



Institut für Systemsoftware

(Aktuelle) Mitarbeiter des SSW



Leitung



Hanspeter Mössenböck

Sekretariat



Karin Gusenbauer



Liliane Loibl

Systemadministrator



Michael Kaffenda

Stammpersonal / Assistenten



Herbert Prähofer



Markus Weninger



Sebastian Kloibhofer



Lukas Makor



Christoph Pichler

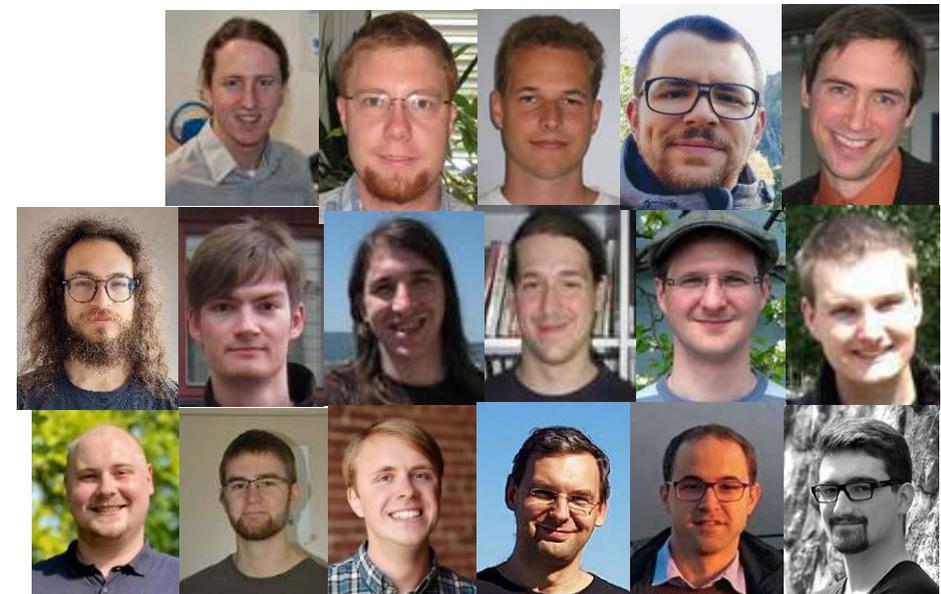


Florian Huemer

Studienassistenten im Bereich Compilerforschung mit Oracle

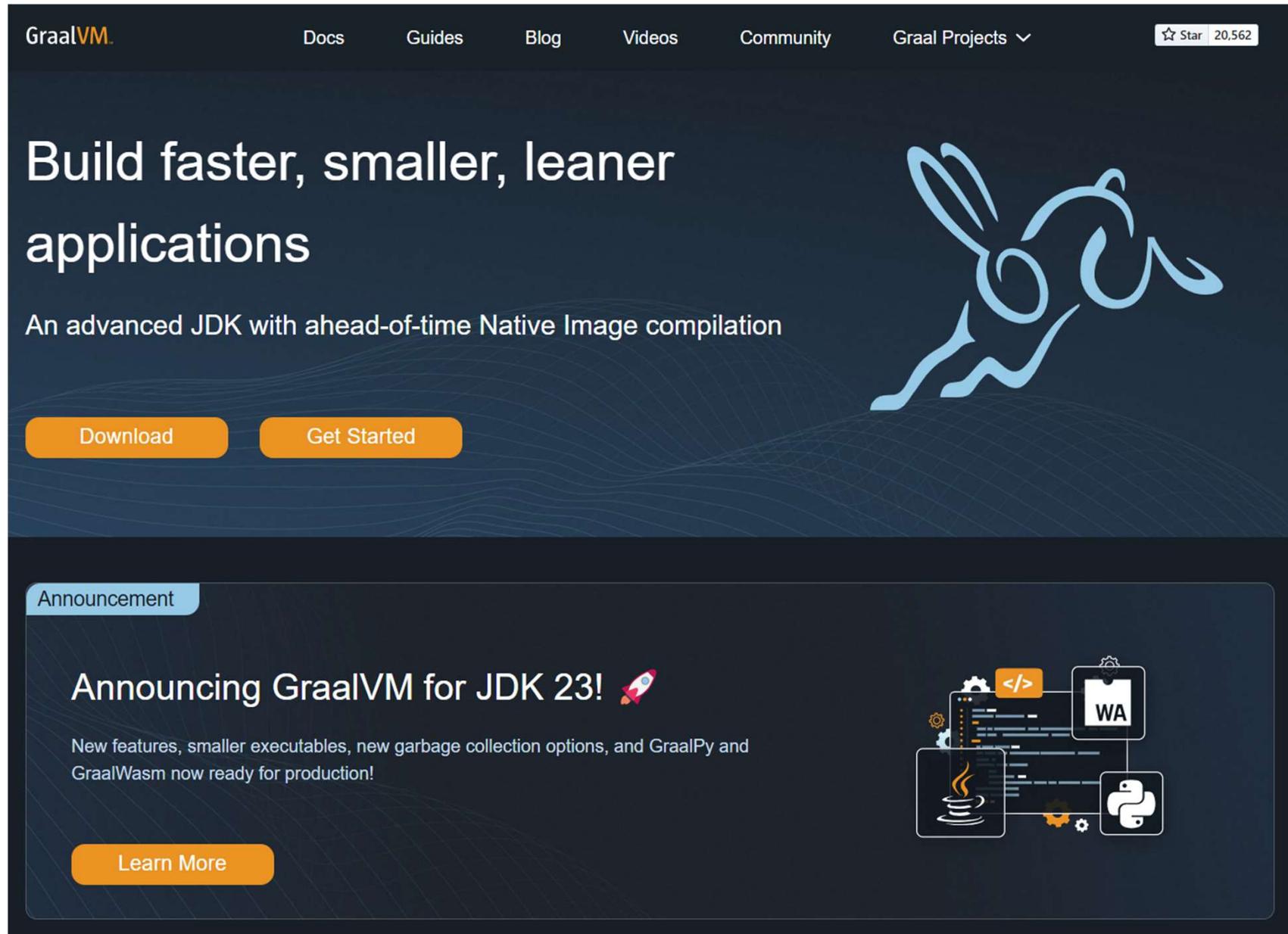


Oracle Labs



eInformatics / Grafischer Debugger „JavaWiz“





The screenshot shows the GraalVM website homepage. At the top, there is a navigation bar with links for Docs, Guides, Blog, Videos, Community, and Graal Projects. A star icon indicates 20,562 stars. The main heading reads "Build faster, smaller, leaner applications" with a sub-heading "An advanced JDK with ahead-of-time Native Image compilation". A blue rabbit logo is on the right. Below the heading are "Download" and "Get Started" buttons. A section titled "Announcement" features the text "Announcing GraalVM for JDK 23!" with a rocket icon, followed by "New features, smaller executables, new garbage collection options, and GraalPy and GraalWasm now ready for production!". A "Learn More" button is at the bottom left. On the right of the announcement is a diagram showing Java, Python, and WA (Wasm) components connected by lines and gears.

GraalVM. Docs Guides Blog Videos Community Graal Projects ▾ ☆ Star 20,562

Build faster, smaller, leaner applications

An advanced JDK with ahead-of-time Native Image compilation

Download Get Started

Announcement

Announcing GraalVM for JDK 23! 🚀

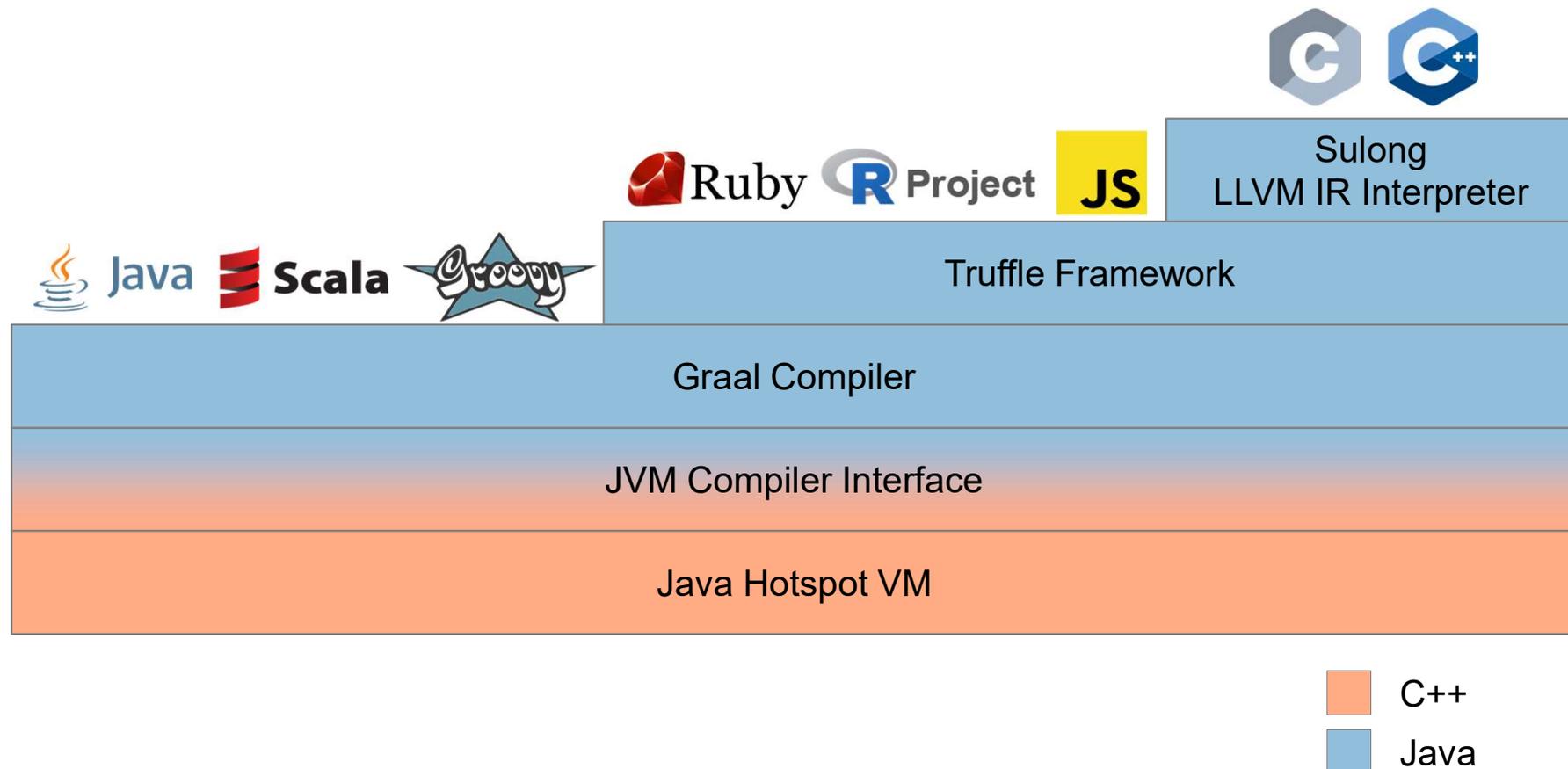
New features, smaller executables, new garbage collection options, and GraalPy and GraalWasm now ready for production!

Learn More

Forschungsbereiche



- **Graal:** dynamic compiler written in Java
- **Truffle:** language implementation framework



JavaWiz - Visualisierungstool für Programmieranfänger

The screenshot displays the JavaWiz IDE interface. On the left, the source code for `BirthdayFinder.java` is shown, with the `readPerson()` method selected. The right pane shows the 'JavaWiz Debugging Session' with a memory visualization tool. The tool is divided into 'Statics' and 'Heap' sections.

Statics:

- BirthdayFinder:** `int CAP` (10), `Person[] persons` (points to heap), `int nPersons` (0).
- Stack:** `String[] args` (empty), `String filename` ("persons.txt"), `Person p` (points to heap).

Heap:

- Person[]:** Array of 5 elements, all `null`.
- String[]:** `empty`.
- String:** `"persons.txt"`.
- Person:** `String firstName` (points to `"Reinhard"`), `String lastName` (`null`), `Date dayOfBirth` (`null`).
- String:** `"Reinhard"`.

The bottom of the IDE shows the 'Javawiz I/O Terminal' and a status bar with 'Open In Browser'.

Tracing Performance Metrics in Kotlin Multiplatform Projects via Compile-Time Code Instrumentation

Markus Weninger

markus.weninger@jku.at

Institute for System Software, Johannes Kepler University Linz, Austria

```
+irCall(writeStringFunc).apply {  
    extensionReceiver = irGetField(receiver: null, bufferedTraceFileSink)  
    putValueArgument(index: 0, irCall(toStringFunc).apply {  
        extensionReceiver = irGetField(receiver: null, stringBuilder)  
    })  
}
```

Offene Theme: <http://sww.jku.at/Teaching/Projects/open.html>

- Leider nicht immer aktuell
- Am besten **direkt Mail an einen der SSW-Mitarbeiter oder Prof. Mössenböck**
- Themenfindung dann typischerweise im persönlichen Gespräch
 - Mit Fokus auf die Interessen des Studenten / der Studentin

Open Projects

For Master's theses, Bachelor's theses or for Software Engineering projects in the Master's program

(Most topics can be adapted in scale to fit any of the above categories)

- **Tools for Visual Teaching and Visual Learning** (e.g., HTML + CSS + Typescript)
Interactive visualizations to help lecturers to teach content in a more engaging way and to help students to learn easier.
Contact: Dr. Markus Weninger
- **CI Upgrades for LaTeX Builds in Software Development 1 (UE)** (e.g., Python, TypeScript, LaTeX)
The lab (UE) for Software Development 1 is organised via a git repository. When preparing an exercise, we use pull requests to gather feedback from colleagues and discuss the subject matter. While we use a customized CI pipeline that automatically builds the lecture documents from TeX files, projects in this area focus on improving this process by integrating further checks and features, such as spell checking, linting, and formatting of TeX files and other auxiliary material (source code, figures), as well as other customizable checkers and feedback tools.
Contact: DI Sebastian Kloibhofer
- **New JavaScript Language Features - ECMAScript proposals** (Java, some JavaScript)
JavaScript is specified in the ECMAScript language specification. It is an evolving language, and is extended by a "proposal" process. Each new or improved feature is specified by one proposal. Current open proposals include Realms, Pipeline operator, In-Place Resizable and Growable ArrayBuffers, Array find-from-last, Array grouping, and several more. As the different proposals vastly differ in effort to implement them, we have topics for projects (project in software engineering), bachelor theses and master theses. The task is to fully implement the current state of the proposal in the GraalVM/Graal.js JavaScript engine.
Contact: Fabio Niephaus (Oracle Labs)
- **Windows Support for GraalPy** (Java, Python, C, Windows APIs)
Python on Windows provides some additional modules to access Windows-specific APIs including the MSVCRT, the registry, and the win32 API. GraalPy currently does not offer these. These modules could be implemented in pure Python using the built-in cffi or ctypes bindings, ported from CPython as a C extension library, or implemented in Java to call into the Windows APIs via Truffle NFI. The task is to weigh the implementation strategies and choose one to implement enough features to pass the standard library tests.
Contact: Fabio Niephaus (Oracle Labs)
- **Allocate Objects into Tail End of Humongous Regions** (C++)
Summary: Improve memory footprint of G1 humongous objects by removing inner fragmentation.
The G1 garbage collector is high-performance regional collector in the OpenJDK Hotspot VM: the Java heap is strictly split into same-sized regions. Objects larger than a single region ("humongous regions") are allocated using separate contiguous sets of regions, and are mostly unmovable for performance reasons. This poses a few problems, in particular at the end of such a humongous region there is often a significant amount of space that is effectively wasted and unavailable for allocation (inner fragmentation, explanation, enhancement request). The goal of this project would be to lessen the problem by implementing regular object allocation into the tail region of a humongous object.
Contact: DI Thomas Schatzl (Oracle Java)
- **Automatic Dynamic Optimization of Remembered Sets** (C++)
Summary: Let the G1 collector automatically determine remembered set container options for either reduced memory usage or improved performance.
The G1 garbage collector is the current default garbage collector in the OpenJDK Hotspot VM. It uses remembered sets to store locations of incoming references to a particular region of the heap. This data structure is basically an implementation of a sparse set of integers: the entire range of possible values is split into evenly sized areas. A top level concurrent hash table stores values in areas that are "in" the set in a so-called remembered set container. Such a container is represented, depending on the number of values to be stored in that area it covers, by different kinds of data structures, e.g. arrays, bitmaps, or even single special integers.
The remembered set implementation switches between containers on the fly depending on current remembered set entry occupancy of an area.
G1 currently sizes these containers statically, i.e. independent of actual distribution of values in a given remembered set. So a particular container has a fixed size being able to hold a fixed amount of values, eg. an "array" remembered set container always has 128 entries, regardless of what the typical occupancy of such an array container is. This wastes memory, because different types of applications (and remembered sets for different areas of the heap) exhibit different occupancy characteristics.
The task is to change G1 to let it reconfigure the remembered set containers based on statistics that need to be gathered while an application is running to optimize for the particular goal, and evaluate the effectiveness of these optimizations on several benchmarks.
Contact: DI Thomas Schatzl (Oracle Java)
- **Fixed Memory Usage Marking for G1** (C++)
Summary: Implement a marking algorithm that uses (small) constant memory and compare with the existing.
The G1 garbage collector is the current default garbage collector in the OpenJDK Hotspot VM. Its algorithm to determine reachable objects is a straightforward implementation of the Tri-Color abstraction. The drawback of this algorithm is that mark stack, a helper data structure, memory requirements is only bounded by the number of live objects which can be a very large number (in the MBs).
There is an algorithm that bounds only needs a very small mark stack and a small helper table to complete marking.
The task for this work comprises:
 - Implement the mentioned algorithm in the G1 garbage collector
 - The description only describes single-threaded operation, extend it to use multiple threads.
 - Compare its performance and memory consumption to the existing algorithm on benchmarks.Contact: DI Thomas Schatzl (Oracle Java)

Laufende und abgeschlossene Bachelorarbeiten: <http://ssw.jku.at/Teaching/Projects/>

Theses in Progress

- **Analyzing Memory Anti-Patterns over Time** (Roman Sperl, Supervisor: **Markus Weninger**)
- **A 64-bit Windows Performance Profiler for Performance Analysis and Optimization of the BMD software NTCS** (Florian Gatzweiler, Supervisor: **Markus Weninger**)
- **Interactive and Playful Visualizations of Graph Algorithms** (Marcel Burgstaller, Supervisor: **Markus Weninger**)
- **An Educational Mobile Game about Sorting Algorithms** (Michael Schöller, Supervisor: **Markus Weninger**)
- **Performance Impact of OpenTelemetry Observability on Enterprise SaaS Platforms** (Robert Pröll, Supervisor: **Markus Weninger**)
- **Utility Functions for Kotlin Compiler IR Generation** (Mario Birmili, Supervisor: **Markus Weninger**)
- **A Tool for Plagiarism Detection in Software Development Assignments** (Sandra Höllinger, Supervisor: **Lukas Makor**)
- **Development of a Networked Pixel Art Drawing Application** (Sebastian Wöber, Supervisor: **Lukas Makor**)
- **Extending Sulong's Debug Expression Evaluation for C++ Expressions** (Halil Bahar, Supervisor: **Christoph Pichler**)
- **Program Optimization Based on Concolic Execution** (Thomas Hackl, Supervisor: **Lukas Makor**)
- **Execution of Rust Programs on Sulong** (Arif Celik, Supervisor: **Christoph Pichler**)

Finished Theses

- 2024: **Framework für Hot-Swapping in Modul-basierten Anwendungen** (Bernhard Gili, Supervisor: **Sebastian Kloibhofer**) - gesperrt
- 2024: **An Educational Browser Game about Sorting Algorithms** (Jasmin Furlinger, Supervisor: **Markus Weninger**)
- 2024: **SMAVIZ: Interactive and Playful Visualizations of String-Matching Algorithms** (Kilian Wolfinger, Supervisor: **Markus Weninger**)
- 2024: **Abstract Syntax Trees for MicroJava** (Paul Lehner, Supervisor: **Christoph Pichler**)
- 2024: **A Tool for Creating Personalized Semester Schedules** (Alexander Burghuber, Supervisor: **Hanspeter Mössenböck**)
- 2024: **Feedback-directed fuzzing for the GraalVM compiler** (Florian Schwarcz, Supervisor: Dr.G.Barany, **Hanspeter Mössenböck**)
- 2024: **Synchronization of Hover Effects Across Multiple Visualizations in JavaWiz** (Sonja Cao, Supervisor: **Markus Weninger**)
- 2024: **Implementing a simple 2D physics engine** (Dominik Staudinger, Supervisor: **Lukas Makor**)
- 2024: **Ktor Reactive Server Applications** (Gregor Lang, Supervisor: **Herbert Prähofer**)
- 2024: **A Component for Visualizing Sequence Diagrams in the Visual Debugger Tool JavaWiz** (Melissa Sen, Supervisor: **Herbert Prähofer**)
- 2024: **LALR(1) Parser Table Generator** (Alexander Voglsperger, Supervisor: **Markus Weninger**)
- 2024: **Flowchart Visualization in JavaWiz** (Andreas Schlömicher, Supervisor: **Herbert Prähofer**)

Interesse?



- Schreibt uns eine E-Mail.
- Oder besucht uns direkt am Institut.
- **Kontakt:**
 - Institut für Systemsoftware
 - <http://sww.jku.at/>
 - Science Park 3, 2. Stock
 - Institutsvorstand: Prof. Hanspeter Mössenböck – hanspeter.moessenboeck@jku.at