Visualisierung des Algorithmus *Dynamic Slicing*

Einführende Worte:

Ziel der Aufgabe war es, für die Lehrveranstaltung *Software Maintenance* ein Tool zu entwickeln, das sowohl den Lehrenden als auch den Studenten als zusätzliche Lernhilfe dienen kann. Es wurde sich dabei am Lehrveranstaltungsstoff orientiert. Für einen selbst gewählten Codeabschnitt lassen sich mit beliebigen Eingabewerte der *Execution Trace*, die Abhängigkeiten und der *Slice* berechnen und grafisch darstellen.

Warum C# und .Net?

Ich habe C# verwendet, da es die von mir am meisten geschriebene Programmiersprache ist. Als Visualisierungslösung habe ich mir eine Plattform unabhängige Anwendung vorgestellt, die einfach zu bedienen und schnell auszuführen ist. C# schien mir dabei die geeignete Sprache und .Net das geeignete Framework zu sein. Nach kurzer Recherche hat sich *Mono* als Mittel der Wahl herausgestellt, um eine .Net Umgebung auch auf Linux und wohlmöglich MacOS abzubilden. Zum Aufsetzen von *Mono* sind lediglich ein paar Kommandos von Nöten. Auf den ersten Blick hat Mono jede Funktionalität unterstützt, welche die Anwendung aufweisen musste. Tatsächlich lassen sich mit der aktuellen *Mono* Version 6.10.0.104 (https://www.mono-project.com/download/stable/) einfache *Buttons*, *Textboxen*, *Timer*, ein *DataViewgrid* und Striche zeichnen. Erst zum Ende der Entwicklung kam heraus, dass Mono bisher keine Pfeilspitzen unterstützt. Beim *Paint-Event* können bei Mono keine Start- und Endelemente von Strichen bestimmt werden. Stattdessen wird eine Meldung eingeblendet, die erklärt, wie die Pfeile eigentlich zu lesen wären. Bis auf diese Kleinigkeit funktioniert die Visualisierung des Dynamic Slice allerdings auf Windows und Linux problemlos.

Warum nicht MacOS?

Zu MacOS lässt sich sagen, dass Mono dort aktuell noch nicht unterstützt wird. Mono verwendet für die Funktionalität von Windows.Forms Carbon. Zum jetzigen Stand lassen sich damit keine .Net Anwendungen auf einem 64bit System ausführen (https://www.mono-project.com/docs/about-mono/supported-platforms/macos/#32-and-64-bit-support). Da bei MacOS schon seit langem keine 32bit Systeme vorkommen, wurde auf die Plattform MacOS verzichtet. Alternativ dazu lässt sich das Visualisierungstool auch auf einer Virtuellen Maschine mit Windows und Linux als Subsystemen ausführen.

Implementierte Klassen

Klassenname	Funktionalität
Form1.cs	Enthält alle Windows.Forms Elemente und administriert jede Aktion auf der Oberfläche.
FormCode.cs	In einem eigenen Fenster lässt sich der Code aussuchen, editieren und validieren.
CodeZeigen.cs	Im eigenen Fenster wird der ausgewählte Code mit jeweiligen Zeilennummern angezeigt.
DynamicSlicing.cs	Hier findet der Algorithmus statt, nachdem alle Eingaben getätigt sind. Der Algorithmus kann mit und ohne Zwischenschritte durchlaufen werden. Diese Klasse bedient sich einzelner Hilfsklassen.
Ebene.cs	Erhält einen formatierten Codeabschnitt und teilt ihn in hierarchische Ebenen ein, sodass Verschachtelungen einfacher durchlaufen werden können.
ETZeile.cs	Trägt Informationen über eine Zeile im *Execution Trace* z.B. ihre Funktion (while, for, if), *Def*- und *Ref*-Variablen und Elternzeile
ClassOperation.cs	Erhält ein mathematisches oder logisches Problem und Werte für Variablen, um daraus ein Ergebnis zu generieren.
ClassExecutionTrace.cs	Baut aus allen Ebenen einen *Execution Trace* auf, dessen Größe von den Eingabevariablen abhängt.
ClassDependencies.cs	Berechnet mit oder ohne Zwischenschritte die einzelnen Abhängigkeiten, nachdem der ET steht.
ClassSlice.cs	Berechnet den Slice, nachdem alle *Dependencies* berechnet wurden.
ClassArrowFeld.cs	Berechnet für die Pfeile der *Dependencies* die jeweilige Position auf dem *Dataviewgrid* möglichst Platz sparend.

Fazit

Mit C# und .Net lässt sich eine Anwendung implementieren, die sich auf Windows und Linux ausführen lässt. Das Visualisierungstool deckt einen großen Umfang an Funktionen ab. Das Tool erkennt Endlosschleifen, For-Schleifen können durch die Werte äußerer Variablen beeinflusst werden, Rechen- und Vergleichsoperationen können flexibel aufgebaut sein und Fehler des Benutzers werden abgefangen. Wenn man darüber

hinwegsieht, dass bei Linux die Pfeile nicht darstellt werden und stattdessen eine Meldung erscheint, wie die Pfeile richtig gehören, lässt sich das Tool gleichermaßen gut auf Windows und Linux einsetzen.