

Bachelor's Thesis

Debug With Me: Shared Debugging Sessions in the Graphical Debugger JavaWiz

Dipl.-Ing. Dr. Markus Weninger, BSc

Institute for System Software

T +43-732-2468-4361

markus.weninger@jku.at

Student: Florian Gatzweiler

Advisor: Dipl.-Ing. Dr. Markus Weninger, BSc

Start date: February 2026

Background and Motivation

JavaWiz is a trace-based graphical debugger specifically designed for software development education. By providing intuitive, dynamic visualizations of program state—such as memory heaps, call stacks, and control flow charts—it helps novice programmers build accurate mental models of complex programming concepts. Because JavaWiz records a complete execution trace, it offers unique "time-travel" capabilities, allowing users to step backward and forward through a program's history.

Currently, JavaWiz is primarily used in a traditional, teacher-centric manner during lectures: the instructor runs the debugger on their local machine and shares the screen via a projector. While effective for demonstration, this approach leaves students as passive observers. They cannot interact with the visualizations on their own devices, nor can the lecturer easily gauge real-time student comprehension or allow students to actively participate in the debugging process.

To transform JavaWiz into a highly interactive, collaborative learning platform, we envision "Debug With Me": a feature that allows students to join a live debugging session via a shareable URL. By synchronizing the debugging state across multiple clients, educators can foster active learning, conduct real-time formative assessments, and allow students to collaboratively explore a program's execution.

Objective

The primary objective of this Master's thesis is to design and implement a collaborative, synchronized debugging environment within JavaWiz. The student will develop a robust backend and frontend architecture to support multi-user sessions accessible via a shareable URL. This includes implementing a granular permission system for execution control, flexible view synchronization, and an interactive annotation system. Furthermore, the underlying trace format must be extended to capture these new collaborative interactions, allowing sessions to be exported and reviewed offline.

Specific Tasks

To achieve these goals, the student will:

- **Session Management and Synchronization:** Implement a real-time synchronization mechanism that allows multiple users to join a shared JavaWiz session via a URL. Ensure that the underlying program state (the current execution step) is consistently synchronized across all connected clients.

- **Role and Permission System:** Develop a control mechanism for the lecturer to manage execution rights. The lecturer must be able to restrict stepping (forward/backward) to themselves, open it to all participants, or temporarily hand over control to a specific student (e.g., "*Can you step us to the exact location where the reference is set to null?*").
- **Flexible View Synchronization:** Implement a toggleable view synchronization feature. The lecturer should be able to enforce a "shared view" (all participants see the exact same visualizations, such as the Array View or Flowchart) or allow "independent views" (participants can open and arrange their own visualizations while the underlying program state remains synchronized with the lecturer's steps).
- **Interactive Markers and Annotations:** Create a system that allows both the lecturer and students to place markers (e.g., emojis, pins, question marks, speech bubbles) directly onto the visualizations. This should support various pedagogical use cases:
 - *Predictions:* e.g., "Place a pin on the flowchart path you think will be taken next."
 - *Questions:* e.g., Students placing a question mark on a specific heap object if its state is unclear.
 - *Explanations:* e.g., Adding speech bubbles to explain *why* a specific variable changed.
- **Trace Extension and Export:** Extend JavaWiz's trace-based architecture to record "live teaching" events. The trace must store metadata about when and where markers or annotations were placed during the session. Implement an export feature so these enriched traces can be downloaded and reviewed offline (e.g., allowing the lecturer to review student predictions post-class).

Non-Goals

- **Collaborative Code Editing:** This thesis does not aim to build a general-purpose collaborative IDE (like VS Code Live Share). The focus is strictly on shared *debugging* and *visualization* of already compiled code.
- **Modifying the JDI / AST Generation:** The core Java Debug Interface (JDI) data collection and Abstract Syntax Tree generation do not need to be fundamentally altered, only the frontend/backend synchronization and trace state metadata.

Modalities:

The progress of the project should be discussed at least every three to four weeks with the advisor(s). A time schedule and a milestone plan must be set up within the first 2 weeks and discussed with the advisor(s). It should be continuously refined and monitored to make sure that the thesis will be completed in time. The final version of the thesis must be submitted not later than 31.01.2027.