HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ins institute for networked solutions

Symbol Mapper

Studienarbeit FS 2017

Abteilung Informatik
Hochschule für Technik Rapperswil

Autoren: Adrian Froehlich, Samuel Krieg
Betreuer: Prof. Stefan Richter
Projektpartner: The ABB Group
Datum: 2. Juni 2017

Inhaltsverzeichnis

1 Abstract

Ausgangslage Auch in der Entwicklung von eingebetteten Systemen finden die in der Softwareentwicklung gängigen Testpraktiken Verwendung. Um die Testumgebung von Mittelspannungs-Frequenzumrichter der Firma ABB zu vervollständigen, soll ein Werkzeug geschaffen werden, welches das manuelle Testen des eingebetteten Systems, durch Debugging von Symbolinformation, in einem Emulator unterstützt. Ziel der Arbeit war es einen Demonstrator zu erstellen, der die Machbakeit des Ansatzes aufzeigt. Der Grundbaustein soll in der Lage sein seine eigenen Symbolinformationen auszulesen. Um die Integration in die Testumgebung einfach zu gestalten, sollte es möglich sein, durch geringe Anpassungen am Grundbaustein, ebenfalls externe Quellen zu analysieren. Um die Integration in die Testumgebung einfach zu gestalten, sollte es möglich sein, durch geringe Anpassungen am Grundbaustein, ebenfalls externe Quellen zu analysieren.

Vorgehen/Technologien In einem ersten Schritt wird das Verständnis für die Dateiformate ELF und Dwarf erlangt. Mit dem gewonnen Wissen soll die Implementierung dieser Formate in den Compilern analysiert werden. Dies ermöglichte eine geeignete Bibliothek zu finden, welche beim Auslesen der Dateiformate hilft. Anschliessend wird die Applikation zum Auslesen von Symbolinformationen mit einer grafischen Benutzerschnittstelle realisiert.

Ergebnis Es wurde eine Applikation erstellt welche das Auslesen und Darstellen von Symbolinformationen von sich selbst ermöglicht. Die Symbolinformationen werden eingelesen, verarbeitet und in einer Baumstruktur hierarchisch geordnet mit Namen und Wert ausgegeben. In einem Graphen kann der zeitliche Verlauf der Symbolwerte verfolgt werden. Da die Anzahl an Debugging Information Entries bei grösseren Projekten sehr hoch ist, gelang es trotz Optimierung nicht, die erwünschte Verarbeitungszeit zu erreichen. Es wurde sichergestellt, dass durch geringe Anpassungen die Applikation ebenfalls Symbolinformationen von externen Quellen verarbeiten kann.

2 Management Summary

Ausgangslage Die Wichtigkeit, fehlerfreier Funktionsweise von Software steigt mit dem Schadenspotenzial durch ihrer Fehlfunktion. Ein Produkt der Firma ABB sind die Mittelspannungs-Frequenzumrichter für Pumpen, welche unter anderem in der Gasförderung eingesetzt werden. Um das Risiko zu minimieren und die korrekte Funktionsweise der Software auf den Frequenzumrichter zu gewährleisten werden wie im Softwareengineering üblich Tests durchgeführt. Einige der Tests sind nach heutigem Stand noch nicht genügend automatisiert und erfordern einen nicht praktikablen händischen Aufwand. Dies Trifft auch auf das Überwachen von Variablenwerten zu. Ziel der Arbeit war es die Machbarkeit der Variablenwertüberwachung und deren grafischen Darstellung aufzuzeigen und einen einfachen Demonstrator umzusetzen, der durch einen geringen Aufwand vonseiten der ABB in Ihr System integriert werden kann.

Vorgehen/Technologien Der Einstieg in die Arbeit setzte in einem ersten Schritt auf das Erlangen eines tieferen Verständnisses in die Funktionsweise von C++ Programmen voraus. Es stellte sich heraus, dass im Programm zusätzliche Informationen gespeichert sind mit welchen sich die gewünschte Funktionalität realisieren lassen. Mit dem gewonnen Wissen konnte nach möglichen verwendbaren Komponenten zur Erstellung des geforderten Programms gesucht werden. Die gefundenen Komponenten wurden einer Analyse unterzogen und nach Abwägung mit den besten Komponenten das Projekt realisiert.

Ergebnisse Es wurde eine Applikation erstellt, welche die gewünschten Funktionalitäten besitzt. Durch das Einlesen der zusätzlichen Informationen, welches ein Programm besitzt, konnten die Variablenwertüberwachung und deren grafische Darstellung umgesetzt werden. Es konnte eine Baumstruktur mit allen Variablen, derer Typen und Werten umgesetzt werden. Ebenfalls wurde ein Diagramm zur zeitlichen Überwachung der Werten umgesetzt. Bei grösseren Projekten stösst die Umsetzung jedoch an ihre Grenzen, weil die Rechenzeit durch die hohe Anzahl an Informationen stark ansteigt. Hier besteht die Möglichkeit die Leistung weiter zu verbessern, um auch grössere Projekte in nützlicher Frist zu analysieren. Wie gewünscht kann das Programm einfach umgeschrieben werden, damit es ein externes Programm anstelle von sich selbst analysiert.

3 Problemanalyse

3.1 Vision

3.1.1 Einleitung

Bei unserer Studienarbeit ging es um das Projekt mit dem Namen SymbolMapper. Dabei Handelt es sich um eine Applikation die für die ABB schweiz durch die HSR realisiert wird. Diese Anwendung soll dan Mitarbeitern der ABB bei ihrer täglichen Arbeit als Tool zur verfügung stehen.

3.1.2 Motivation

Das tool soll es Mitarbeitern ermöglichen, schnell und übersichtlich Werte/Variabeln von Spannungsumwandlern auszulesen zu können. Mit dieser erleichterten erfassung soll es so möglich sein die effizienz für diese Anwendung erheblich zu steigern.

3.1.3 Ziel

Das primäre Ziel dieser applikation ist es Mitarbeitern der ABB zu ermöglichen, schnell und einfach auf die internen Variabeln zuzugreifen. Dabei sollen die Werte aus einem DWARF fiel ausgelesen und ein einer Tree-view übersichtlich und asprechend dargestelt werden. Dies ermöglicht eine einfaches weiterverarbeiten der Inhalte.

3.1.4 Ausgangslage

Die ABB Schweiz produziert Spannungsumwandler. Für diese soll eine einfache möglichkeit geschaffen werden, die Werte der internen Variabeln auszulesen, die in einem Dokument vom Typ DWARF abgespeichert sind zuzugreifen. Im Moment besitzt die ABB keine möglichkeit, das ganze schnell und übersichtlich darzustellen respektive anzuzeigen.

3.2 Nicht funktionale Anforderungen

Es wurden von Seiten der ABB verschiedene nicht funktionale Anforderungen gestellt welche erreicht werden müssen. Einerseits sind dies Anforderungen bezüglich der Performance und Andererseits betreffen Sie die Umgebung und den Aufbau der Applikation.

3.2.1 NFA1: Startzeit

Die Zeit welche das Programm zum aufstarten benötigt darf eine Sekunde nicht überschreiten.

3.2.2 NFA2: Buildzeit des Symbol Trees

Die Zeit zum Builden des Programms darf 10 Sekunden nicht überschreiten.

3.2.3 NFA3: Betriebssystem

Die Applikation muss mindestens ab Windows 7 lauffähig sein.

3.2.4 NFA4: Wartbarkeit

Um die Wartbarkeit zu gewährleisten sollte die Applikation auf der Basis von langzeitstabilen Code aufbauen.

3.2.5 NFA5: Programmiersprache

Es ist gefordet die Applikation mit der Programmiersprache C++ zu schreiben. Der erforderte Standard ist C++11.

3.3 Funktionale Anforderungen

Es wurden von Seiten der ABB wurden zwei Hauptanforderungen an die Funktionalität des Programms gestellt, die Aufgabenstellung erweitert diese um weitere Punkte.

3.3.1 FA1: Berechnung von Adressen

Das Programm soll in der Lage sein die Adresse von Symbolen zu berechnen. Ebenfalls sollen die Adressen möglichst einfach abrufbar sein.

3.3.2 FA2: Symbol Tree

Das Programm soll in der Lage sein eine visuelle Repräsentation des gesamten Symbol Trees darzustellen. Ebenfalls soll es dem Benutzer möglich sein ein Symbol auszuwählen um dessen Wert auszulesen.

3.3.3 FA3: Aktualisierung der Werte

Der Benutzer kann eine Frequenz definieren mit derer der Wert eines Ausgewählten Symbols aktualisiert wird.

3.3.4 FA4: Darstellung des Werteverlaufs

Das Programm kann den zeitlicheverlauf Werteverlauf eines Symboles in einem Echtzeitdiagramm anzeigen.

3.4 Use Cases

Anhand der funktionalen Anforderungen und der Aufgabenstellung wurden folgende Use Cases erarbeitet.

3.4.1 Aktor

Der Nutzer der Applikation ist der einzige Aktor und wird im Folgenden User gennant.

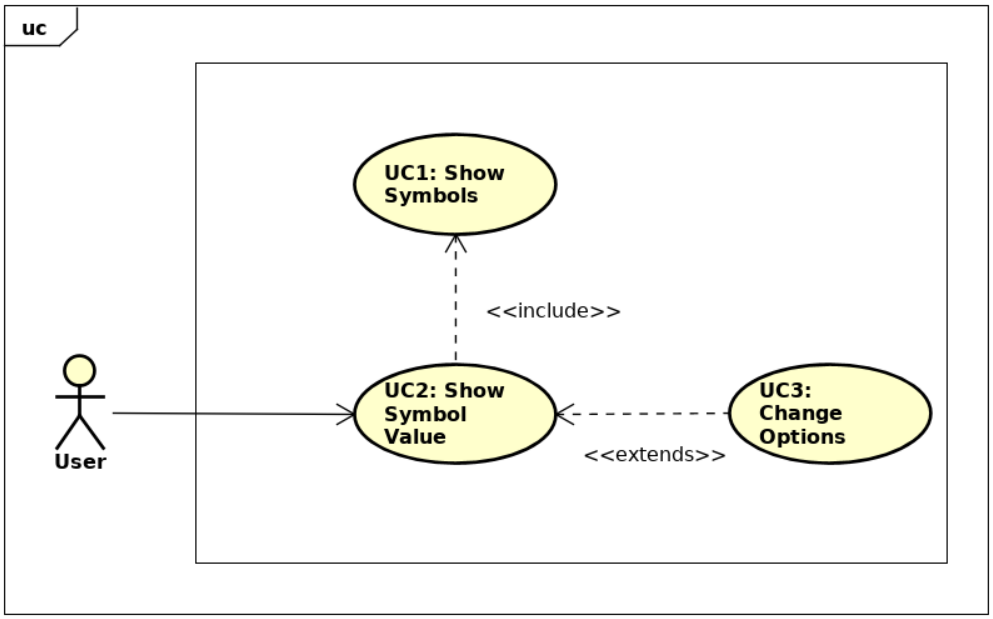

Abbildung 3.1: Use Case Diagramm

3.4.2 Bemerkung

Der User möchte gerne den Wert eines Symboles auslesen (UC3). Damit dies möglich ist, muss er zuerst ein geeignete Source ausgewählet werden (UC1). Symbol Mapper listet die gefundenen Symbole auf aus welchen der User Symbole auswählen kann, von welchen er den Wert haben möchte.

3.4.3 UC1: Show Symbols (Brief)

Der User hat die Möglichkeit verschiedenen Sources auszuwählen, von denen er die Symbole angezeigt haben möchte. Symbol Mapper liefert dazu einen Dateibrowser in dem eine Source ausgewählt werden kann. Der User bestätigt die Auswahl oder kann den Dialog jederzeit wieder verlassen. Symbole Mapper listet die vorhanden Symbole der ausgewählten Source auf.

3.4.4 UC2: Show Symbol Value (Brief)

Der User hat die Möglichkeit den Wert der aufgelisteten Symbole abzufragen. Durch die Auswahl eines gelisteten Symbols kann er den Wert ausgeben lassen.

3.4.5 UC3: Change Options (Brief)

Der User kann die Optionen der Werteausgabe anpassen. Dies ermöglicht dem User zum Beispiel die erhöhung der Aktualisierungsrate des Wertes. Es handelt sich hierbei um ein optionales Feature, welches nur bei ausreichender Zeit berücksichtigt wird.

3.4.6 UC1: Show Symbols (Fully Dressed)

Primary Actor: User

Precondition: Sourceauswahl wurde vom User angeklickt

Postcondition: Symbol werden angezeigt

Stakeholder and Interests:

- User: will Symbol einer Source angezeigt bekommen.

Goal: User sieht Symbole der ausgwählten Source

Trigger: Kilckevent auf Schaltfläche zur Auswahl der Source

Main Success Scenario (or Basic Flow):

1. Symbol Mapper liefert ein Dateibrowser
2. User wählt gewünschte Source aus
3. Symbol Mapper markiert gewünschte Source
4. User bestätigt Auswahl
5. Symbol Mapper öffnet aktive Source (falls welche ausgewählt) und liefert Übersicht über Symbole

Extensions (or Alternative Flows):

*a. jederzeit - Applikation stürzt ab
   1. User startet Applikation neu

1-3a. User bricht Vorgang ab
   1. Symbol Mapper schliesst Dateibrowser

5a. User wählt ungültige Datei oder Source-File
   1. Symbol Mapper macht User auf ungültige Auswahl aufmerksam

5b. Symbol Mapper produziert fehler beim Lesen der Source
   1. Symbol Mapper macht User auf fehler aufmerksam.

3.4.7 UC2: Show Symbol Value (Fully Dressed)

Primary Actor: User

Precondition: Symbol Mapper gestartet

Stakeholder and Interests:

- User: will Symbolwert auslesen

Goal: Analyse von Symbolwert einer laufenden Applikation

Trigger: User möchte das Programm verwenden

Main Success Scenario (or Basic Flow):

1. Symbol Mapper zeigt Oberfläche an
2. User wählt Source aus.

   Siehe Sub-UC "UC1: Show Symbols"
3. User markiert gewünschtes Symbol und klickt auf die Schaltfläche für das Anzeigen des Wertes.
4. Symbol Mapper bestätigt Auswahl und zeigt Wert des Symboles an.
5. Schritt 3-4 wird solange wiederholt bis alle gewünschten Symbole den Wert anzeigen.

Extensions (or Alternative Flows):

*a. jederzeit - Applikation stürzt ab
   1. User startet Applikation neu

1-3a. User bricht Vorgang ab
   1. Symbol Mapper schliesst Dateibrowser

4a. Symbol Mapper generiert Fehler beim Auslesen
   1. Symbol Mapper macht User auf den Fehler aufmerksam

4b. User stoppt die Werteanzeige eines Symbol.
   1. Symbol Mapper zeigt Wert nicht mehr an.

3.4.8 UC3: Change Options

Da es sich bei UC3 um ein optionales Feature handelt, wurde auf eine Fully Dressed Beschreibung verzichtet.

3.5 Literaturstudium

3.5.1 ELF Datei

Eine ELF Datei was kurz für Executable and Linkable Format ist eine ein File Format für ausführbare Dateien. Der Vorteil dieses File Formats liegt unter anderem darin das es cross-Platform unterstützt wird und so auf einer breiten Basis angewendet werden kann. Jedes ELF FIle besteht dabei aus folgenden Komponenten.

- Einem Elf Header dieser gibt unter anderem an ob im ELF File um 32 oder 64 Bit Adresse handelt.
- Programm Header Table, welche die Speicher Bereiche beschreiben. Diese Speicher Bereiche enthalten Informationen welche vom Programm zur Laufzeit benötigt werden. Sie zeigt also dem Kernel wie man zum Beispiel das Programm startet.
- Section Header Table, welche die sections beschreiben. Das können Daten sein die wichtig sind für linking sowie die relocation(namentlich relocation table, symbol table).
- Daten werden von den anderen tables referenziert sie beinhaltet unteranderem Bereiche wie die allocation table or symbol table etc. Jedes Byte in einem Elf File kann maximal zu einer section gehören allerdings kann es zu Bytes ohne Zugehörigkeit kommen.

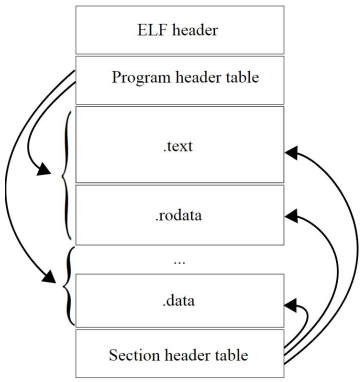

Abbildung 3.2: Struktur eines ELf Files

3.5.2 DWARF Übersicht

Bei der DWARF Information handelt es sich um ein Dateiformat das den Debugger beim Source Level Debugging unterstützt. Es ermöglicht dem Compiler also Static Variablen und die generelle Struktur des Programmes aufzuschlüsseln. Die DWARF Information Selbst befindet sich dabei innerhalb der ELF Datei in einem separaten Abschnitt. Die DWARF Datei ist in unterschiedliche Compilation Units gegliedert, was mit verschiedenen Dateien/Files gleichgesetzt werden kann. In den besagten Compilation Units sind die Informationen zu Funktionen und Variablen in einer Baum Struktur angelegt. Die einzelnen Elemente dieses Trees nennt man DIE(Debugging Information Entry).

3.5.3 DWARF Compilation Unite

Programme Können aus mehr als einem File bestehen. Diese Files werden unabhängig voneinander Kompiliert und anschliessend zusammengefügt. Jedes dieser individuel voneinander kompilierten Programme ist eine Compilation unit. Die DWARF Information für jedes dieser Compilation Units beginnt mit einer Speziellen DIE. Diese DIE enthält generelle Information die für das kompilieren benötigt werden. Das beinhaltet Informationen wie Directory, Name des source files, Programmiersprache die genutzt wurde, sowie offsets der DWARF data_sections. Zusätzlich zu diesen Informationen ist im Fall das die Compilation Unit zusammenhängend ist(zum Beispiel einem Stück im Memory), noch ein weiteres Attribut vorhanden das die Adresse der Compilation unit beinhaltet. Dieser Umstand erleichtert es zum Beispiel dem Debugger zu bestimmen, welche compilation unit für Code an einer bestimmten Memory Adresse verantwortlich ist. Falls die Compilation Unit aber nicht zusammenhängend ist dann wird eine Liste von Memory Adressen durch den Linker respektive den compiler bereit gestellt. Als Zusatz kann noch erwähnt werden das Diese spezielle DIE der oberste Node im besagten DWARF tree ist.

3.5.4 DWARF DIE Übersicht

DIE(Debuging Information Entry) sind die einzelnen Nodes im Tree der die Dwarf Information ausmacht. Die DIE selbst besitzt dabei einen sogenannten Tag, dieser beschreibt was die DIE für eine Funktion hat(Bsp. Eine Variabel hat den Tag: DW_TAG_Variable). Hinzu kommt noch eine Liste von Attributen die jede DIE besitzt, und die sie weiter beschreibt. Diese Attribute können alles Mögliche sein und gehen vom Namen der DIE bis hin zur Startadresse einer Funktion. Diesen Attributen

ist dabei auch ein Name zugewiesen, der mit DW_AT beginnt. Hinzu kommt das DIEs in 2 Kategorien eingegliedert werden können. Die erste Kategorie von DIEs beschreibt Daten und die Zweite Funktionen/ausführbarer code.

```
<34> DW_TAG_namespace
       DW_AT_name __cxx11
       DW_AT_decl_file 0x10
       DW_AT_decl_line 0xdf
       DW_AT_sibling <0x664c>
```

Abbildung 3.3: Beispiel für den Aufbau und Verschachtelung von Debugging Information Entries [1]

3.5.5 DIEs für Daten

Basistypen: DIEs für Basistypen werden mit DW_TAG_baste_type gekennzeichnet. Diese Typen werden dann mit den Basis typen auf der Zielmaschine gelinked. Somit funktioniert die Definition für diese Basistypen im DWARF File für verschiedene Sprachen/Architekturen was das DWARF File univerell êinsetzbar macht. Variabeln: Variablen sind im Prinzip ziemlich simpel, so sind sie eigentlich nichts anderes als reservierter Speicher mit einem Namen. Welche Werte dann in einer Variable abgespeichert werden können und ob sie modifiziert werden kann usw. wird durch den Typ geregelt. Was die Variable primär ausmacht ist der Scope, sowie die physikalische Adresse der Variabel. Das erlaubt es, das auch mehrere Variablen mit dem gleichen Namen in einem Programm vorkommen können. Dabei unterscheidet das DWARF File Format zwischen 3 verschiedenen Variablen typen: Konstanten, formale Parameter und Variablen. Die meisten dieser Variablen besitzen in den DIEs dabei ein Location Attribut das zeigt wo es sich befindet allerdings haben Variablen nicht immer ein fixe Adresse. So kann es sein das sie dynamische alloziert werden. Dann muss der Standort der variable berechnet werden, was mittels dem Offset zu einer bekannten Adresse einer anderen DIE(Parent) geschieht.

3.5.6 DIEs für Funktionen/asuführbarer Code

Funktionen und Subprogramme werden von DWARF als unterschiedliche Sache des gleichen Typs behandelt. So besitzen beide denselben Tag(DW_TAG_subprogram). Allerdings unterscheiden sich die beiden typen im Bezug auf ihre Attribute. So haben Subprogramme Attribute, die den unteren und oberen Adressbereich im Memory anzeigen wenn das Subprogramm zusammenhängend ist oder eine Liste von Adressen wenn es nicht zusammenhängt. Dabei ist noch zu erwähnen das falls nicht anders spezifiziert die untere Adresse als einstiegspunkt in den Code verwendet wird.

3.5.7 Speicherung/Entschlüsselung der DWARF Daten

Im Prinzip sind die DWARF Daten in einem Tree angeordnet, also jede DIE kann Children und Siblings haben. Allerdings ist aus Platz technischen Gründen, eine Kompression der Daten nötig. Dabei unterstützt das DWARF Format unterschiedlichste Möglichkeiten, dies zu erreichen. Eine Möglichkeit wäre es den Tree nur logisch mittels Präfix zu speichern. Das heisst, es wird gespeichert ob eine DIE über Kinder verfügt und falls ja, dass das die nächste DIE ein Kind davon ist. Wenn eine DIE hingegen keine

Kinder hat, dann ist die nächste DIE ein Geschwister. Über diesen weg können links zu Geschwistern und Kindern entfernt werden. Eine weitere Möglichkeit zur Komprimierung der DWARF Daten, ist das abkürzen der Attribute. Hier wird, da viele Compiler eine limitiert Anzahl von Attributen erzeugen, eine Tabelle für eben jene angelegt und zu diesen verlinkt. So wird die zu speichernde Information zwar stark reduziert allerdings auf kosten einer erheblich erhöhten Komplexität.

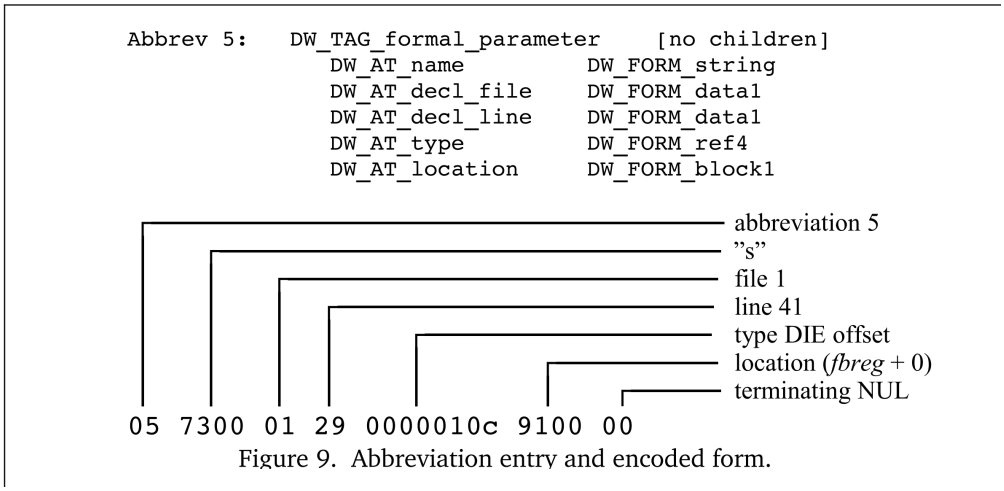

Figure 9. Abbreviation entry and encoded form.

Abbildung 3.4: Beispiel für kompremierte DWARF Daten [1]

3.5.8 Name Manling

Der C++ Compiler muss in der Lage sein muss mehrere Symbole mit dem selben Namen für Funktionen und Daten zu generieren. Zum Beispiel für zwei Funktionen mit dem selben Namen aber unterschiedlichen Argumenten. Um dieses Problem zu lösen verwendet der Compiler Name Mangling. Bei dieser Technik wird der Symbolname kodiert mit zusätzlichen informatienen wie den Funktionsargumenten und ermöglicht so die Verwendung von gleichen Namen. Problematisch ist, dass bei C++ zwei verschiedene Name Mangling Formate gibt. Eines findet Verwendung beim GCC das Andere beim MSVC. [2]
Beispiele von Name Mangling:

GCC _ZNK8KxVectorIP13KxLogObserverjE10idxIsValidEj

MSVC ??_R1A@?0A@EA@?$KxSet@VKxSymbol32@@I@@8

3.6 Libraries

3.6.1 QCostum Plot

Bei QCostum Plot handelt es sich um eine Library für das Plotten und Visualisieren von Daten/Datensätzen. Durch ein breites Spektrum, unter anderem auch im Bereich der Echtzeit Analyse von Daten, viel diese Library schnell positiv auf. Ein weiterer wichtiger Punkt dieser Library war, das es sich hier um ein OpenSource Produkt handelt, das unter die «default license GPL, feel free to use QCP» fällt, und somit keine Gebühren für die Nutzung anfallen. Neben einem ansprechenden Design, war der letzte entscheidende Faktor der auch noch überprüft wurde, die äusserst zufriedenstellende Performance.

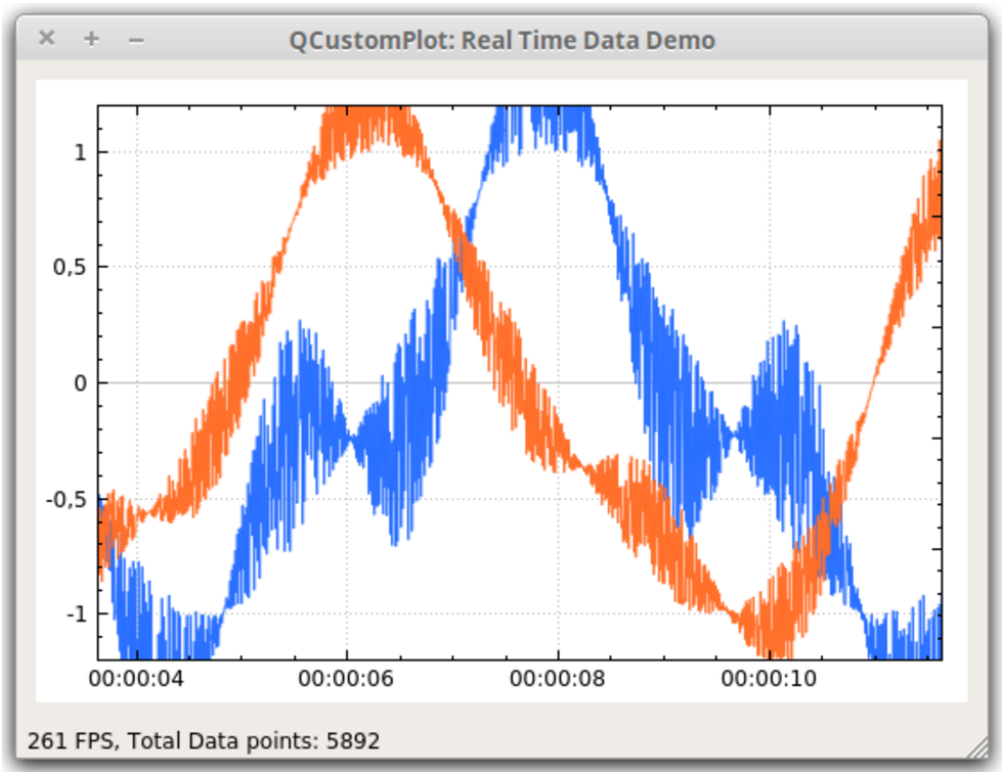

Abbildung 3.5: Beispiel für einen Graphen[3]

3.6.2 Libelfin

Liblfin ist eine opensource Library die allerdings noch nicht fertiggestellt ist. Trotzdem verfügt sie aber bereits über einen enormen Funktionsumfang wen es um das Auslesen und Aufbereiten von Informationen/Daten aus Elf Files und DWARF Informationen geht. Features dieser library sind dabei:

- Bei Liblefin handelt es sich um Native C++11 code, der unteranderem designed ist um gut mit C++11 features zu interagieren.
- Liblefin verfügt bereits über eine komplette Implementation für das Parsen von DIEs(Debugging information entries).
- Unterstützt alle DWARFv4 DIE value types ausser Locationlists und Macros.
- Fast kompleter Evaluator für DWARFv4 expression and location Beschreibungen.
- Kompletter Interpreter für Line tables.
- Ein Iterator der einfach compile_units und DIE trees traversieren kann.

Da diese Library neben der Ausgabe des Basic DIE trees auch wie in diesem Projekt benötigt die SymbolTable des Elf files ohne probleme auslesen kann, stellte sie sich schnell als primärer Kandidat für die Umsetzung des Projekts heraus.

3.6.3 GCC

Als eine weitere Möglichkeit zum Auslesen war die Nutzung von Elementen des GCC Compilers gedacht. Zunächst machte dieser Ansatz ein vielversprechender eindruck, stellte sich allerdings relativ schnell als problematisch heraus. Ein erstes Problem war zunächst einmal die benötigten Funktionen

zu ermitteln, da der gcc Compiler aus unzähligen Klassen besteht die teils über 50 000 Zeilen Code ausmachen. Ein weiteres Problem war die Vernetzung der einzelnen Funktionen, so war es uns nicht möglich die funktionalität einzelner Klassen schnell auszubauen, da die Kllassen untereinander stark vernetzt waren. Somit stellte sich dieser Ansatz sehr schnell für in diesem Pojekt als zu aufwändig heraus.

4 Lösungsentwurf

4.1 Konzept

Zur Umsetzung der Anforderungen muss die Applikation eine externe Datei(Executable oder ELF-File) oder sich selbst prozessieren können. Das Ziel ist eine grafische Repräsentation des Symbol Tree mit den entsprechenden Werten der Symbole zu erreichen.
Durch Auslesen und Parsen der ELF/DWARF-Informationen werden einersetzt die Symbol Table sowie die DIEs zugänglich gemacht. Dies ermöglicht das Abbilden der einzelnen DIEs in Nodes. Ausgehend von den gefunden Symbolen in der Symbole Table können die Nodes hierarchisch zu einem Baum verkettet werden. Ebenfalls werden Symboladressen mittels dem Offset aus den DIEs berechnet. Dies wird während der Verkettung, von der Symbole Table ausgehenden Symboladressen, durchgeführt. Sind alle notwendigen Nodes verknüpft erfolgt die Darstellung in der grafischen Benutzeroberfläche. Durch eine Tree-View werden die zu einem Baum verknüpften Nodes dargestellt und traversierbar gemacht. In einem weiteren Bereich der Benutzeroberfläche wird der zeitliche Verlauf der Symbole/Node-Werte in einem Diagramm dem Benutzer sichtbar gemacht.

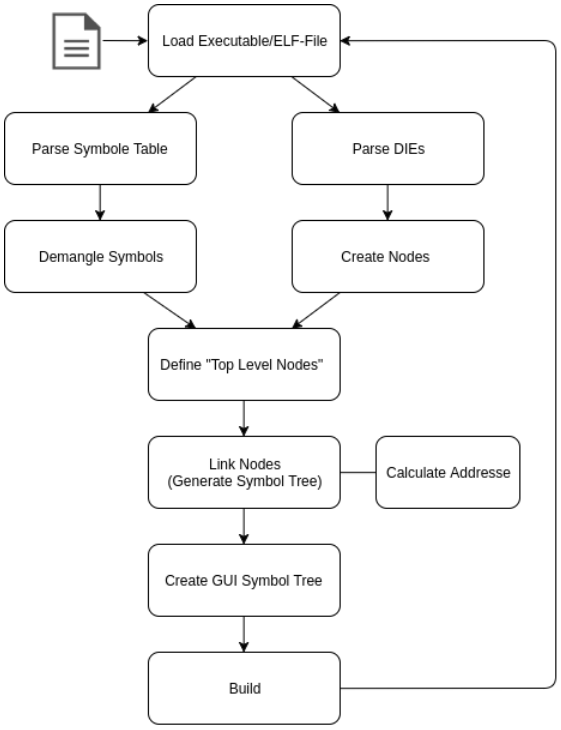

Abbildung 4.1: Grafische Darstellung des Programmablaufs

4.2 Library Entscheid

Für das evaluieren der schlussendlich zu verwendenden Komponenten kam eine grobe Bewertungsmatrix zum Einsatz. Diese schenkt folgenden punkten Beachtung:

- Performance: Respektive zeit die aufgewendet werden muss einen gewünschten Task durchzuführen.
- Lizenz: Dieser Punkt ist insofern wichtig da der finanzielle Aspekt auch eine rolle spielt.
- Basisfunktionen: Wie gut die Komponenten die notwendigen Tasks ausführen können.
- Dokumentation: Verfügbare Informationen zur Implementation und zum Gebrauch der Komponente.
- Zusatzfunktionen: Funktionen die noch nicht benützt werden aber möglicherweise in zukunft gebrauch finden könnten.
- Einsetzbarkeite: Wie gut lässt sich die gewünschte Funktion extrahieren respektive einsetzten.

4.2.1 Eignung der libraries

Name	Performance(5)	Lizenz(10)	Basisfunktionen(15))
Libelfin	3	10	14
elfio	4	10	4
gcc	3	9	13

Tabelle 4.1: Vergleich der Libraries Nr.1

Name	Dokumentation(5)	Zusatzfunktionen(5)	Einsetzbarkeit(10)	Total(50)
Libelfin	2	2	9	40
elfio	5	2	8	33
gcc	2	4	2	33

Tabelle 4.2: Vergleich der Libraries Nr.2

Wie in der Tabelle erkennbar, ist für dieses Projekt Libelfin die geeignetste Lösung. Dieser Umstand verdankt Libelfin primär der Tatsache, dass es sämtliche Funktionen, die von Seiten der Applikation benötigt werden, zur Verfügung stellt. So besteht die Möglichkeit neben dem DWARF Information auch die Elf Informationen wie Symbol Table etc. auszulesen.

4.3 Naming Mangling Entscheid

Es wurde entschieden, dass einzig das Name Mangling Format des GCC unerstützt wird.

4.4 Aufbau

Der Aufbau des Programms kann in Fünf Teile gruppiert werden.

4.4.1 Parsen von ELF/DWARF-Daten

Als ersten Schritt müssen die Daten, sei es das Programm selbst, ein anderes Programm, oder ein ELF-File, eingelesen werden. Unter der Benutzung der Library "libelfin" können die Daten geparset und aufbereitet werden. Somit stehen die Daten über die Library dem Programm zur Verfügung.

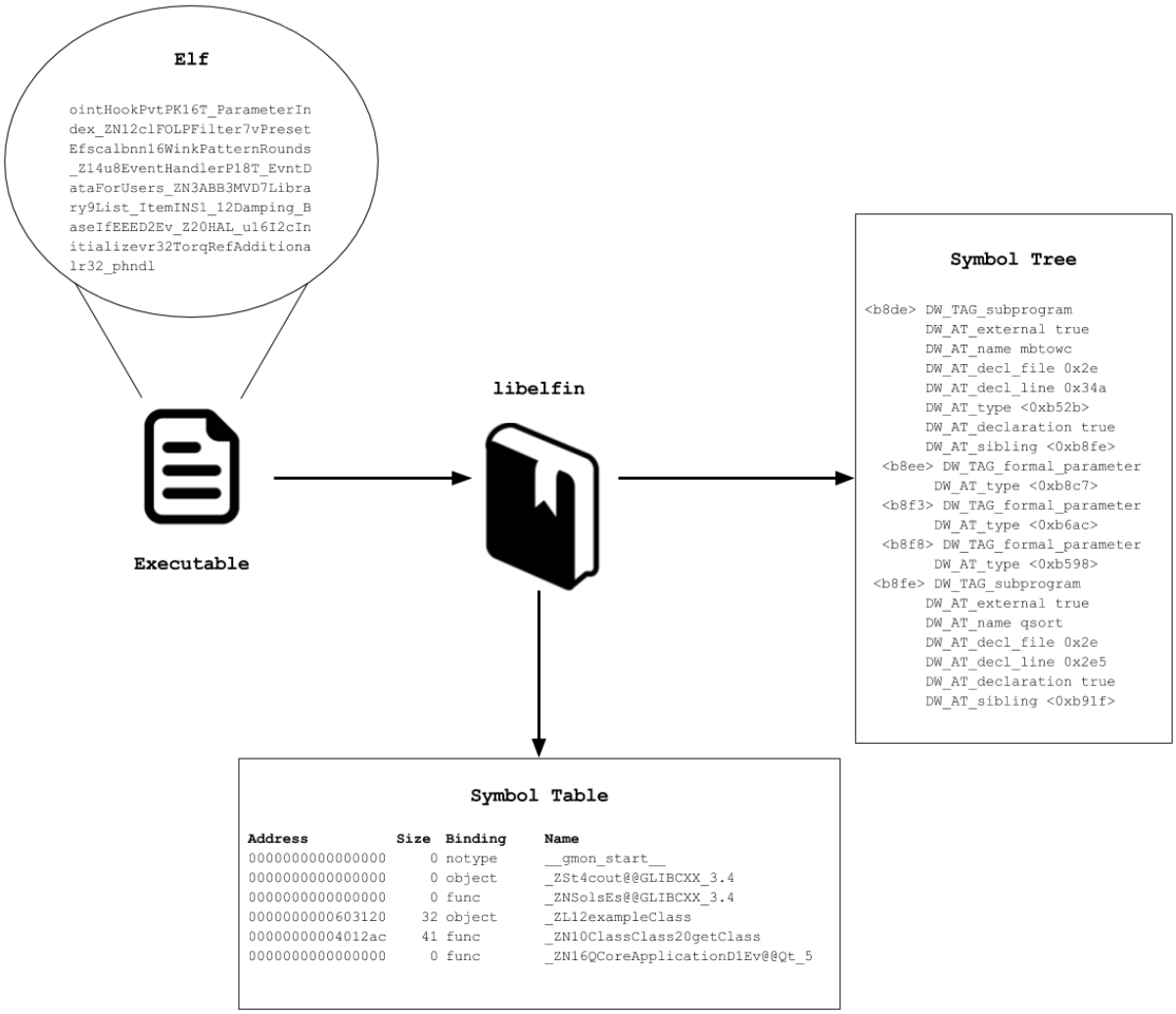

Abbildung 4.2: Grafische Darstellung des Parsens von dem ELF-Format mit Hilfe der Library "libelfin"

4.4.2 Erstellung von Nodes

Mittels der Library "libelfin" können die DIEs (Debugging Information Entries) ausgelesen werden. Das Ziel ist es, den DIEs ensprechende Nodes zu erstellen und in einer passenden Datenstruktur zu speichern. Dies wird erreicht in dem mittels den DIE-Tags die in DIEs enthaltenen Informationen in den Node geschrieben werden.
Wichtig Attribute sind:

Name Zur Identifikation des Symbols. Bsp: testVariable

Offset Zur Identifikation und Verlinkung der Nodes. Bsp: 0xb52b

Type Enthält eine Referenz auf den DIE welche den Typ definiert. Bsp: 0x181ed

Es müssen jedoch nicht alle Attribute vorhanden sein.

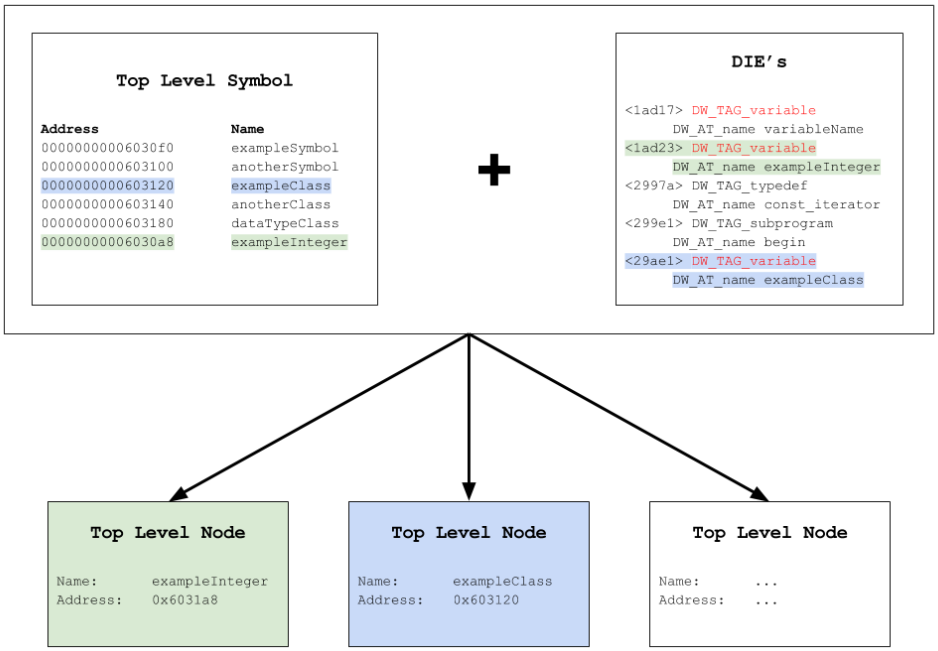

Abbildung 4.3: Grafische Darstellung der Herleitung von "Top-Level-Nodes"

4.4.3 Finden von Top-Level-Symbolen

Um den Startpunkt für die Verlinkung von den Nodes zu finden, muss durch Verwendung der Library "libelfin" die Symbol Table ausgelesen werden. Durch filtern nach dem Type "Object" und einem validen Symbolname können die Top-Level-Symbole gefunden werden. Das Demanling der in Frage kommenden Symbolnamen ermöglicht in der Datenstruktur der Nodes nach einem Node mit dem selben Namen zu suchen. Alle gefundenen Nodes bei welchen der Name übereinstimmt sind Top-Level-Nodes und werden in einer weiteren Datenstruktur abegelegt. Ebenfalls muss die Adresse des entsprechenden Symbols aus der Symbol Table ausgelesen und im Node platziert werden. Dies dient als Startwert der Adressberechnung.

4.4.4 Verlinkung von Nodes

Die Verknüpfung der Nodes funktioniert wie folgt: In jedem Node ist die Adresse der Typendefinition gespeichert. Mittels dieser Adresse wird der Node der Typendefinition herausgesucht. Im Anschluss werden all Children des Typendefinitions-Nodes und den Basis Node angefügt und für jedes dieser Children wird die Verlinkung nochmals rekursiv aufgerufen.

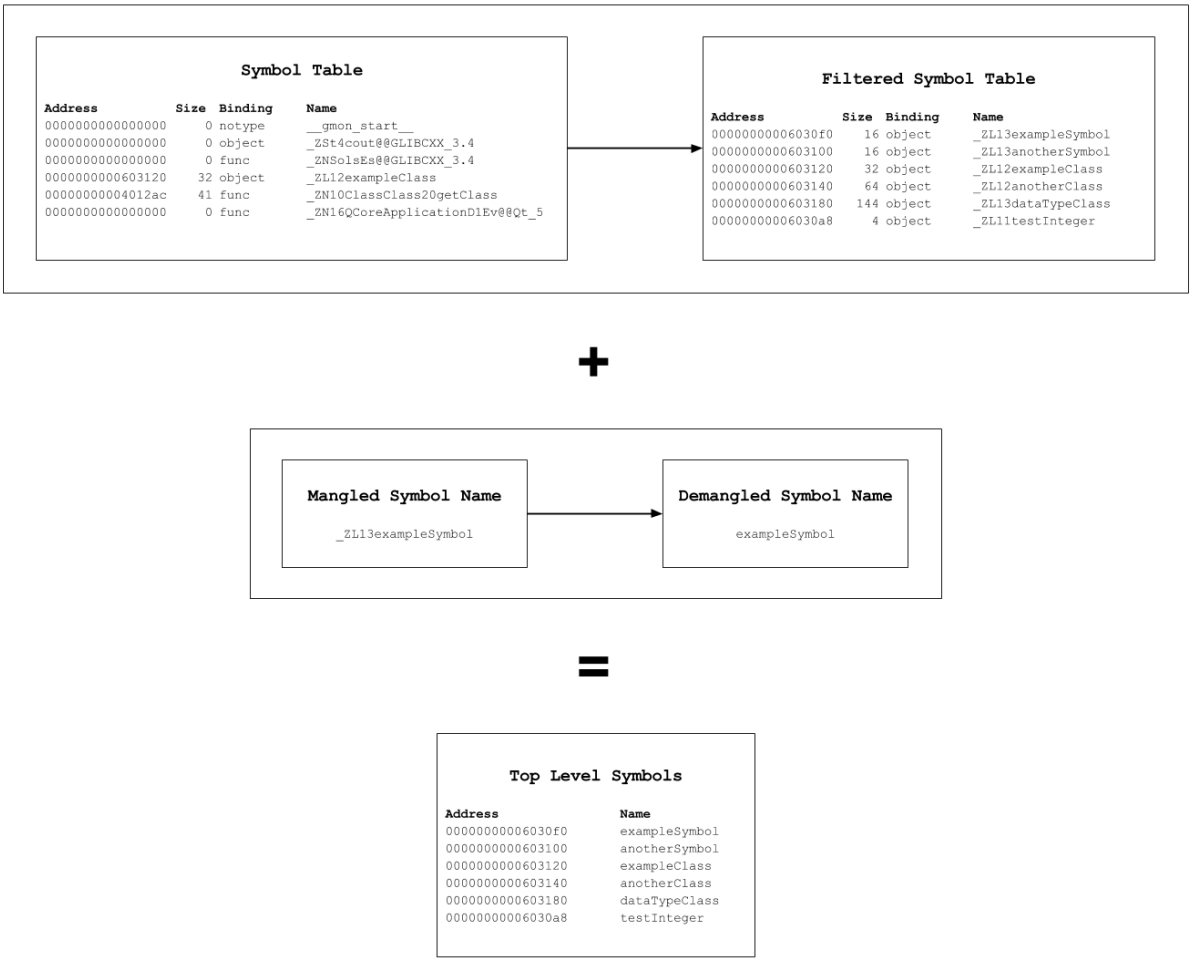

Abbildung 4.4: Grafische Darstellung der "Top Level Symbol"-Gewinnung aus der "Symbol Table"

4.4.5 Erstellung der grafischen Repräsentation

Für die Erstellung des Trees auf der GUI Oberfläche wird simpel über alle gewünschten Nodes iteriert und mittels einer rekursiven Funktion auf die Children wird der Tree ergänzt.

4.5 Externes Design

Die Applikation SymbolMapper erhält ein möglichst einfach gehaltene Benutzeroberfläche. Ziel ist, dass der Benutzer einen möglichst einfachen und schnellen Einstieg in die Benutzung von SymbolMapper hat. Die Benutzeroberflächenelemente sollen soweit als möglich selbst erklärend und logisch in ihrer Funktion für den Anwender sein. Es wurden von zwei Konzepten Mock-ups von eifachen Handskizzen erstellt um das Aussehen und die Funktionen der Benutzeroberfläche zu planen.

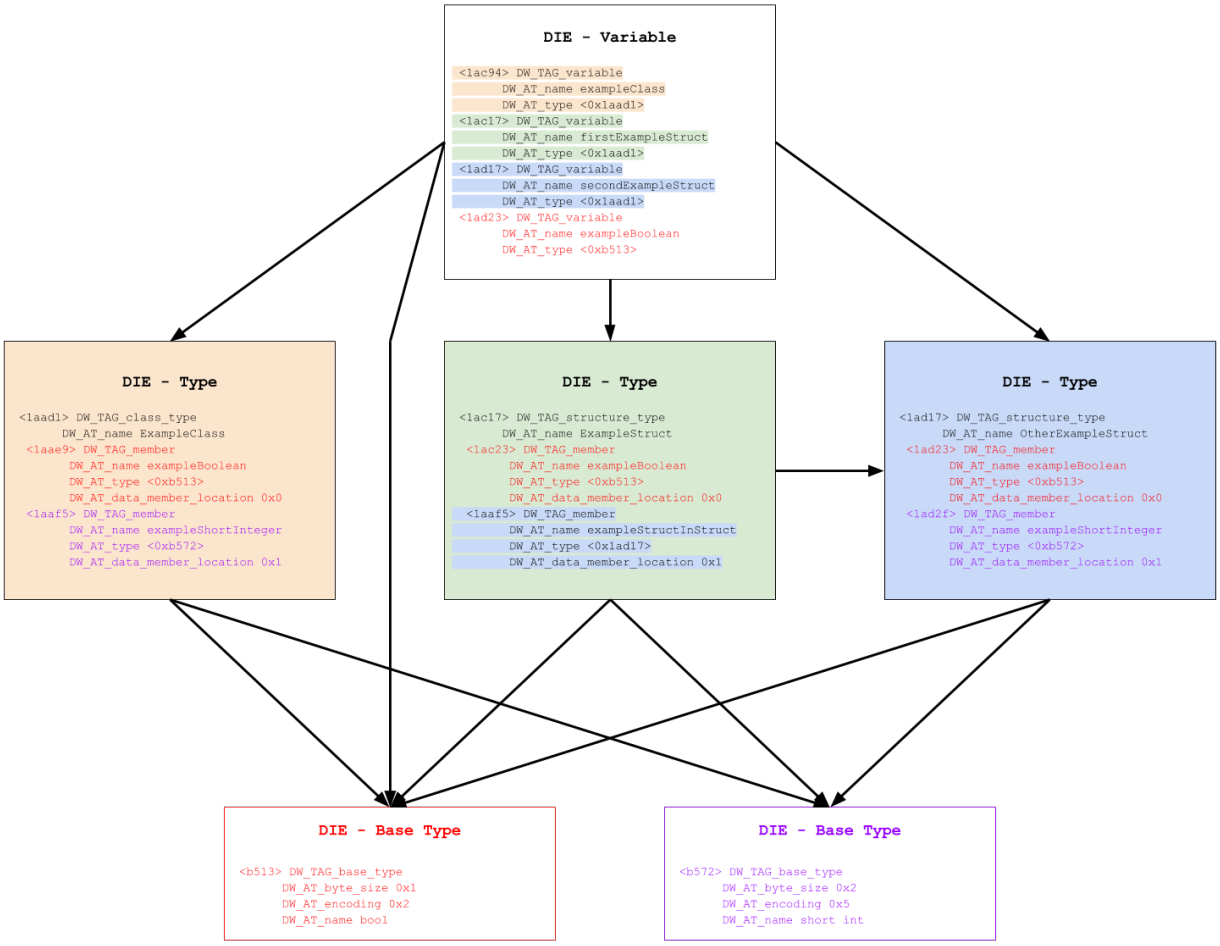

Abbildung 4.5: Darstellung einer Verlinkung von DIE's

4.5.1 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche wird der Einfachheit halber auf ein Fenster beschränkt. Dies soll der Übersicht dienen und die Benutzerfreundlichkeit fördern. Die Oberfläche ist in drei Teile aufgeteilt: Symbolbaum, Diagramme, und Optionen.

Symbolbaum

In diesem Bereich werden die Symbole in einer Baumstruktur mit den ausgelesenen Werten aufgelistet. Die Baumstruktur ist beim Start zugeklappt um einen Überblick zugeben. Durch aufklappen der Konten können die entsprechenden Kind Symbole angezeigt werden. Beim Elternknoten werden als Werte die Werte der direkt nachfolgenden Kindersymbole angezeigt.

Diagrammbereich

Dieser Bereich dient zur zeitlichen Darstellung des Werteverlaufs von Symbolen. Die Anzahl der Diagramme kann durch hinzufügen und löschen von Diagrammen variiert werden. Dieses Feature ist optinal und wird bei genüged Zeit umgesetzt.

Add Graph Fügt ein Diagramm zum Diagrammbereich hinzu.

Remove Graph Entfernt ein Diagramm vom Diagrammbereich.

Diagramm

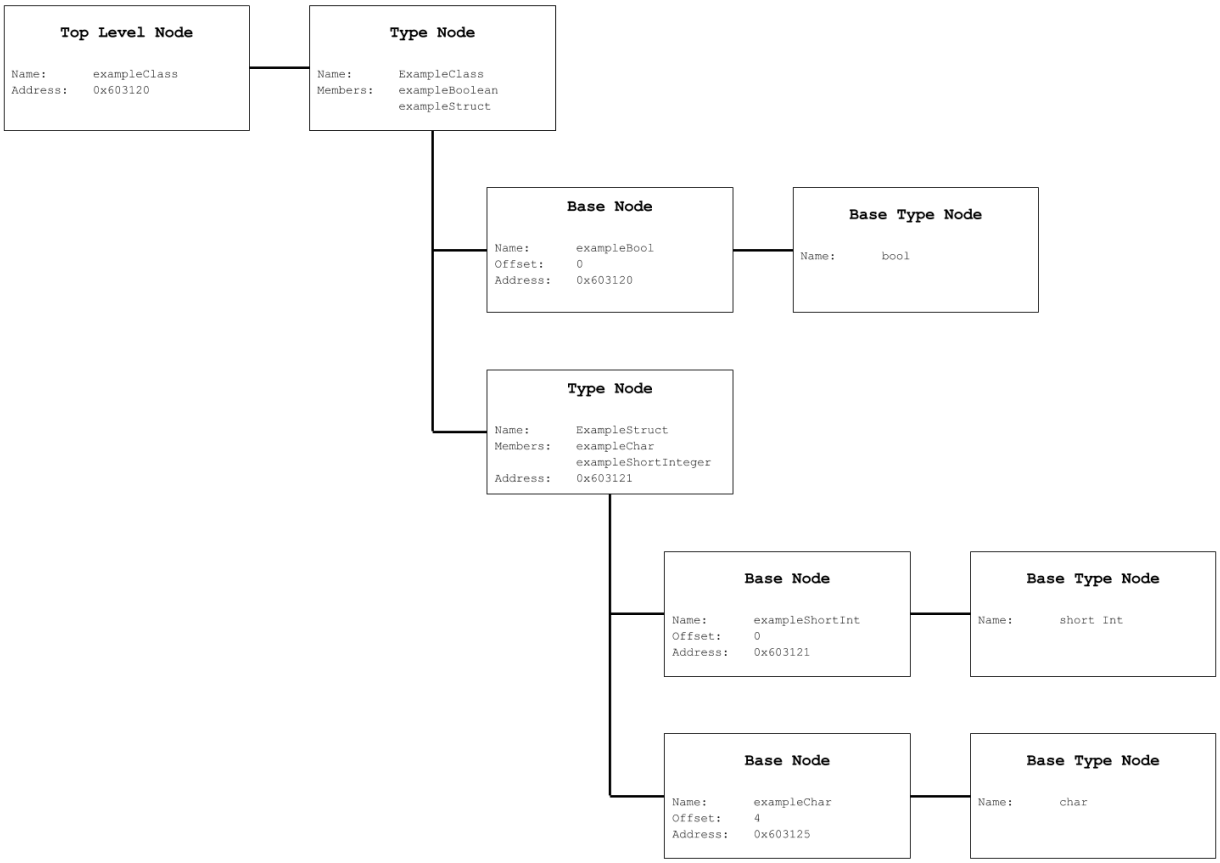

Abbildung 4.6: Grafische Darstellung der Verlinkungsstruktur von den verschiedenen "Nodes"

Im Diagramm können einzelne Variablen zum Diagramm hinzugefügt und gelöscht werden. Dieses Feature ist optional und wird bei genüged Zeit umgesetzt.

Add Variable Fügt ein Variable zum Diagramm hinzu.

Remove Variable Entfernt ein Variable vom Diagramm.

Optionen

In diesem Bereich werden die Optionen des im moment aktiven Symbols angezeigt.

Refreshrate Ändert die Werteabfragerate des Symbols. Die Eingabe kann durch einen Slider und durch eine Spinnerbox erfolgen.

Weitere Optionen Weitere Optionen werden bei Bedarf eingefügt.

4.5.2 Alternative grafische Benutzeroberfläche

Dieser Benutzeroberflächenentwurf besteht aus zwei Hauptfenster. Ein Fenster ist für die Überwachung der Symbolparameter zuständig und das zweite Fenster bietet eine Übersicht der Symbole und dient zur Auswahl der zu überwachenden Symbole. Dies Konzept soll eine bessere Übersicht der Symbole und deren Parameter gewähren.

Überwachungsfenster

Das Überwachungsfenster bietet auf einen Blick eine Übersicht mehrer Symbole und derer Pa-

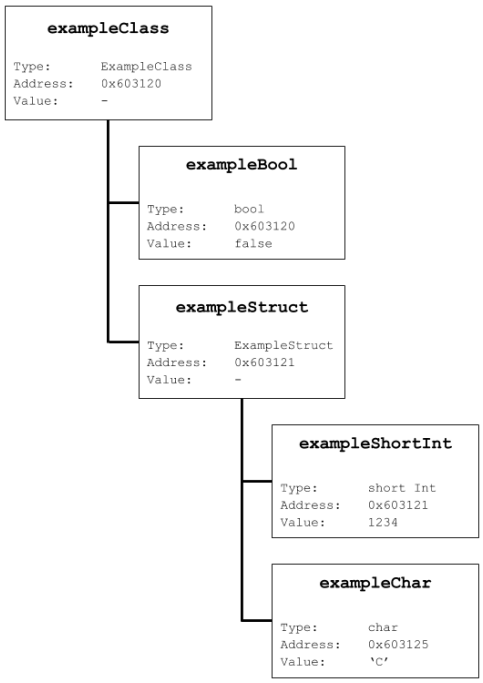

Abbildung 4.7: Grafische Darstellung des "Symbol Trees"

rameter. Das Fenster ist in zwei Teile unterteilt. Im oberen Bereich dem "Watcher" werden Symbole mit deren Parameter angezeigt, wie z.B. Datentyp und Wert. Ebenfalls können bei dem Optionensymbol Funktionen wie Maximalwert oder Durchschnittswert definiert und eingeblendet werden. Durch Klick auf den Add-Knopf öffnet sich das Symbolübersicht- und Auswahlfenster. Im unteren Breich befindet sich ein Diagramm "Graph" welcher für die Darstellung des zeitlichen Werteverlauf der Symbol zuständig ist.

Symbolübersicht- und Auswahlfenster

Im Symbolübersicht- und Auswahlfenster sind alle Symbole zu finden und können für die Überwachung im Überwachungsfenster ausgewählt werden. Das Fester besitzt verschiedene Darstellungen der Symoble welche über einen Reiter ausgewählt werden können. Es existiert eine Baumansicht der Symbole mit hierarchischer Verschachtelung, alternativ können die Symbole auch in einer einfachen Liste dargestellt werden, welche die Sortierung der Symbole ermöglicht. Für den schnellen Zugriff existiert eine Suche welche alle passenden Symbole in einer Liste ausgibt. Die Auswahl der Symbole kann auf zwei verschiedene Arten umgesetzt werden. Durch jeweils eine "check box" oder "dual list" für das Hinzufügen zum "Graph" und zum "Watcher".

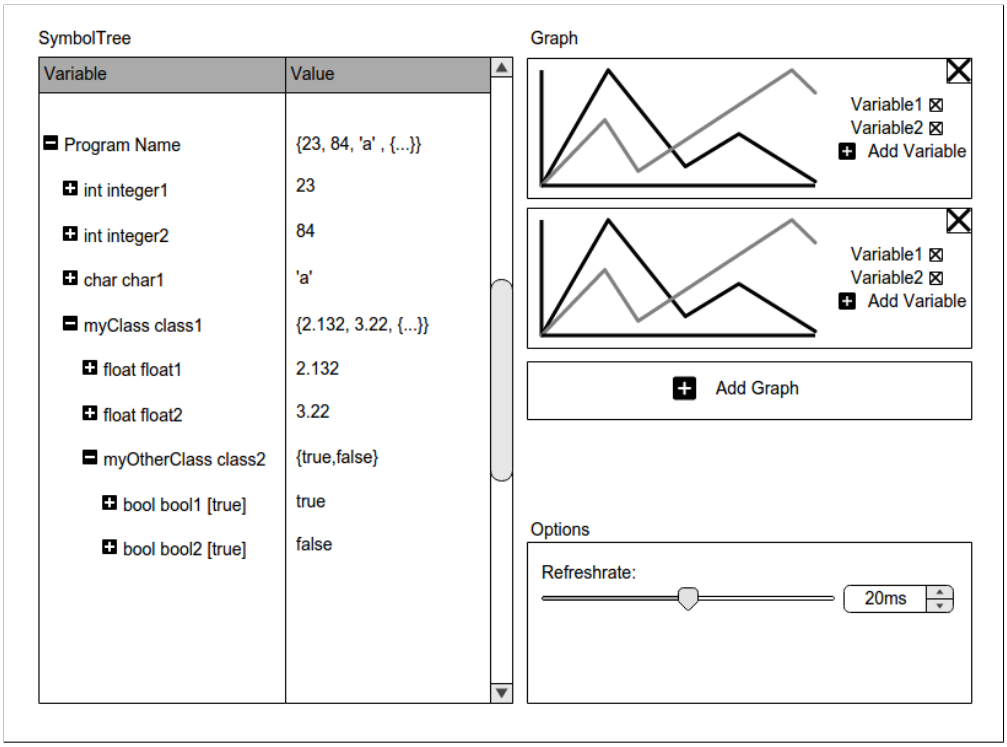

Abbildung 4.8: Mock-up von grafischer Benutzeroberfläche

Abbildung 4.9: Unterteilung der grafischen Benutzeroberfläche

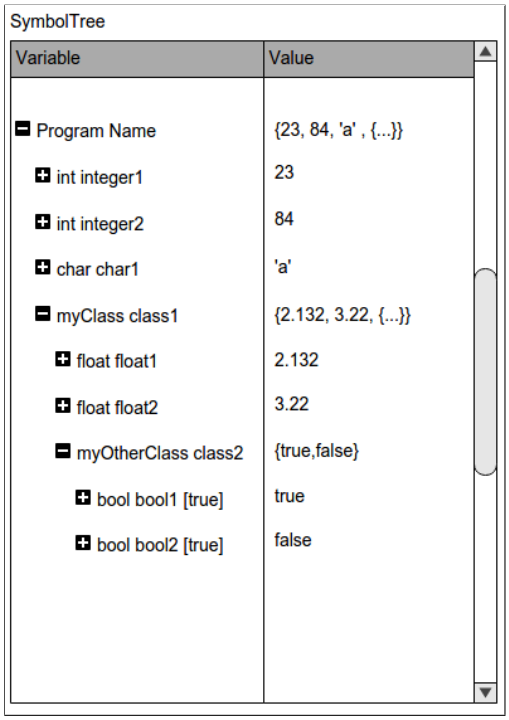

Abbildung 4.10: Bereich des Symbolbaum der grafischen Benutzeroberfläche

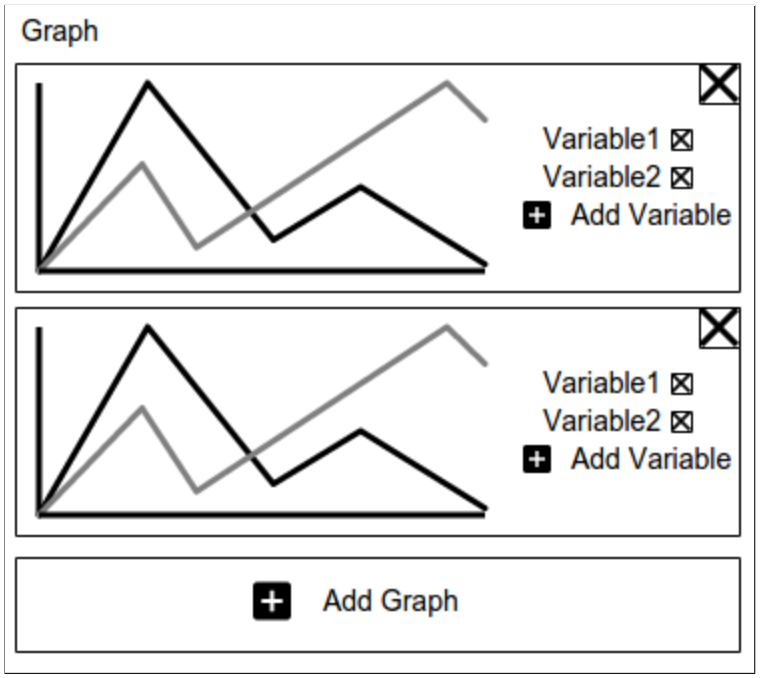

Abbildung 4.11: Bereich der Diagramme der grafischen Benutzeroberfläche

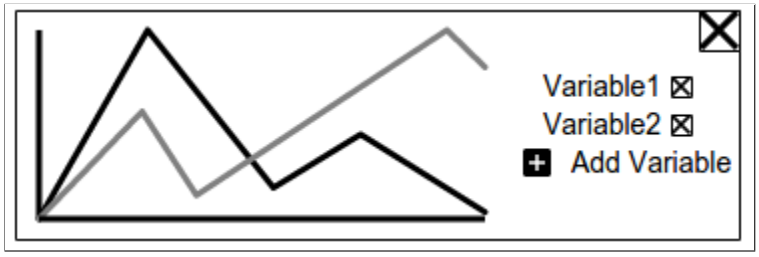

Abbildung 4.12: Beispiel eines Graphens der grafischen Benutzeroberfläche

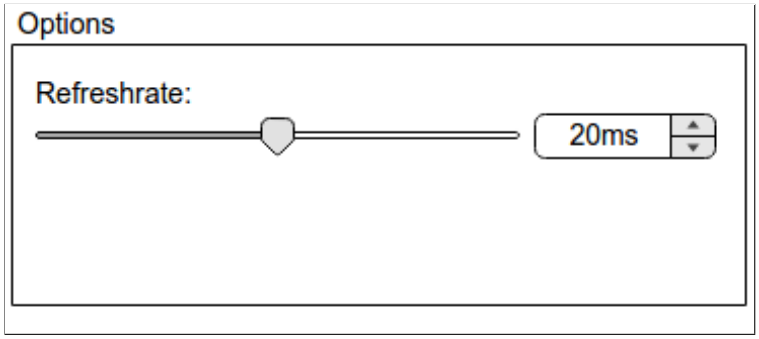

Abbildung 4.13: Bereich der Optionen der grafischen Benutzeroberfläche

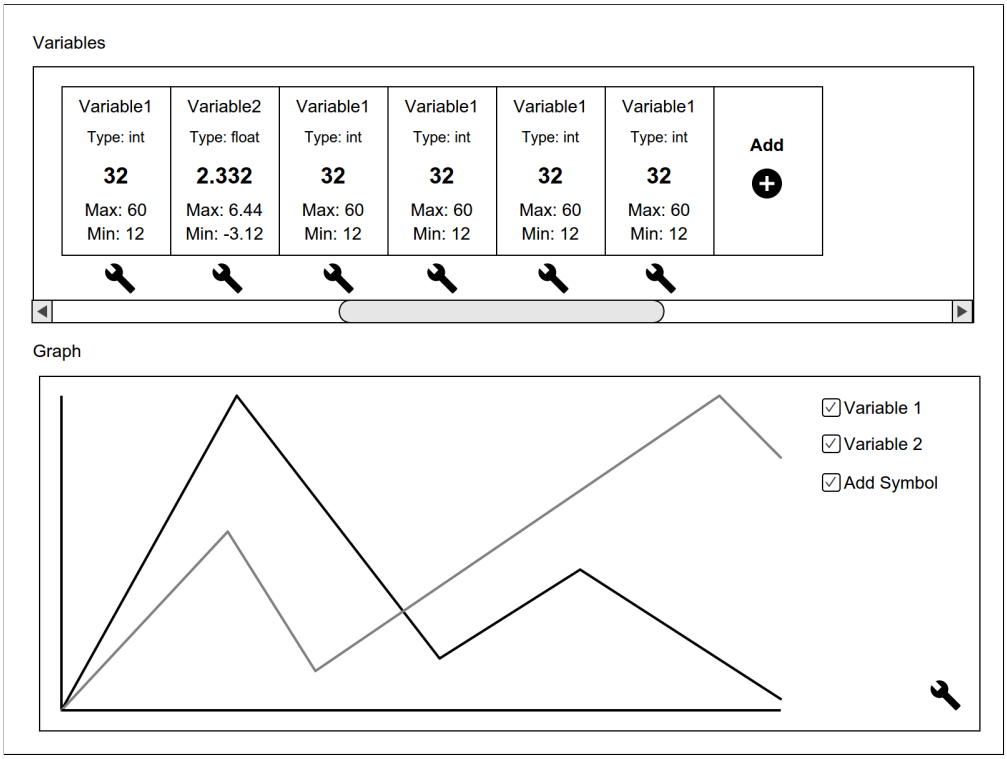

Abbildung 4.14: Mock-up von der alternativen grafischen Benutzeroberfläche zur Überwachung der Symbolparameter

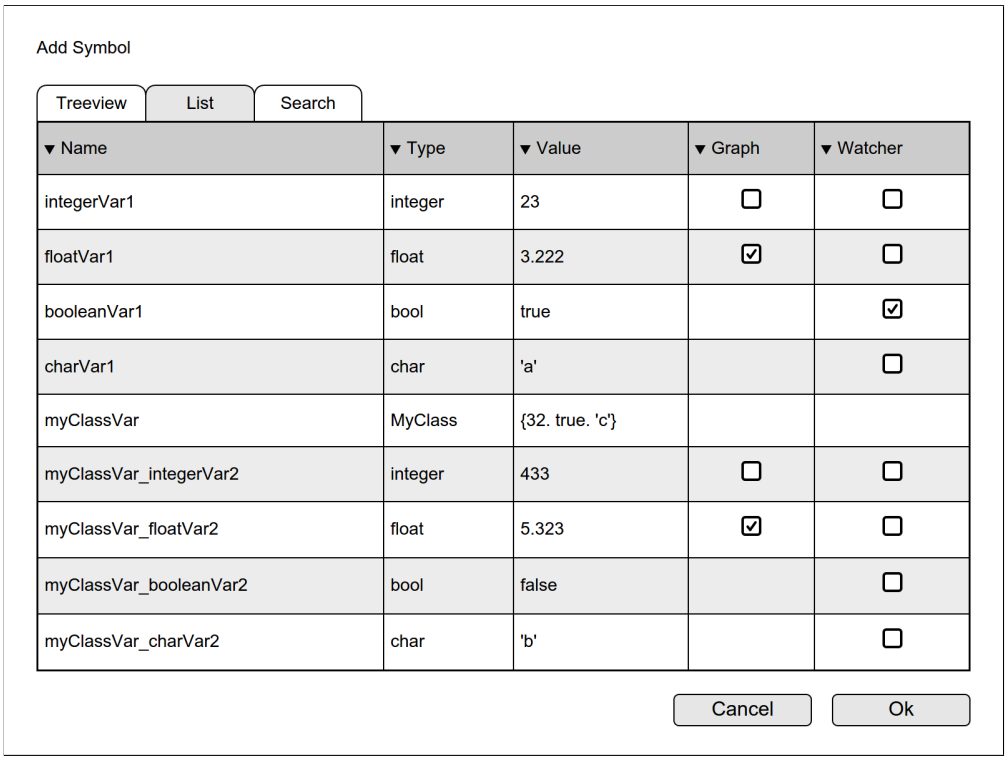

Abbildung 4.15: Mock-up von der alternativen grafischen Benutzeroberfläche zur Symbolauswahl in der Variante "check box".

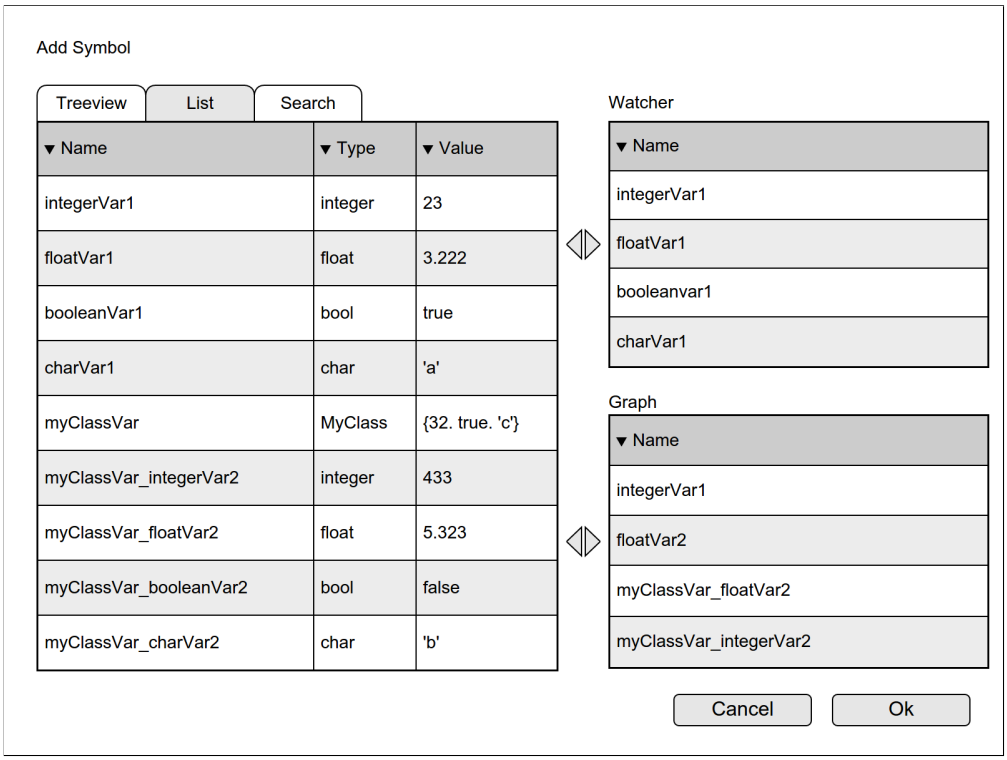

Abbildung 4.16: Mock-up von der alternativen grafischen Benutzeroberfläche zur Symbolauswahl in der Variante "dual list".

5 Ablauf und Tests

5.1 Ablauf

Für die Umsetzung dieses Projekts wurde der Ansatz gewählt, das für das Herausfiltern der benötigten Symbole, die Information aus der Symboltable verwendet wird. Dies dient dazu die Informationsflut zu verringern, da sonst auch Elemente von Libraries usw. die Ausgabe ungemein aufblähen würden. Dabei wird mittels einer Funktion der libelfin Library, in einem ersten Schritt, eine Liste von demangelten Namen erstellt, die dann für die dann weitere Verwendung gespeichert werden. Darauf folgend, werden aus dee DWARF File Information die top DIEs die für die Compile unites stehen ausgelesen. Dazu wird wiederholt eine Funktionen der libelfin library genutzt. Hier ist zu erwähnen das libelfin auftrumpfen kann, da aufgrund der bereitgestellten Features, über diese compile unites iteriert werden kann. Während dieser Iteration, wird im Anschluss für jedes Kind ein neues Objekt erstellt, in denen die notwendigen Teile abgespeichert werden. Diese Objekte werden in zwei Orten abgespeichert. Erstens in einer Liste die Nodes abspeichert, die als benötigte Nodes aus dem Elf File gelesen wurden, und in einer Map die zur Abspeicherung aller Nodes dient. In Dieser Map wird als Schlüssel der Offset des Nodes verwendet. Das Erstellen eigener Objekte, anstelle des verwenden der bereits vorhandenen Objekte der libelfin Library wurde notwendig, da das springen von einem Objekt auf ein anderes sofern es über mehrere compile units ging nicht möglich ist, dieses Vorgehen ist aber im notwendig. Sobald alle nötigen Nodes extrahiert und abgespeichert wurden werden sie mit ihren Kindern verlinkt. Um das zu erreichen wird die Definition der Nodes abgerufen, was mittels eines Attributs geschieht das im Node gespeichert ist. Dieses Attribut enthält die Offset Adresse die als Schlüssel in der Map fungiert. Von der Definition her werden dann die Kinder hinzugefügt. Für jedes dieser Kinder wird anschliessend die Funktion rekursiv nochmals aufgerufen, bis so der komplette Tree erstellt ist. Wenn das geschehen ist wird er auf der graphischen Benutzeroberflache mittels eines Tree-widgets angezeigt.

5.2 Performance

Durch abändern der Struktur auf neu erstellte Objekte Kamm es in einer ersten Version zu grossen performance Einbrüchen (Basisversion). Um dem entgegen zu wirken, wurden die Datentypen von listen auf Vektoren abgeändert und anschliessend der Speicher vorgängig reserviert. Diese Massnahme sorgte dafür, dass eine leichte Performance Steigerung wahrgenommen werden konnte. In einem nächsten Schritt wurde bei eine weiteren Analyse ein Fehler in der Erstellung der Objekte erstellt, durch dessen Beseitigung schlussendlich ein Teil der Leistung wieder hergestellt werden konnte. In Der Nachfolgenden Liste sieht man noch die Werte für die Verarbeitung des ABB ELF Files.

ABBfile	Zeit	Node Gesamt	Nodes verwendet
-	2:57:02.9	1,642828	45200

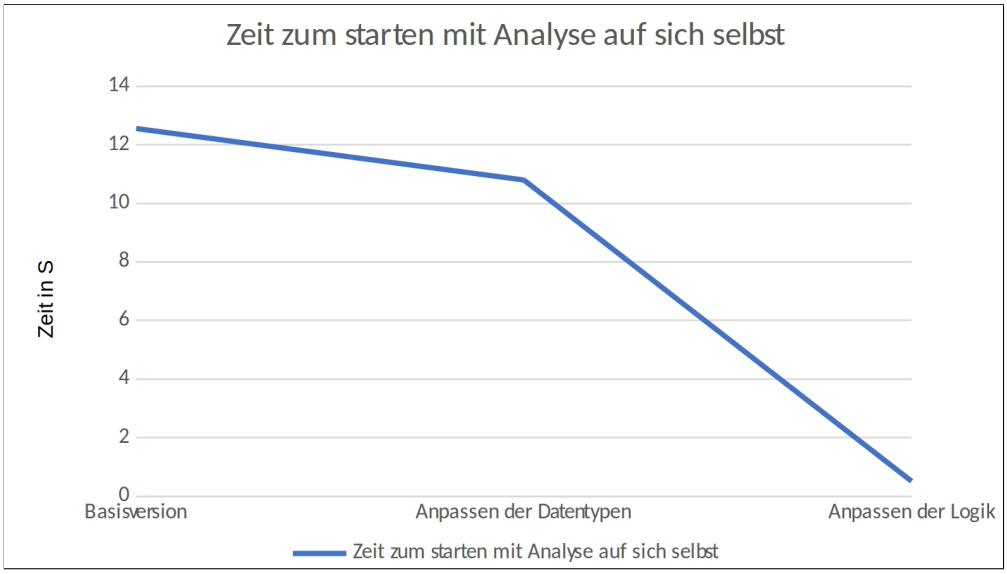

Abbildung 5.1: Leistungssteigerung der verschiedenen Versionen

5.3 Tests

5.3.1 Unit-Tests

Test	Vorgehensweise	Reaktion	Resultat
Applikationsstart	-	-	-
Einnlesen von einem Standardfile	Übergebe Pfad and Applikation	Erfolgreiches einlesen der Datei	Passed
Ungültiger Pfad beim einlesen	Übergebe ungültigen Pfand an Applikation	Applikation beendet	Passed

Qcostum Plot	-	-	-
Ausgabe eines numerischen Wertes	Wähle variable mit numerischer Natur	Graph wird ausgegeben	Passed
Übergabe eines nicht Nummerischen wertes	Wähle Variabel mit nicht numerischer Natur	Keine ausgabe des Graphen	Failed

CreatTree	-	-	-
Treitem ohne Kinder	Übergebe Node ohne Kinder	Erstellt nur einen Node	Passed
TreeItem mit Kindern	Übergebe Node mit Kindern	Erstellt eine Treeview der Items	Passed

Linker	-	-	-
Node mit Type und ohne Kinder	Rufe Linker mit Korektem Node ohne Kinder auf	Node bleibt unverändert	Passed
Node ohne Type	Rufe linker auf mit Node ohne Type	Node bleibt unverändert	Passed
Node ungültigem Type	Rufe linker auf mit Node mit ungültigem Type auf	Node bleibt unverändert	Passed
Node mit falschem Type	Rufe linker auf mit node der falschen Type besiitzt	Füge falsche Kinder dem Node hinzu	Passed
Node mit Type(mit Kindern)	Rufe linker mit Korektem Node auf	Kinder werden zum Node hinzugefügt	passed

Dumptree	-	-	-
Erstellen einer Node für DWARF DIE	èbergebe DIE an Funktion	Node mit Informationen wird erstellt	Passed
Übergebe leehres DWARF Objekt	Führe Funktion aus mit Lehrem DWARF Objekt	Erstellt leehren Node	Passed

Mainwindow	-	-	-
Übergebene Datei enthält keine Dwarfinformation	Lade ein File ohne DWARF information in die Applikation	Keine Daten werden eingelesen	passed
Datei mit DWARF Information wirt übergeben	Lade Vorgesehenes File in Applikation	Tree wird ausgegeben	Passed

Symbolfinder	-	-	-
Übergabe falschen Pfades an Symbolfinder	Übergebe ungültigen Pfad an Symbolfinder	Applikation beendet	Passed
Symbol in Elf-File bereits demangled	Applikation liest bereits demangeltes Symbol ein	Symbol wird nicht weiter verarbeited	passed

5.3.2 Usability tests

Aufgrund der Tatsache das es sich bei dieser Applikation um eine Anwedung für Informatiker handelt , haben wir bei den Usability Tests auf komolitonen aus dem Studiengang Informatik zurückgegriffen. Die Tests wurden dabei mit Papierprototypen durchgeführt.

Vorname Nachname	Céline Ott
Alter	24
Beruf	HSR Student
Beurteilunng	Die Appliaktion macht einen Symblen Und übersichtlichen Eindruck. alles sit wo es sein soll und man hat keine Probleme alles zu finden.

Vorname Nachname	David Meister
Alter	29
Beruf	HSR Student
Beurteilunng	Das Qui ist übersichtlich aber ich finden es wäre villeciht noch hilfreich wenn man den Graphen pausieren könnte, sonst sieht die Applikation gut aus.

Vorname Nachname	Andreas Stalder
Alter	24
Beruf	HSR Student
Beurteilunng	Eine aufgeräumte Anwendung allerdings wäre eine suchfunktion für den Tree noch gnaz cool sond ok

6 Projektmanagement

6.1 Projektplanung

Zur unterstützung der Projektplanung wurde MS Project in der Kombination von Google Sheets eingesetzt.

6.1.1 Zeitplan

Erste Kollisionen mit der Zeitplanung traten bereits relativ früh innerhalb des Projekts auf, so wurde zu viel Zeit bereits für die Literatur Recherche aufgewendet. Dieser Umstand in Kombination mit Problemen mit der Entwicklungsumgebung sowie unvorhergesehenen Implementationstechnischen Schwierigkeiten führten dazu das sie die Zeit kumulierte und es so zu einer erheblichen Verspätung innerhalb des Projektes kam.

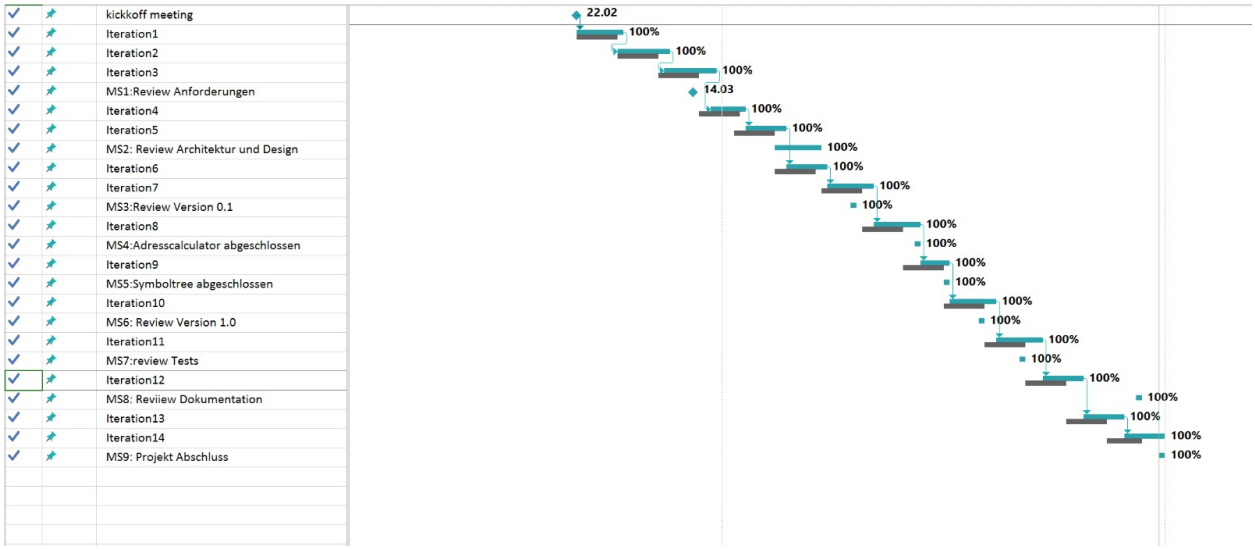

Abbildung 6.1: Zeitplanung Soll/Ist

6.1.2 Zeitaufwand

Zur Kompensierung des Mehraufwands durch den Eintritt der Risiken konnten die geplanten Zeiten nicht eingehalten werden.

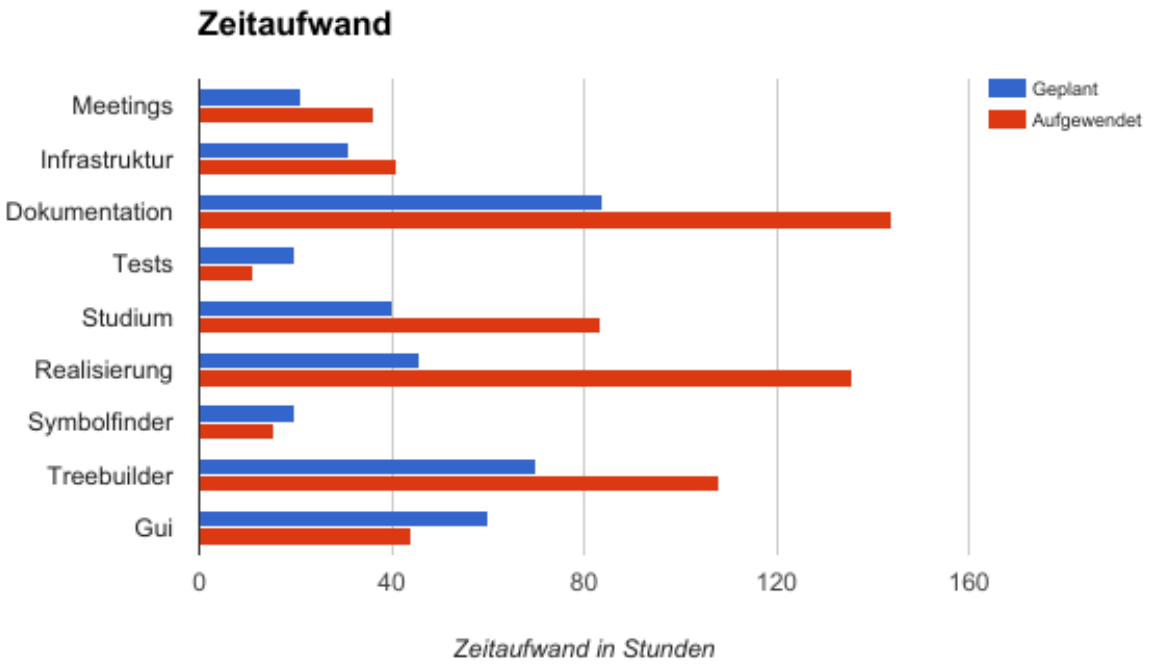

Abbildung 6.2: Aufgewendete Zeit für die Arbeit

6.2 Projektorganisation

Das Projektteam besteht aus zwei gleichgestellten Mitgliedern, Adrian Fröhlich und Samuel Krieg. Weil sich die Mitgliederzahl in Grenzen hält, wird kein Projektleiter benötigt. Wesentliche entscheide werden im Dialog zwischen den Projektmitglieder gefällt.

Vorname	Name	E-Mail
Samuel	Krieg	skrieg@hsr.ch
Adrian	Fröhlich	afroehlich@hsr.ch

Tabelle 6.1: **Teammitglieder**

6.2.1 Externe Schnittstellen

Stefan Richter fungiert als Betreuer und Bewerter der Projektarbeit.
stefan.richter@hsr.ch

Christof Gutscher ist die Kontaktperson zum Industriepartner ABB.
christof.gutscher@ch.abb.com

6.3 Management Abläufe

6.3.1 Meeting im Team

Die Team Meetings finden wöchentlich jeweils am Montag um 17:00 statt. Durchgeführt werden die Meetings an der HSR im Zimmer (1.206) mit einer Dauer von ca 30min bis 1h. Ziel ist es Fragen zu klären, Arbeiten zu verteilen und Reviews durchzuführen. Bei einem Team Meeting wird das weitere Vorgehen, sowie durchgeführte Arbeiten, fällige Arbeiten und auftretende Probleme besprochen.

6.3.2 Meeting mit Betreuer

Die Besprechungen mit dem Stefan Richter finden jeweils am Freitag von 15:00 bis 16:00 statt. Durchgeführt werden die Meetings im Büro von Stefan Richter

6.3.3 Meeting mit Industriepartner

Es sind eine Zwischenpräsentation sowie eine Abschlusspräsentation geplannt, ansonsten wünscht der Industriepartner keine weiteren Treffen.

6.4 Risikomanagement

Um den die Wahrscheinlichkeit eines Misserfolg der Arbeit zu minimieren sind im Risikomanagement die potentiellen Risiken bei der Umsetztung der Arbeit Identifiziert und Gewichtet. Ebenso sind vorbeugende Massnahmen definiert die den Eintritt eines Schadens minimieren sollen. Sollte ein Schaden dennoch eintretten ist das Verhalten festgelegt um den Schaden möglichst gering zu halten.

6.4.1 Risiko Identifizierung und Analyse

Es sind folgende Risiken Identifiziert:

Die Gewichtung der Risiken basieren auf der Erfahrung des Teams. Zur der Gewichtung des Schadens wurde die etablierte Formel dazu eingesetzt: Maximaler Schaden (h) x Eintrittswahrscheinlichkeit (%) = Gewichteter Schaden (h)

Für jedes Risiko sind präventive Massnahmen zur Risikoverminderung definiert, sowohl auch das Verhalten bei Risikoeintritt.

6.4.2 Umgang mit Risiken

Die aufgeführten Risiken sind in der Zeitplanung nicht speziell vorgesehen. Falls beim Eintreten eines geplanten Risikos ein erhöhter Zeitbedarf entsteht, so muss dies mit hoher Wahrscheinlichkeit mit Mehrarbeit der Teammitglieder kompensiert werden. Falls die nötige Mehrarbeit ausserhalb der Möglichkeiten liegt, so muss in Absprache aller Teammitglieder mit dem Betreuer nach einer anderer Lösung (z.B. Einschränkung von Programmfeatures, etc.) gesucht werden.

RisikoNr.	Titel	Beschreibung
R1	Probleme beim Auslesen der DWARF-Daten	Die verwendete Library kann nicht mit dem zu Verfügung stehendem DWARF-File umgehen.
R2	Probleme beim Auslesen der Symbole	Die gesuchten Symbole können mit der verwendeten Methode nicht aus der Symbol Table ausgelesen werden.
R3	Probleme beim Darstellen der Symbole	Die ausgelesenen Symbole können nicht wie gewünscht dargestellt werden.
R4	App-Start dauert zu lange	Die Applikation benötigt zu lange um aufzustarten.
R5	Performanceprobleme	Das Auslesen der Symbole dauert zu lange.
R6	Externe Komponenten	Es kommt zu Komplikationen mit externen Komponenten.
R7	Erhebliche Einarbeitungszeit in Technologien	Es braucht mehr Zeit als geplant um das Verständnis in externen Komponenten zu erlangen.

Tabelle 6.2: Identifikation von Risiken

6.4.3 Risikoüberwachung

Die Risiken werden periodisch in den Teammeetings kontrolliert und ebenfalls laufend beurteilt. Die dokumentation ist im Dokument "Technische Risiken.xlsx" zufinden.

6.4.4 Eingetretene Risiken

Während des Projekts sind folgende Risiken eingetreten. Trotz Risikoüberwachung konnte das Ausmass nicht eingedämmt werde. Die geplanten Risikoverminderung und das definierte Verhalten bei Eintritt war nutzlos oder konnte aus Zeitmangel nicht umgesetzt werden.

- R3 Das Risiko ist eingetreten und konnte mit geringem Mehraufwand gelöst werden.
- R4 + R5 Durch die unglaublich grosse Anzahl an zu verarbeitenden Nodes sind diese Risiken eingetroffen. Der Maximale Schaden wurde bei weitem überschritten ohne dass das Problem gelöst werden konnte.
- R6 Dieses Risiko ist durch einen Fehler in der Library eingetroffen. Ein Wechsel der Library stellte zu diesem Zeitpunkt aus Zeitmangel keine Alternative dar.
- R7 Trotz Tutor als Risikoverminderung ist dieses Risiko im vollen Ausmass eingetroffen. Der Hauptgrund ist die falsche Wahl der Komponenten.

6.5 Qualitätssicherung

Im Einsatz sind verschieden Massnahmen die, die Qualität des Ouputs sicherstellen sollen.

RisikoNr.	Max. Schaden (Stunden)	Eintrittswahrscheinlichkeit (Prozent)	Gewichteter Schaden (Stunden)
R1	35	20	7
R2	45	25	11.25
R3	40	15	6
R4	10	10	1
R5	20	20	4
R6	40	20	8
R7	50	10	5

Tabelle 6.3: Gewichtung von den Risiken

RisikoNr.	Risikoverminderung	Verhalten bei Eintritt
R1	Spezifikation und Test von verwendeter Library bis MS2 abschliessen	Verwenden anderer Library
R2	Bis Abschluss MS2 testen ob verwendete Methode funktioniert	Nach alternativen Methode umschauen
R3	Ausarbeitung alternativen Darstellungsvarianten	Auswahl alternativen Darstellungsmethode
R4	Parallelisierung der einzelnen Komponenten und	Optimierung der Nebenläufigkeit und des Codes
R5	Machbarkeit mit Prototyp vorgängig testen. Abklärung bis spätestens MS3	Verwendung besserer Algorithmen
R6	Bis MS2 testen der Libraries	Verwenden alternativer Library
R7	Fachexperte beiziehen	Zeitplanung anpassen, evtl. Programmumfang reduzieren

Tabelle 6.4: Umgang mit Risiken

6.5.1 Projektmanagement

Es wird nach jeder Arbeitssession oder beim Wechsel einer Arbeit der Aufwand auf dem entsprechendem Ticket verbucht. Dies hilft den Zeitplan, Vorgaben und den Umfang einzuhalten.

6.5.2 Team Meetings

Durch die periodischen Team Meetings soll sichergestellt werden, dass die Kommunikation im Team erfolgreich abläuft. Ebenso werden Periodische Kontrollen durchgeführt wie zum Beispiel die Risikoüberwachung

6.5.3 Versionsverwaltungssystem

Durch verwendung eines Versionsverwaltungssystem wird eine Datenchaos verhindert. Code darf nur funktionstüchtig und fehlerfrei in das Versionskontrollsystem eingespiesen werden. Zusätzliche Features werden in separaten Branches entwickelt und erst nach vollständigem Abschluss mit den Main-Brach vereint.

6.5.4 Code Reviews

Zur Kontrolle der Codequalität wird jedes Feature von der zweiten Person betrachtet. Fällt die Kontrolle negativ aus, wird dies Kommuniziert und die Verbesserung gefordert. Imfalle das ein Teammitglied nicht mehr weiter kommt kann Pairprogramming angefordert werden.

6.5.5 Testing

Alle Teammitglieder sind angehalten zu Ihrem fabriziertem Code Testing zu bebetreiben.

6.5.6 Code Style Guidelines

Zur Verbesserung der Lesbarkeit und Vereinheitlichung des geschriebenen Codes findet der Qt Coding Style[1] verwendung.

6.6 Entwicklungsumgebung

Die Entwicklungsumgebung wird jedem Teammitglied selbst überlassen. Es sind jedoch folgende Softwarekomponenten zwingend zu verwenden.

6.6.1 Integrierte Entwicklungsumgebung

Als Entwicklungsumgebung wird ausschliesslich die Platformübergreifende QT Creator[2] verwendet. Durch die integrierten Werkzeuge zur Erstellung von Benutzerschnittstelle kann eine mögliche Problematik des Zusammenspiels der Komponenten ausgewichen werden. Vorteil haft ist die kostenlose Nutzung unter der GNU General Public License[3]. Durch die unterschiedlichen, von den integrierte Entwicklungsumgebung verwendet Build-Management-Tools(Make, CMake, QMake), bietet die Verwendung einer einheitlichen integrierte Entwicklungsumgebung den Vorteil der einfachst möglichen Zusammenarbeit.

[1] https://wiki.qt.io/Qt_Coding_Style
[2] https://www.qt.io/ide/
[3] http://doc.qt.io/qt-5/licensing.html

6.6.2 Versionsverwaltung

Für die Versionsverwaltung wird auf Git[4] gesetzt mit Github[5] als Internethostingservice. Zur Unterstützung bei merge conflicts wird Meld[6] eingesetzt.

6.6.3 Dateiverwaltung

Alle Dokumente und Dateien welche nicht durch das Versionskontrollsystem verwaltet werden sind auf Google Drive[7] abgelegt. Dies ermöglicht einen schnellen und eifachen Dateiaustausch und bietet trotzdem eine rudimäntere Versionskontrolle.

6.6.4 Projektmanagementsoftware

Als Projektmanagement Tool zur Zeiterfassung und Arbeitsverwaltung wird Project[8] verwendet.

6.6.5 Kommunikation

Die Kommunikation im Team findet im Instantmessenger Whatsapp[9] sowie über E-Mails statt.

6.6.6 Weiter Software

Es wird diverse weitere Software eingesetzt um die Arbeit zu unterstützen:

- MindMup2[10]: Erstellen von Mind Maps.
- draw.io[11]: Erstellen von Diagrammen.

[4] https://git-scm.com/
[5] https://github.com/
[6] http://meldmerge.org/
[7] https://www.google.com/drive/
[8] https://products.office.com/en/project/project-and-portfolio-management-software?tab=tabs-1
[9] https://www.whatsapp.com/
[10] https://www.mindmup.com/
[11] https://www.draw.io/

7 Schlussfolgerung

7.1 Ergebnisdiskussion

Die Arbeit war für beide Teammitglieder eine spannende Herausforderung mit Höhen und Tiefen die es zu bewältigen galt.

7.1.1 Vorgehensweise

Projektmanagement
Trotz aktivem Prokektmanagement lief nicht alles Rund. Die zu Beginn festgelegten Zeitenplanung konnte durch das unvorhergesehen gleichzeitige Eintreten mehrere Risiken, trotz Kompensatsionsversuch durch Erhöhung der Arbeitszeit, nicht eingehalten werden. Das Eintreten der Risiken führte ebenfalls dazu das Arbeitspakete hinzugefügt und korrigiert werden mussten. Da der Umfang der Applikation bereits auf ein Minimum reduziert war und die Arbeitszeit, ab Eintritt der Risiken, auf dass maximal mögliche erweitert wurde, litt die Qualität der Arbeit darunter.

Literaturstudium
Mangels Vorwissen und fehlendem Verständnis erwies sich das Literaturstudium als langwierig. Durch den Zeitmangel konnte nicht genügend Wissen erlangt werden was sich im späteren Verlauf des Projektes bemerkbar machte. So wurden Erkenntisse erst zu spät erlangt, welche eigentlich von beginn an von Nöten gewesen wären.

Evaluationen und Prototyp
Die Evaluation gestaltete sich schwierig als gedacht durch die mangelnde Erfahrung im C++ Umfeld. Denoch konnte mit erheblichem Mehraufwand die nötigen Komponenten gefunde werden. Ein vollständig funktionierender Prototyp konnte nicht umgesetzt werden. Dies rächte sich bei der Umsetztung der Applikation wo Probleme erschienen die vermutlich bereits beim Prototyp hätten erkannt werden können.

7.1.2 Technologien

C++
Mangels C++ Erfahrung im Team mussten wir uns zuerst in die Sprache einarbeiten was zusätzlich Zeit benötigte. Im laufe der Arbeit haben wir uns jedoch mit C++ angefreundet und die segmentation faults nahmen ab.

Qt Creator
Qt Creator als IDE erleichterte die Erstellung und Integration der Benutzerinterfacekomponente, allerdings sind die unterstützenden Funktionen für den Programmierablauf sehr spärlich und kaum besser als bei einem guten Texteditor.

qmake
Als Build-Management-Tool wurde qmake verwendet welches mit Qt Creator kommt. Die stellte

sich als grosse Fehlentscheidung und als einen der grössten Zeitfresser im Projekt heraus. Da qmake nicht sehr verbreitet ist, findet sich im Internet relativ wenig Material wie Tutorials und guten Anleitungen. Als Folge davon musste mittels tagelangem nervenzerreissendem Trial and Error das Wissen selbst erarbeit werden. Vom Einbinden und Verlinken von Libraries bis hin zu Konfiguration von Tests, nichts ging einfach von statten.

Travis
Der Einsatz von Travis für Continuous Integration stellte sich als gute Wahl heraus. Die Konfigurations und der Einsatz verliefen problemlos.

libelfin
Die Library erwies einen guten Dienst. Sie ist verständlich und einfach anzusprechen. Allerdings generiert die Library an einem entscheidenden Punkt einen Segmentation Fault welcher von uns nicht behoben werden konnte. Dies führte zu beträchtlichem Mehraufwand, da unsere geplante Implementierung umgestellt und erweitert werden musste.

Google Tests
Das Test-Framwork von Google funktionierte problemlos und glänzt mit einer guten Dokumentation.

7.1.3 Kommunikation

Die Kommunikation zwischen den Teampartnern lief zu Beginn problemlos ab. Jedoch mit der Steigerung des Arbeitseinsatz konnte nicht mehr immer am selben Ort gearbeitet werden. Dies verlagerte die Kommunikation auf den schriftlichen Weg, was sich auf die Verständigung negativ auswirkte. Missverständnisse häuften sich und spannten die Situation an.

7.1.4 Menschliche Aspekte

Die Teamkomposition erwies sich zu Beginn als positiv, stellte sich jedoch im späteren Verlauf des Projektes als nicht optimal heraus. Einerseits weil verschiedene Arbeitsmetalitäten im Team vorhanden waren und andereseits weil sich der Druck erhöhte und Missverständnisse durch den suboptimalen Kommunikationskanal häuften. Dies veränderte den Umgang im Team in das Negative, senkte die Effizienz und erschwerte die Zusammenarbeit massiv.

7.2 Erkenntnisse

Dies Projekt hat unser Team um einigen Erkentnissen bereichert. Wir haben Erfahrung gewonnen in der Zeitschätzung für das Erarbeiten von neuen Technologien, und dem Literaturstudium. Beim nächsten Projekt werden wir die Erfahrung zu Zuge kommen lassen und einige Zeit mehr einplanen für diese Punkte. Bei der Thematik der Risiken haben wir uns sowohl im Ausmass wie auch in der Eintrittswahrscheinlichkeit verschätzt. In Zukunft werden wir versuchen die Risiken besser einzuschätzen und vorallem verstärkten Fokus auf die Risikominimierung legen.
Es hat sich ebenfalls gezeigt, dass das getrennte Arbeiten an verschiedenen Orten die Leistungsfähigkeit senkt und reduziert werden sollte.
Bezüglich der eingesetzten Technologien wurde uns bewusst wie wichtig der Bekanntheitsgrad als Auswahlkriterium ist. Wir werden in Zukunft dies in unsere Evaluierung einfliessen lassen.

7.3 Resultate

Die entwickelte Applikation deckt die erwünschten Kernpunkte ab. Allerdings konnte ein Teil der definierten Anforderungen nicht erfüllt werden. Ebenso konnten trotz massiven Performanceverbesserungen die nicht funktionale Anforderungen dies bezüglich nicht erfüllt werden. Bei der Qualität der Umsetztung mussten durch Zeitmangel Einbussen in Kauf genommen werden um die wichtigsten Anforderungen zu erfüllen.

NFA1: Startzeit Diese Anforderung wurde nicht erfüllt. Da es bei der Umsetztung zu nicht lösbaren Performanceproblemen gekommen ist. Ereichen könnte man diese Anforderung auf Kosten von NFA2 in dem die Erstellung des SymbolTrees zur Buildzeit ausgeführt wird.

NFA2: Buildzeit des Symbol Trees Diese Anforderung wird erfüllt, da zur Buildzeit nichts berechnet wird.

NFA3: Betriebssystem Diese Anforderung wurde nicht erfüllt. Das Programm läuft Out of the Box nur mit Linux.

NFA4: Wartbarkeit Diese Anforderung wurde erfüllt.

NFA5: Programmiersprache Diese Anforderung wurde erfüllt.

FA1: Berechnung von Adressen Diese Anforderung wurde erfüllt.

FA2: Symbol Tree Diese Anforderung wurde erfüllt.

FA3: Aktualisierung der Werte Diese Anforderung wurde nicht erfüllt. Es musste aus Zeitgründen auf den UC3 verzichtet werden welcher diese Anforderung umfasst. Die Integration sollte jedoch mind einem minimalen Aufwand machbar sein.

FA4: Darstellung des Werteverlaufs Diese Anforderung wurde erfüllt.

7.4 Ausblick

Verbesserungs- und Weiterentwicklungspotential ist vorhanden. Grundsätzlich könnte das Erfüllen aller gestellten Anforderungen angestrebt werden. Für die Weiterentwicklung können zusätzliche Use Cases gefunden und hinzugefügt werden.

7.4.1 Verbesserung

Performance
Die Performance könnte weiter gesteigert werden durch hochgradige Parallelisierung mittels GPGPU (General Purpose Computation on Graphics Processing Unit).

Code Qualität
Durch Refactoring könnte die Codequalität gesteigert werden.

Betriebssystem
Anpassung und Erweiterung des Build-Files zur Lauffähigkeit unter Windows.

Drag and Drop
Integration einer "Drag and Drop"-Funktion um Variablen für das Diagramm auszuwählen vereinfacht auszu

7.4.2 Weiterentwicklung

Symbolsuche

Um das Finden von Symbolen im der Treeview zu vereifachen könnte eine Suche integriert werden.

Alarmierung

Auf Symbolen könnten Triggerpunkte definiert werden die bei entsprechendem Symbolwert eine Alarmierung auslösen.

Exportfunktion

Es könnte eine Exportfunktion integriert werden welche, beispielsweise den SymbolTree mit allen zugehörigen werten Exportiert. Oder es könnte der zeitliche Verlauf des Variablenwertes exportiert werden.

8 Lizenzen

Qt

Qt is available under different licensing options designed to accommodate the needs of our various users:

Qt licensed under commercial licenses is appropriate for development of proprietary/commercial software where you do not want to share any source code with third parties or otherwise cannot comply with the terms of the GNU LGPL version 3.

Qt licensed under the **GNU Lesser General Public License (LGPL) version 3**[1] is appropriate for the development of Qt applications provided you can comply with the terms and conditions of the GNU LGPL version 3 (or GNU GPL version 3).

QCustomPlot

If not noted otherwise, all source code and software is licensed under the **GNU General Public License version 3 (GNU GPLv3)**[2] or later.

Icons

You are free to:Share — copy and redistribute the material in any medium or format for any purpose, even commercially.**Creative Commons Attribution-No Derivative Works 3.0 Unported** [3]

[1] https://www.gnu.org/licenses/lgpl-3.0.html
[2] https://www.gnu.org/licenses/gpl.html
[3] https://creativecommons.org/licenses/by-nd/3.0/

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

allocation Ist der Prozess einnem Programm physischen oder virtuellen speicher zuzuweisen.. 45

Basis Datentyp sind die Grunddatentypen wie zum Bsp. Int oder Long. 45

Byte Informationseinheit. 45

Check Box ist ein GUI-Widget welches dem Benutzer erlaubt binäre Entscheidungen zu treffen.. 45

compiler Werkezug/Programm zum übersetzten Von Code.. 45

cross-Platform Beschreibt den Zustand das etwas für mehrere Systeme oder Architekturen zur verfügung stehen kann.. 45

Debugger Werkzeug zum identifizieren von Fehlern in einem Computer Programm. 45

Dual List ist ein GUI-Widget welches aus zwei Listen besteht welche sich Listenelement teilen. Dient der Zuordung von Elemente zu einer der zwei Listen.. 45

Header Steht für den beginn eines Dokuments oder eines Datensatzes. 45

iteration Ist die Möglichkeit das durchgehen von Elementen in einem Container. 45

kernel zentraler Teil des Betriebssystem.. 45

kompression Verkleinern des benötigten speicher durhc anpassen des Datensatzes.. 45

libelfin ist ein C++ Library welche das Parsen von ELF/DWARF übernimmt.. 45

library Ressourcen für ein Computer Programm.. 45

linking Bschreibt das zwi Elemente miteinander verbunden werden.. 45

Node ist der englische Begriff für einen Knoten. Er representiert die einzelnen Daten in einer Baumstruktur.. 45

offset Unterschied zu einer Startadresse.. 45

opensource Code der öffentlich und unentgeltich zur verfügung gestellt wird. 45

relocation Der Prozess des zuweisens einer load Adresse für positions abhänigen Code. 45

source file Ein Filee das den Grund Code enthält.. 45

Symbol ist der englische Begriff für einen Bezeichner. Ein Bezeichner (selten auch Identifikator, englisch identifier) ist in der Informatik ein Identifikator, mit dem ein Programmierer in einem Programm ein Objekt, z. B. einen Datentyp, eine Variable oder eine Funktion, eindeutig benennt.[4]. 45

Symbol Table ist der englische Begriff für die Symboltabelle. In der Informatik ist eine Symboltabelle eine von Übersetzerprogrammen wie Compiler oder Interpreter verwendete Datenstruktur, die jedem Symbol im Quellcode Angaben wie die Stelle des Auftretens, den Datentyp oder einen Zeiger auf eine Struktur im Speicher zuordnet.[5]. 45

Symbol Tree ist der englische Begriff für den Symbol Baum. Der Symbol Baum ist die hierarchische Darstellung von Bezeichner(Symbols). 45

9 Literaturverzeichnis

[1] M. J. Eager, *Introduction to the DWARF Debugging Format*. Eager Consulting, February 2007.

[2] "Gcc and msvc c++ demangler." http://create.usc.edu/research/50788.pdf (21. Mai 2017).

[3] E. Eichhammer, "Qcustom plot." http://qcustomplot.com/ (28. Mai 2017).

[4] Wikipedia, "Bezeichner." https://de.wikipedia.org/wiki/Bezeichner (23. Mai 2017).

[5] Wikipedia, "Symboltabelle." https://de.wikipedia.org/wiki/Symboltabelle (23. Mai 2017).

10 Eigene Schlussfolgerung

10.1 Adrian Fröhlich

Die Arbeit selbst war an sich äusserst interessant und auch spannend allerdings würde ich im Nachhinein betrachtet viele Dinge unterschiedlich angehen respektive lösen. So hatten wir schon zu beginn unterschiedlichste Schwierigkeiten die uns zum Teil zu lange aufhielten und zu viel Zeit kosteten. Allerdings möchte ich an dieser Stelle auch erwähnen das uns Herr Richter immer tatkräftig für Fragen und Probleme mit Rat zur Seite gestanden ist, und ich möchte an dieser stelle ihm nochmals meine Danksagung kundtun.

10.2 Samuel Krieg

Die Durchführung dieser Studienarbeit im Frühlingssemester 2017 an der HSR, stellte mich vor die Aufgabe einen tiefen Einblick in die Funktionsweise von der Programmiersprache C++ zu werfen. Es stellten sich Fragen wie: Was ist eine ELF-File und wozu wird dieses benötigt? Ziel der Arbeit war es ein Programm zu schreiben, dass sich selbst oder ein anderes Programm analysiert und die darin vorkommenden Symbole mit den entsprechenden Datentypen und Werten in einer Baumstruktur darstellt. Die Werte der Symbole sollten durch ein Diagramm in ihrem zeitlichen Verlauf vom Benutzer beobachtet werden können. Dies Alles wurde im Auftrag des Industriepartners ABB durchgeführt und soll zur Komplettierung der Testsuit für Embedded-Systems von Mittelspannungs-Frequenzumrichter dienen. Gestartet hat das Projekt mit einem Kickoff-Meeting bei der ABB, bei welchem wir kurz in die Thematik eingeführt wurden. Als nächster Schritt wurde das Projekt für die Umsetzung geplant. Dies beinhaltete ein Literaturstudium gefolgt von Entwurf und der Implementierung des Programms. Über die Projektdauer wurden die verschiedenen Schritte und Überlegungen dokumentiert und schlussendlich in diesem Dokument zusammengeführt.

Mit dieser Arbeit musste ich mangels Alternativen einen Sprung ins kalte Wasser wagen. Wie sich herausstellte, bin ich tief abgetaucht. Meine fehlenden C++ Kenntnisse sowie die Zusammenarbeit im Team mit dem Partner, den ich kaum kannte stellten nur ein Teil der Herausforderung dar. Mir fehlte selbst das Eifachste Glossar, so wusste ich zum Beispiel, nicht was ein Symbol ist. Dieses Wissen konnte ich mir zwar im Verlauf der Arbeit aneignen. Die dazu benötige Zeit fehlte an anderen Orten, folglich konnte ich die Arbeit nicht so gewissenhaft wie gewünscht durchführen. Während des Projektverlaufs kamen weitere Hürden hinzu, die es zu bewältigt galt. So wurden die Risiken in der Projektplanung unterschätzt und bei deren Eintritt falsch behandelt. Als Folge davon geriet das Zeitmanagement durcheinander und Arbeiten mussten priorisiert werden. Wir priorisierten jedoch zu wenig konsequent, dies führte zur Verschwendung von wertvoller Zeit an falschen Orten und ist in Zukunft besser durchzuführen. Was mich während der Arbeit am meistens frustriert hat, waren die unterschiedlichen Qualitätserwartungen und das unterschiedliche Qualitätsempfinden im Team. Ich habe mehr als einmal die Nerven verloren, was darin resultierte, dass ich öfters mit dem Gedanken der Aufgabe der Arbeit spielte und mich nicht mehr korrekt gegenüber meinem Teampartner verhielt. Ich möchte mich hiermit bei dir, Adrian entschuldigen für mein teils inkorrektes Verhalten. Trotz all dem habe ich bei dieser Arbeit vieles über C++ und dessen Umfeld gelernt, und des Weiteren gelangen

mir wichtige Erkenntnisse im Bereich des Projektmanagements und der Projektplanung, welche ich als positiven Aspekt aus der Arbeit mitnehme.

Durch diese Arbeit bin ich um viele Erfahrungen, , welche ich nicht missen möchte, reicher geworden. Ich habe den Kontakt zur C++ gewonnen, was mir trotz anfänglichen Zweifel Freude bereitet hat. Im Bezug auf die Sprache C++ und die Umgebung nehme ich aus dieser Arbeit mit, dass der Spielraum in dem man sich bewegen kann, im Vergleich zu anderen Sprachen, erweitert ist. Es fehlt jedoch vielerorts an bekannten Hilfen und Unterstützungen die das Programmieren beschleunigen und vereinfachen. Im grossen und ganzen wünsche ich mir, die Arbeit wäre etwas reibungsloser ablaufen. So kann ich mit meiner Leistung nur sehr begrenzt zufrieden sein. Ich bin mir jedoch sicher, dass mir durch das Gelernte dieselben Fehler kein zweites Mal unterlaufen werden. Ich beende diese Arbeit mit einem positiven Blick zurück auf eine intensive Zeit.

Bei Prof. Stefan Richter möchte ich mich bedanken für die kompetente Betreuung, die aufgebrachte Zeit und benötigte Geduld. Seine Umgang weiss ich sehr zu schätzen.

Aufgabenstellung

Eingebettete Reglungssysteme arbeiten mit echten physikalischen Gegebenheiten, die nicht vollständig numerisch vorhersagbar sind. Um das Verhalten ihrer Produkte besser verstehen zu können, müssen Entwickler solcher System deshalb bestimmte Werte während der Durchführung physikalischer Experimente in Echtzeit beobachten können.
Andererseits sind die Entwickler zahlreichen Beschränkungen durch die Entwicklungsumgebung unterworfen; z.B. fehlen gute Debugger auf vielen Plattformen. Die Plattform, auf die diese Arbeit zielen soll, bietet einen Dienst, der auf Angabe der Adresse einer Variable deren Inhalt zurückgibt. Im Falle strukturierter, objekt-orientierter Programmierung ergibt dies einen sehr grossen Overhead für Entwickler, weil sie die absoluten Adressen jedes Daten-Members eines Objektes berechnen müssten, möglicherweise sogar rekursiv für verschachtelte Objekte.
Ziel dieser Studienarbeit ist die Erarbeitung eines Demonstrators, der zeigt, dass es grundsätzlich möglich ist, mit frei verfügbaren Bibliotheken den Inhalt von strukturierten Objekten auf dieser Plattform in einer graphischen echtzeit-fähigen Benutzeroberfläche darzustellen.

1. Der Demonstrator soll alle globalen Variablen und deren Daten-Member in einer Baumstruktur mit aufklappbaren Knoten anzeigen.
2. Der Benutzer kann den aktuellen Wert jedes Daten-Members erhalten, indem er einmal darauf klickt.
3. Der Benutzer kann eine Zeitspanne definieren, innerhalb der Inhalt des ausgewählten Daten-Members regelmässig erneut angezeigt wird.
4. Der Benutzer kann diese regelmässig erhaltenen Werte in verschiedenen Echtzeitdiagrammen anzeigen.

Das Hauptaugenmerk dieser Arbeit liegt auf der Client-Seite. Deswegen soll der Demonstrator anstatt auf das wirkliche eingebettete System zu zielen, sich selbst analysieren und darstellen können. Es ist auch Teil der Aufgabe, die Grenzen des Ansatzes aufzuzeigen (Anzahl Daten-Member, die simultan beobachtet werden können, kleinste Periode, etc), und bessere Methoden zur Darstellung schneller Echtzeit-Daten aufzuzeigen.
Alle Executables sind in DWARF, Standard-Komponenten, z.B. zum Parsen der DWARF-Files, sollen bevorzugt verwendet werden. Das GUI-Toolkit muss Qt sein.

Prof. Stefan Richter

Stefan Richter

Unterschrift

Rapperswil 2.6.17

Ort, Datum

Abbildung .1: Aufgewendete Zeit für die Arbeit

Eigenständigkeitserklärung

Erklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Adrian Fröhlich

Unterschrift

Rapperswil 2.6.17

Ort, Datum

Samuel Krieg

Unterschrift

Rapperswil 2.6.17

Ort, Datum

Abbildung .2: Aufgewendete Zeit für die Arbeit

Vereinbarung

Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit "Tool for debugging embedded control system" von Adrian Fröhlich und Samuel Krieg unter der Betreuung von Prof. Stefan Richter geregelt.

Urheberrecht

Die Urheberrechte stehen den Studenten zu.

Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von den Studenten, von der HSR wie von ABB nach Abschluss der Arbeit verwendet und weiter entwickelt werden.

Adrian Fröhlich

Unterschrift

Rapperswil, 2.6.17
Ort, Datum

Samuel Krieg

Unterschrift

Rapperswil, 2.6.17
Ort, Datum

Prof. Stefan Richter

Stefan Richter
Unterschrift

Rapperswil, 2.6.17
Ort, Datum

Prof. Dr. Markus Stolze

Unterschrift

2 Juni 2017 Rapperswil
Ort, Datum

Abbildung .3: Aufgewendete Zeit für die Arbeit