teknik logging, debugging real-time, analisis memori, reverse engineering, dan static code analysis.
b. Pengujian Teknik Debugging
    a. Setiap teknik debugging diterapkan pada aplikasi simulasi yang menghadapi permasalahan seperti tidak merespons, penurunan performa, dan kebocoran memori.
    b. Strategi yang diuji mencakup:
        1) Debugging Berbasis Kompilasi: Menyertakan informasi debugging saat proses kompilasi untuk melacak kesalahan lebih efektif.

2) Teknik Logging: Menggunakan log runtime untuk memantau alur eksekusi program.
3) Debugging Real-Time: Melakukan analisis step-by-step dengan breakpoint untuk mengidentifikasi bug dalam kode sumber.
4) Analisis Memori dan Dump File: Mengevaluasi kondisi memori untuk menemukan penyebab crash atau kebocoran memori.
5) Static Code Analysis: Memeriksa kode sumber menggunakan linting untuk mendeteksi kesalahan sebelum runtime.
6) Reverse Engineering: Menganalisis aplikasi pihak ketiga tanpa kode sumber.

c. Eksperimen Studi Kasus
   a. Studi kasus dilakukan pada aplikasi nyata yang menghadapi masalah performa atau stabilitas.
   b. Mengimplementasikan kombinasi teknik debugging untuk menemukan akar masalah, menganalisis memori, dan memperbaiki bug.
   c. Teknik real-time debugging dan logging runtime digunakan untuk memantau alur eksekusi dan perubahan variabel pada runtime.

d. Analisis Data
   a. Data yang dikumpulkan mencakup observasi kualitas debugging, waktu yang diperlukan untuk menyelesaikan bug, dan efektivitas setiap teknik.
   b. Analisis dilakukan dengan membandingkan strategi debugging berdasarkan kriteria seperti kecepatan, kemudahan implementasi, dan kemampuan mendeteksi masalah yang kompleks.

## 4. HASIL DAN PEMBAHASAN

### Hasil

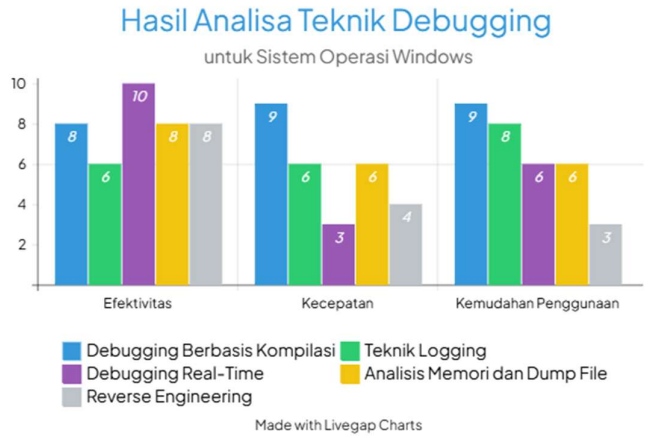
Dari pengujian berbagai teknik debugging pada aplikasi simulasi dan kasus nyata, hasilnya dirangkum dalam Tabel 1 di bawah ini.

**Tabel 1.** Evaluasi Teknik Debugging untuk Sistem Operasi Windows

| **Teknik Debugging** | **Efektivitas** | **Kecepatan** | **Kemudahan Penggunaan** | **Catatan** |
|---|---|---|---|---|
| Debugging Berbasis Kompilasi | Tinggi | Cepat | Mudah | Cocok untuk pengembangan awal aplikasi. |
| Teknik Logging | Sedang | Sedang | Mudah | Ideal untuk aplikasi yang sulit direproduksi di lingkungan pengembangan. |
| Debugging Real-Time | Sangat Tinggi | Lambat | Menengah | Memerlukan keahlian tinggi, efektif untuk masalah kompleks. |
| Analisis Memori dan Dump File | Tinggi | Sedang | Menengah | Berguna untuk aplikasi yang crash atau gagal merespons. |
| Reverse Engineering | Tinggi | Lambat | Sulit | Efektif untuk aplikasi pihak ketiga tanpa kode sumber. |
| Remote Debugging | Sedang | Cepat | Sulit | Membutuhkan konfigurasi tambahan untuk aplikasi jarak jauh. |

**Pembahasan**

Analisis Penggunaan Teknik Debugging




a. **Debugging Berbasis Kompilasi**: Teknik ini efektif untuk mendeteksi kesalahan pada tahap awal pengembangan. Informasi debugging yang ditambahkan saat

kompilasi memungkinkan pelacakan kesalahan yang lebih mudah, meskipun terbatas pada kesalahan pra-runtime.

b. **Teknik Logging**: Metode ini cocok untuk melacak alur program dan mendeteksi masalah runtime. Meskipun mudah diterapkan, desain log yang kurang optimal dapat membatasi manfaat teknik ini.
c. **Debugging Real-Time**: Teknik ini memberikan wawasan mendalam untuk mengidentifikasi bug dengan breakpoint dan analisis langkah-demi-langkah. Namun, membutuhkan keahlian tinggi dan waktu yang lebih lama, terutama pada aplikasi yang kompleks.
d. **Analisis Memori dan Dump File**: Teknik ini sangat berguna untuk menganalisis crash dan kebocoran memori, tetapi memerlukan keahlian teknis serta alat khusus.
e. **Static Code Analysis**: Teknik ini membantu meningkatkan kualitas kode dengan mendeteksi kesalahan sebelum runtime, meskipun tidak dapat menggantikan teknik lainnya untuk analisis runtime.
f. **Reverse Engineering**: Berguna untuk memahami aplikasi tanpa kode sumber, seperti malware atau aplikasi pihak ketiga. Namun, teknik ini memerlukan keahlian tinggi dan proses yang memakan waktu.
g. **Remote Debugging**: Memberikan fleksibilitas dalam menganalisis aplikasi pada sistem jarak jauh, tetapi memerlukan konfigurasi tambahan dan menghadapi kendala seperti akses jaringan terbatas.

**Permasalahan "Application Not Responding"**

Salah satu masalah utama yang sering dihadapi dalam sistem operasi Windows adalah kondisi di mana aplikasi tiba-tiba berhenti merespons atau "Application Not Responding." Masalah ini tidak hanya mengganggu pengalaman pengguna tetapi juga dapat menyebabkan hilangnya data atau kegagalan operasional pada aplikasi yang bersifat kritis. Permasalahan ini biasanya dipicu oleh:

a) **Konflik Antar Proses**: Ketika dua atau lebih proses berusaha mengakses sumber daya yang sama secara bersamaan tanpa mekanisme sinkronisasi yang memadai.
b) **Kebocoran Memori**: Aplikasi terus mengalokasikan memori tanpa melepaskannya kembali, menyebabkan kehabisan sumber daya.
c) **Kesalahan dalam Pengelolaan Sumber Daya Sistem**: Seperti file handle atau koneksi jaringan yang tidak ditutup dengan benar.

Penanganan masalah ini memerlukan pendekatan yang terstruktur. Teknik debugging seperti logging dan analisis memori menjadi alat penting untuk mendiagnosis masalah ini secara mendalam. Logging dapat membantu mengidentifikasi peristiwa-peristiwa yang menyebabkan aplikasi berhenti merespons dengan mencatat urutan eksekusi program secara rinci. Sebaliknya, analisis memori memungkinkan pengembang untuk memeriksa pola penggunaan memori dan mengidentifikasi area yang mungkin menjadi penyebab kebocoran atau konflik.

Untuk kasus "Application Not Responding," penerapan logging yang baik dapat memberikan informasi terkait titik mana dalam aplikasi yang menjadi penyebab bottleneck. Sementara itu, analisis dump file yang dihasilkan selama crash dapat memberikan wawasan tentang status memori dan kondisi sistem pada saat aplikasi gagal. Kombinasi kedua metode ini dapat membantu pengembang menemukan akar masalah dengan lebih cepat dan efisien.

## 5. KESIMPULAN DAN SARAN

Penelitian ini menyimpulkan bahwa strategi debugging yang tepat dapat secara signifikan meningkatkan efektivitas dalam mengatasi masalah perangkat lunak pada sistem operasi Windows. Teknik seperti logging dan analisis memori terbukti berguna dalam mengidentifikasi masalah runtime, sementara debugging berbasis kompilasi dan static code analysis membantu memastikan kualitas kode sebelum runtime. Meskipun setiap teknik memiliki kelebihan dan tantangannya, kombinasi metode sering kali diperlukan untuk menangani masalah yang kompleks, seperti kebocoran memori atau aplikasi yang tidak merespons. Studi ini memberikan panduan bagi pengembang untuk memilih dan menerapkan strategi debugging berdasarkan kebutuhan spesifik, membantu meningkatkan stabilitas dan performa aplikasi Windows.

## DAFTAR REFERENSI

Baecker, R., DiGiano, C., & Marcus, A. (1997). Software visualization for debugging. *Communications of the ACM*. Retrieved from https://dl.acm.org/doi/pdf/10.1145/248448.248458

Chiu, M. C., & Yang, L. S. (2024). Integrating explainable AI and depth cameras to achieve automation in grasping operations: A case study of shoe company. *Advanced Engineering Informatics*. Retrieved from https://www.sciencedirect.com/science/article/pii/S1474034624002313

Du Boulay, B., Cox, R., Lutz, R., & Romero, P. (2007). Debugging strategies and tactics in a multi-representation software environment. *International Journal of Human-Computer Studies*. Retrieved from http://users.sussex.ac.uk/~bend/papers/dyn.pdf

Gao, S., & Lin, Q. (2012). Debugging classification and anti-debugging strategies. *Proceedings of SPIE Machine Vision Conference*. Retrieved from http://www.andrew.cmu.edu/user/miaoy1/papers/icsct/shanggao.pdf

Grigoreanu, V., Burnett, M., & Wiedenbeck, S. (2012). End-user debugging strategies: A sensemaking perspective. *Proceedings of ACM Computer-Human Interaction*. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=98b4deeb1ab1e359cc11f681ca52448f95d44ff5

Hewardt, M., & Pravat, D. (2007). *Advanced Windows debugging*. Microsoft Press. Retrieved from https://books.google.com

Kapur, P. K., Pham, H., Singh, G., & Kumar, V. (2024). *Reliability engineering for industrial processes*. Springer. Retrieved from https://www.researchgate.net/publication/380008151

Li, C., Peng, F., Song, X., & Hu, W. (2016). An eye tracking research on debugging strategies towards different types of bugs. *Proceedings of IEEE Software Engineering*. Retrieved from https://ieeexplore.ieee.org/abstract/document/7552192/

Li, S., Yang, S., & Diao, W. (2024). Android's cat-and-mouse game: Understanding evasion techniques against dynamic analysis. *Proceedings of the IEEE Symposium on Software Reliability Engineering*. Retrieved from https://ieeexplore.ieee.org/abstract/document/10771369/

Metwalli, S. A. F. M. (2024). A suite for testing and debugging quantum programs. *Keio University Repository*. Retrieved from https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/download.php/KO90001001-20246277-0003.pdf?file_id=184892

Norby, A. (2024). A cross-architecture framework for anti-debugging techniques. *Dakota State University Theses Repository*. Retrieved from https://scholar.dsu.edu/cgi/viewcontent.cgi?article=1454&context=theses

Rodriguez-Padilla, C., Guerrero, M. M., & Segura, M. G. (2024). A study on teaching cyber-physical systems with a customized branded mobile robot. *Proceedings of the IEEE*

*Global Conference on Artificial Intelligence and Robotics*. Retrieved from https://ieeexplore.ieee.org/abstract/document/10578683/

Rybina, G. V., & Grigoryev, A. A. (2024). Educational design of dynamic intelligent system prototypes using tutoring integrated expert systems. *Proceedings of the 7th International Conference on Intelligent Systems Design and Applications*. Retrieved from https://ieeexplore.ieee.org/abstract/document/10551996/

Soulami, T. (2012). *Inside Windows debugging*. Microsoft Press. Retrieved from https://books.google.com

Spinellis, D. (2016). *Effective debugging: 66 specific ways to debug software and systems*. Addison-Wesley Professional. Retrieved from https://books.google.com

Zayour, I., & Hamdar, A. (2016). A qualitative study on debugging under an enterprise IDE. *Information and Software Technology*. Retrieved from https://www.sciencedirect.com/science/article/pii/S0950584915001767

Zeller, A. (2009). *Why programs fail: A guide to systematic debugging*. Morgan Kaufmann. Retrieved from https://books.google.com