Goto table of contents

THEMA STUDIENARBEIT

Exploring WebAssembly for versatile plugin systems through the example of a text editor

im Studiengang

TINF22IT1

an der *Duale Hochschule Baden-Württemberg Mannheim*

von

Name, Vorname:	Hartung, Florian
Abgabedatum:	15.04.2025

Bearbeitungszeitraum:	15.10.2024 - 15.04.2024
Matrikelnummer, Kurs:	6622800, TINF22IT1
Wiss. Betreuer*in der Dualen Hochschule:	Gerhards, Holger, Prof. Dr.

Erklärung zur Eigenleistung

Ich versichere hiermit, dass ich meine Projektarbeit mit dem

THEMA
Exploring WebAssembly for versatile plugin systems through the example of a text editor

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum	Unterschrift

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae.

Zusammenfassung

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet, ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum defuturum, quas natura non depravata desiderat. Et quem ad me accedis, saluto: 'chaere,' inquam, 'Tite!' lictores, turma omnis chorusque: 'chaere, Tite!' hinc hostis mi Albucius, hinc inimicus. Sed iure Mucius. Ego autem mirari satis non queo unde hoc sit tam insolens domesticarum rerum fastidium. Non est omnino hic docendi locus; sed ita prorsus existimo, neque eum Torquatum, qui hoc primus cognomen invenerit, aut torquem illum hosti detraxisse, ut aliquam ex eo est consecutus? – Laudem et caritatem, quae sunt vitae.

Contents

;

1 Introduction

1.1 Motivation & problem statement

- Plugin systems are great!
- WASM is great

TODO: write motivation last

- WASM's usecases are very limited -> it is still very new and evolving
- Plugin systems suffer from many issues: Security, interoperability, Portability, (Developer experience?)

1.2 Research question

- Can WebAssembly be *the* universal technology for building versatile plugin systems?

1.3 Method (structure of this work)

- How will this work be structured?
- What technologies, languages and terms are used?

2 Fundamentals

Als extra kapitel

2.1 WebAssembly (WASM)

- What is WebAssembly and how does it work?
- Originally designed as: fast, safe, low-level bytecode for the web
- History and today's usage of WebAssembly

2.1.1 WebAssembly System Interface (WASI)

2.1.2 WebAssembly Component Model

- language-agnostic interfaces
- WebAssembly Interface Type (WIT) specification
- bindings can be generated for a lot of languages from a WIT definition

2.2 Plugin systems

- What are plugin systems?
- Why are plugin systems important?
- Where are plugin systems used?

2.3 Rust

- Short explaination of why Rust is used for this project:
  - Safety & Performance
  - Tight coupling with WASM (wasmtime is written in Rust)

3 Requirement analysis for plugin systems

- Als Marktanalyse
- Analysis of related work and other projects/software
- Formal definition of requirements
- Projects
  - Text editors with WASM plugin systems: Zed
  - Common text editors with plugin systems: VSCode, IntelliJ, Eclipse
  - WASM plugin systems: Extism (cross-language plugin framework), maybe for audio processing?
  - Software with WASM plugin systems: Microsoft Flight Simulator, ZelliJ (terminal multiplexer)
- Works
  - Containerization with WASM in HPC: “Exploring the Use of WebAssembly in HPC” 10.1145/3572848.3577436
  - Structural and behavioural analysis of plugin components in Java: “Predictable Dynamic Plugin Systems” https://link.springer.com/chapter/10.1007/978-3-540-24721-0_9

3.1 Notes: Interesting projects

Zed Editor für Mac/Linux mit WASM Plugin System https://zed.dev/docs/extensions/developing-extensions https://github.com/zed-industries/zed/blob/94faf9dd56c494d369513e885fe1e08a95256bd3/crates/extension_api/wit/since_v0.2.0/http-client.wit

- WASM mit kompletter WIT Schnittstellendefinition (Generisches Typ/Funktions-basiertes System)pro Version
- Fokus auf Isolation von WASM Code
- Nur shim library für Rust vorhanden: zed_extension_api

Microsoft Flight Simulator

- Bietet unterschiedliche SDKs: WASM, JS, SimConnect SDK (?), SimVars (?)

ZelliJ Terminal multiplexer

- WASM mit WASI Unterstützung

3.1 Notes: Interesting projects

- API unabhängig von der Sprache mit Protobuf (Message-basiertes System, https://protobuf.dev/)
- Permission System gruppiert Events & Commands zusammen

Language Server Protocol

- wird oft als Ersatz für Plugin genutzt, um

4 WebAssembly for plugin systems

4.1 Overview

4.2 Choosing a plugin API

4.3 Safety

4.4 Performance

4.5 Summary

5 Implementing a WebAssembly plugin system for a text editor

5.1 Requirements

5.2 Design

5.3 Implementation

5.4 Example plugin development

5.5 Verification and validation

6 Results & Discussion

7 Outlook