



Hochschule für Forstwirtschaft
Rottenburg

Hochschule für Angewandte Wissenschaften

BACHELORARBEIT

Entwicklung der Software „Wuchshüllenrechner“
zur Unterstützung der Entscheidung
„Zaun oder Einzelschutz“

Tobias Helfenstein
Freiherr-vom-Stein-Straße 37
55606 Kirn

I Allgemeine Angaben

Autor

Tobias Helfenstein | tobias.helfenstein@mailbox.org

Freiherr-vom-Stein-Straße 37
55606 Kirn

Studiengang: Forstwirtschaft
Vertiefung: Allgemeine Forstwirtschaft

Erstprüfer

Prof. Dr. Sebastian Hein | hein@hs-rottenburg.de

Professur für Waldbau, Waldbautechnik, Forstpflanzenzucht und Ertragskunde

Hochschule für Forstwirtschaft Rottenburg (HFR)
Schadenweilerhof
72108 Rottenburg am Neckar

Zweitprüfer

Dr. Anton Hammer | hammer.anton@gmail.com

HammerRuppert-Consulting

Leopoldstraße 19 A
76530 Baden-Baden

Zitierempfehlung

HELFENSTEIN, T.; HAMMER, A.; HEIN, S. (2016): Wuchshüllenrechner
Version 1.0.0. Hochschule für Forstwirtschaft, Fachbereich Waldbau

Copyright © 2016

72108 Rottenburg am Neckar

Alle Rechte, insbesondere das Recht der Vervielfältigung, Verbreitung und Übersetzung vorbehalten. Kein Teil des Werkes darf in irgendeiner Form ohne schriftliche Genehmigung reproduziert oder über elektronische Systeme verbreitet werden. Die Genehmigung ist bei der HFR einzuholen. Bei gesperrten Arbeiten ist jegliche Art der Weiterverwendung verboten.

II Vorwort

Die erste Idee zur Entwicklung einer computergestützten Kalkulationshilfe für den Vergleich von Zaun- und Einzelschutz hatte Herr Prof. Dr. Hein. Ende 2013 bot er mir an, zu dieser Thematik eine Abschlussarbeit zu schreiben, nachdem er im Jahrgang gehört hatte, dass ich mich außerhalb des Studiums intensiver mit Computern und speziell mit deren Software beschäftige. Anschließend habe ich einige Fachartikel zur Problemstellung gelesen, um daraufhin Herrn Prof. Dr. Hein das Interesse am Thema zu bestätigen. Dabei fand ich besonders die waldbaulichen Gedanken interessant. Gleichzeitig war ich hinsichtlich der betriebswirtschaftlichen und mathematischen sowie informatischen Lösung gespannt. Ich war umso mehr begeistert, weil die Entwicklung einer solchen Software für mich nicht das erste Projekt dieser Art war und ich somit auf die bereits gemachten Vorerfahrungen zurückgreifen konnte. Unabhängig davon hatte ich es mir schon länger zum Ziel gemacht, die Programmiersprache *Python 3* zu erlernen.

Dass es nun bis ins Jahr 2016 gedauert hat, bis die fertige Anwendung entwickelt werden konnte, ist vor allem dem Interesse an den Inhalten des Studiums geschuldet, da es nicht möglich ist, immer mal wieder zwischendurch am Projekt zu arbeiten und gleichzeitig alle Quelltextzeilen im Hinblick auf das Verständnis abrufbereit zu haben. Daher lässt sich eine Software leichter in einem zusammenhängenden Zeitraum schreiben.

Damit ich den *Wuchshüllenrechner* programmieren konnte, war eine leistungsfähige Hardware notwendig, die mir seit der Schulzeit von meinen Eltern zur Verfügung gestellt wurde. Dafür und für deren Unterstützung bin ich sehr dankbar. Des Weiteren möchte ich mich ausdrücklich bei allen Testpersonen bedanken und besonders bei den Herren *Kuchenbecker*, *Josten*, *Scherle* und *Fechner*.

Für die unkomplizierte Betreuung und Unterstützung möchte ich mich schließlich bei den Prüfern Herrn *Dr. Hammer* und Herrn *Prof. Dr. Hein* herzlich bedanken.

Der Verfasser

5. Juli 2016

III Inhaltsverzeichnis

I	ALLGEMEINE ANGABEN	III
II	VORWORT	IV
III	INHALTSVERZEICHNIS	V
IV	ABKÜRZUNGSVERZEICHNIS	VIII
V	ABBILDUNGSVERZEICHNIS.....	IX
VI	TABELLENVERZEICHNIS.....	X
VII	QUELLTEXTVERZEICHNIS	X
VIII	ZUSAMMENFASSUNG.....	XI
IX	ABSTRACT	XII
1	EINLEITUNG.....	1
1.1	Problemstellung und Zielsetzung	1
1.2	Gründe für eine Softwarelösung	3
2	THEORETISCHE GRUNDLAGEN	5
2.1	Funktionsweise der Entscheidungshilfe	5
2.2	Überlegungen zum Programmaufbau.....	7
2.3	Auswahl der Programmiersprache	11
2.4	Festlegung der Softwarelizenz.....	13
3	METHODISCHES VORGEHEN.....	15
3.1	Literaturanalyse	15
3.2	Entwicklung einer Software.....	16
3.2.1	Design des Programmablaufes	16
3.2.2	Nutzung von sprachentypischen Konventionen	18
3.2.3	Festlegung einer Dateisystemstruktur	19
3.2.4	Probelauf mit einzelnen Befehlen	19

3.2.5 Programmierung.....	20
3.2.6 Entwicklung der Diagrammansicht	21
3.2.7 Evaluation durch den Entwickler	23
3.2.8 Evaluation durch einen ausgewählten Nutzerkreis	25
4 ERGEBNISSE.....	27
4.1 Funktionsweise des Algorithmus.....	27
4.1.1 Sortierung nach Baumart	28
4.1.2 Kalkulation mittels Algorithmus.....	29
4.2 Die Programmoberfläche.....	32
4.2.1 Die Menüleiste.....	33
4.2.1.1 Der Menüeintrag Kalkulation.....	33
4.2.1.2 Der Menüeintrag Sprache.....	34
4.2.1.3 Der Menüeintrag Hilfe	35
4.2.2 Die Gruppe Datenerfassung.....	35
4.2.2.1 Die Listenansicht	37
4.2.3 Der Dialog Schutz bearbeiten	39
4.2.3.1 Die Eingabe der Baumart	42
4.2.3.2 Die Vorschlagswerte für Zaun und Wuchshülle	43
4.2.4 Die Diagramm- und Ergebnisansicht.....	44
4.2.4.1 Das Ergebnis	45
4.2.4.2 Die Kontrollleiste	46
4.2.4.3 Der Informationsbereich.....	48
4.3 Beispielkalkulation mit Berg-Ahorn.....	49
4.4 Ergebnis der Evaluation.....	52
5 DISKUSSION DES ERGEBNISSES	53
5.1 Berücksichtigung von Mischkulturen.....	53
5.2 Verständnisproblematik.....	59
5.2.1 Trennung von Zaun und Wuchshülle	59
5.2.2 Keine Übernahme der Kostenwerte	60
5.2.3 Einheiten der Kostenwerte und Eingabehilfen	61
5.2.4 Darstellung des Ergebnisses.....	63
5.3 Weitere Schutzverfahren	65

5.4 Waldbauliche Faktoren.....	66
5.5 Verwendung von Python 3.....	68
6 SCHLUSSFOLGERUNGEN UND AUSBLICK	71
6.1 Abschließende Bewertung	71
6.2 Bewertung der Evaluation	73
6.3 Ausblick.....	74
6.4 Hard- und Software der Entwicklungsumgebung	76
7 QUELLENVERZEICHNIS	79
7.1 Literatur	79
7.2 Internetseiten	80
7.3 Gespräche	81
7.4 Gesetzestexte	82
7.5 Kataloge.....	82
7.6 Bildinhalte.....	82
8 ANHANG.....	83
8.1 Einrichtung der Entwicklungsumgebung.....	83
8.1.1 Betriebssysteme von Apple	83
8.1.2 Betriebssysteme von Microsoft	85
8.2 Beispielanfrage zur Evaluation.....	86
8.2.1 Inhalt der E-Mail	86
8.2.2 Beispieldatensatz	88
8.2.3 Anleitung zum Entpacken und Starten	89
9 EIDESSTATTLICHE ERKLÄRUNG.....	91

Vorbemerkungen zum Sprachgebrauch

Nach dem Grundgesetz sind Frauen und Männer gleichberechtigt. Alle maskulinen Personen- und Funktionsbezeichnungen in dieser Arbeit gelten für Frauen und Männer in gleicher Weise.

IV Abkürzungsverzeichnis

AFZ	Allgemeine Forstzeitschrift
BAh	Berg-Ahorn (<i>Acer pseudoplatanus</i>)
CSV	Comma-separated values
EUR	Währungseinheit Euro oder €
FVA	Forstliche Versuchsanstalt Baden-Württemberg
GPLv3	GNU General Public License Version 3
GUI	Graphical User Interface
IDE	Integrated Development Environment
JPEG	JPEG File Interchange Format
KWF	Kuratorium für Waldarbeit und Forsttechnik
Kir	Vogel-Kirsche (<i>Prunus avium</i>)
Lfm.	Laufmeter oder laufender Meter
MVC	Model-View-Controller-Entwurfsmuster
OOP	Objektorientierte Programmierung
PDF	Portable Document Format
PNG	Portable Network Graphics
S.	Seite
St.	Stück
TIFF	Tagged Image File Format
UML	Unified Modeling Language
UStG	Umsatzsteuergesetz
VBA	Visual Basic for Applications
XML	Extensible Markup Language
cm	Zentimeter
f.	folgende
ff.	fortfolgende
m	Meter
m ²	Quadratmeter
py	Python Quelltextdatei mit der Endung <i>py</i>
pyc	Python Byte-Code-Datei mit der Endung <i>pyc</i>
vgl.	vergleiche

V Abbildungsverzeichnis

Abbildung 1: Model-View-Controller-Entwurfsmuster	9
Abbildung 2: Klassendiagramm für <i>Plant</i> , <i>Fence</i> , <i>Tube</i> und <i>VariantItem</i>	17
Abbildung 3: Probelauf mit einer Liste für Baumarten	20
Abbildung 4: Skizze der Diagramm- und Ergebnisansicht.....	22
Abbildung 5: Realisierte Diagramm- und Ergebnisansicht.....	22
Abbildung 6: Syntaxfehler im Quelltext.....	24
Abbildung 7: Beispiel der Ausgabeanweisung <i>print()</i> im Umfeld eines Fehlers	25
Abbildung 8: Der Aufbau des Hauptfensters	32
Abbildung 9: Die Menüleiste am oberen Rand des Hauptfensters	33
Abbildung 10: Alle Untermenüpunkte unter Kalkulation.....	33
Abbildung 11: Alle Untermenüpunkte unter Sprache.....	34
Abbildung 12: Alle Untermenüpunkte unter Hilfe	35
Abbildung 13: Alle Steuerelemente der Gruppe <i>Datenerfassung</i>	35
Abbildung 14: Eine Schutzvariante konnte nicht berechnet werden	37
Abbildung 15: Detaillierte Darstellung der Listenansicht.....	38
Abbildung 16: Der Dialog <i>Schutz bearbeiten</i> in der Voreinstellung.....	40
Abbildung 17: Detaillierte Diagramm- und Ergebnisansicht.....	44
Abbildung 18: Detaillierte Darstellung der Kontrollleiste	46
Abbildung 19: Die Liniendarstellung mit zwei Wuchshüllenvarianten	47
Abbildung 20: Detailansicht des Informationsbereiches	49
Abbildung 21: Veranschaulichung der Beispielkalkulation.....	51
Abbildung 22: Die Datenliste der Mischkultur im Wuchshüllenrechner	55
Abbildung 23: Mittels <i>Exportieren</i> erstellte Ergebnisgrafik des Vergleichs	56
Abbildung 24: Getrennte Auflistung von Zaun- und Wuchshüllenelementen	59
Abbildung 25: Abgefragte Kostenwerte für die Pflanze und die Pflanzung.....	61

Abbildung 26: Umrechnungshilfe für die <i>Anzahl der Pflanzen</i>	62
Abbildung 27: ToolTip am Informationssymbol	62
Abbildung 28: Aufbereitete Diagramm- und Ergebnisansicht	63
Abbildung 29: Ein nicht schutzbedürftiger Fichten-Naturverjüngungsvorrat	67
Abbildung 30: Sicherheitsrichtlinie des Betriebssystems	68

VI Tabellenverzeichnis

Tabelle 1: Entscheidungsmatrix für Programmiersprachen	11
Tabelle 2: Kostenstruktur der Berg-Ahorn-Monokultur	50
Tabelle 3: Veränderte Kostenstruktur für das Beispiel einer Mischkultur	54
Tabelle 4: Preisdifferenzen zwischen den Baumarten	54
Tabelle 5: Absolute Zahlenwerte	58
Tabelle 6: Differenzwerte von Berg-Ahorn und Vogel-Kirsche	58
Tabelle 7: Differenzwerte zwischen Berg-Ahorn und Vogel-Kirsche	58

VII Quelltextverzeichnis

Listing 1: Lizenzhinweis am Beginn jeder Quelldatei	13
Listing 2: Urheberrechtshinweis ebenfalls am Beginn jeder Quelldatei	14
Listing 3: Langform der bedingten Anweisung (if)	18
Listing 4: Kurzform der bedingten Anweisung (if)	18
Listing 5: Inhalt der Quelldatei <i>main.py</i>	21
Listing 6: Die Klasse <i>Plant</i> im Quelltext	27
Listing 7: Der in Python 3 realisierte Algorithmus	30

VIII Zusammenfassung

In vielen Forstbetrieben ist oftmals die Wilddichte so hoch, dass es zwingend notwendig ist, die gefährdeten Jungpflanzen vor Verbiss zu schützen. Dabei stellt sich die Frage, ob es sinnvoller ist, einen Zaun zu bauen oder einen Einzelschutz zu verwenden. Die Vielzahl an Einflussfaktoren und die komplexe ökonometrische Fragestellung führen oft zu einer Fehlentscheidung, die waldästhetisch unbefriedigend ist und hohe Ausgaben bedingt. Daher entwickelte Dr. Anton Hammer in seinem im Jahr 2012 erschienenen Fachbeitrag „Entscheidungshilfen zu: Zaun oder Einzelschutz mit Wuchshüllen“ bereits eine Lösung zur Klärung der Situation.

Mit der Software „Wuchshüllenrechner“ wurde, basierend auf dem Fachbeitrag, das dargestellte *Decision Support System* als computergestützte Kalkulationshilfe in deutscher und englischer Sprache erarbeitet. Sie erlaubt es, neben den Vorschlagswerten eigene Kostensätze einzugeben, um einen betriebswirtschaftlichen Vergleich zwischen Zaun und Einzelschutz durchzuführen. Das Ergebnis wird anschließend in einer anschaulichen Diagrammansicht dargestellt und erläutert. Daraus lässt sich insbesondere anhand der *Baumartenwahl*, der *Zaunlänge (Flächenumfang)* und der *Pflanzenzahl* eine rational begründete Schutzentscheidung ableiten. Darüber hinaus bietet die Anwendung die Möglichkeit, mithilfe von wechselnden Eingaben waldbauliche und ökonomische Rahmenbedingungen spielerisch zu testen und zu analysieren.

Im Allgemeinen war es bei der Entwicklung des *Wuchshüllenrechners* wichtig, eine benutzerfreundliche sowie plattformunabhängige Anwendung zu gestalten. Obwohl sich die Software insbesondere an Betriebsleitungen und Mitarbeiter des Controllings in größeren Forstbetrieben richtet, soll sie auch Unterstützung für die Forstpraktiker bieten. Speziell mithilfe der Umrechnungsdialoge lassen sich so die benötigten betrieblichen Kostenwerte praktisch ermitteln. Weil der *Wuchshüllenrechner* quelloffen bereitgestellt wird, darf das Programm frei genutzt und erweitert werden.

IX Abstract

In many forest enterprises the number of game animals is often too high and requires the protection of young forest plants to prevent deer browsing. Then the question occurs, whether either fencing or the use of tree shelters is economically more favourable. In many cases the large number of influencing factors and the complex econometric problem cause a wrong decision, which is forest aesthetically unsatisfactory and results in high expenses. Because of this Dr. Anton Hammer provided a solution to improve the situation, when he published his specialist article with the title “Entscheidungshilfe zu: Zaun oder Einzelschutz mit Wuchshüllen” in 2012.

The described decision support system was developed as a software solution with the application name “Wuchshüllenrechner”. This application is based on the named specialist article and offers a German version with an English translation. Besides the default values it is possible to enter the particular cost rates of the forest enterprise to compare the fence with the tree shelters. Afterwards the result is illustrated and explained in a descriptive chart view. With the parameters *tree species*, *fence length (perimeter)* and *number of forest plants* the users can derive a rational protection recommendation. Additionally, the application allows the users to playfully test and analyse silvicultural and economic conditions with variable input data.

During the development of the “Wuchshüllenrechner”, it was important to design a user-friendly and a platform-independent application. Although the software particularly addresses the management as well as the controllers of a forest enterprise, the application is supposed to support the foresters. Especially the cost translation dialogs support users to calculate the required cost rates. Because the “Wuchshüllenrechner” is open source software, it may be used and modified for free.

1 Einleitung

1.1 Problemstellung und Zielsetzung

Mit seiner Veröffentlichung „Entscheidungshilfen zu: Zaun oder Einzelschutz mit Wuchshüllen“ zeigte HAMMER bereits ein grundlegendes Problem insbesondere beim Schutz von Kulturen auf (vgl. HAMMER 2012, S. 19 ff.). Bei der Anlage einer Kultur ist es häufig notwendig, dass der Schutz der Pflanzen vor Verbiss und Fegen gewährleistet wird, weil der Wildbestand nicht ausreichend reguliert ist. Oftmals wird vom praktischen Forstbetrieb als Schutztyp ein Zaun verwendet, wesentlich seltener kommt vermutlich die Wuchshülle zum Einsatz. Die Gründe hierfür sind vielfältig.

Um den Praktikern die Entscheidung zwischen diesen beiden Schutztypen zu vereinfachen, entwickelte HAMMER eine Art Werkzeug, dem eine einfache Kostenkalkulation zugrunde liegt. Auf Basis der einfach zu ermittelnden Eingabeparameter *Anzahl der zu schützenden Pflanzen* und *Länge des Zaunes* entscheiden dann die Praktiker, welcher Schutz für die Kultur günstiger ist. HAMMER sieht die Notwendigkeit seines Werkzeuges darin begründet, dass der praktische Forstbetrieb viel zu selten eine ähnliche Kostenkalkulation durchführt und die Entscheidung nach Gefühl oder Erfahrung gefällt wird. Obwohl ein Zaun die Schutzlösung des Betriebes darstellt, könnte durchaus die Wuchshülle die betriebswirtschaftlich günstigere Lösung sein. Mit der erläuterten Entscheidungshilfe, in Englisch *Decision Support System* genannt, lässt sich deshalb eine rationale Begründung für die Verwendung des jeweiligen Schutztyps errechnen.

HAMMER gestaltet seine Kalkulation möglichst einfach und lässt dabei die wesentlichen waldbaulichen Schutzvor- und -nachteile von Zaun und Wuchshülle bewusst unbeachtet. Dennoch ist es notwendig, dass der jeweilige Betrieb die eigene Kostenstruktur im Hinblick auf die Anlage einer Kultur sowie deren Pflege und Schutz grundlegend erfasst. Aus diesen Grunddaten kann sich jeder Forstbetrieb eigene Entscheidungshilfen für die waldbaulich relevanten Baumarten ableiten und den Praktikern als Hilfsmittel zur Verfügung stellen. Denn üblicherweise

kennen die einzelnen Revierförster die betriebswirtschaftliche Kostenstruktur nur grob, sodass keine eigene Kalkulation möglich ist.

Am Beispiel der Bayerischen Staatsforsten ist zu sehen, dass sich größere staatliche Forstverwaltungen ähnliche Gedanken machen. Hierzu wurde für das gesamte Unternehmen ein betriebswirtschaftlicher Vergleich zwischen Zaun und Einzelschutz erstellt. Das Ergebnis stellt dar, dass in manchen Fällen die Wuchshülle die kostengünstigere Schutzlösung gewesen wäre. Dabei handelt es sich jedoch um eine Nachkalkulation (vgl. HAMMER 2016). Sinnvollerweise können die zuständigen Revierförster bei der Anlage einer Kultur auf ein ähnliches Hilfsmittel zurückgreifen, wie es HAMMER in seiner Veröffentlichung schildert.

Auf der bereits beschriebenen Grundlage möchte diese Abschlussarbeit ein computergestütztes Werkzeug erarbeiten, weshalb folgende Ziele formuliert wurden:

- die Veröffentlichung von HAMMER soll als Grundlage dienen
- Entwicklung einer einheitlichen und allgemeingültigen Computeranwendung als Entscheidungshilfe zwischen Zaun und Einzelschutz zur Vorkalkulation bei der Anlage einer Kultur
- das Ergebnis wird so aufbereitet, dass es für die Benutzer leicht verständlich ist
- Benutzer können innerhalb der Anwendung zwischen den Eingabedaten und dem Ergebnis unterschiedliche Ansätze mithilfe einer variablen grafischen Darstellung spielerisch testen und analysieren
- Übersetzung der Anwendung in verschiedene Sprachen: darunter Deutsch, Englisch und Japanisch
- im Allgemeinen soll die Software plattformunabhängig sein

1.2 Gründe für eine Softwarelösung

HAMMER hat mit seinem Fachbeitrag einen soliden Lösungsansatz veröffentlicht, den jeder selbst in einer eigenen Exceltabelle nachvollziehen kann. Dies führt jedoch dazu, dass unterschiedliche Umsetzungsvarianten im Umlauf sind, die im Einzelfall sicher gut funktionieren, aber weder einheitlich sind noch allgemeingültig verwendet werden können. Das ergab zum einen das Gespräch mit einigen Forstpraktikern. Zum anderen beweist die Suche im Internet, dass eine solche Computeranwendung für einen größeren Anwenderkreis noch nicht existiert. Damit wird die Entwicklung der Anwendung *Wuchshüllenrechner* begründet.

Mit der Weiterentwicklung der „Entscheidungshilfen zu: Zaun oder Einzelschutz mit Wuchshüllen“ als *Wuchshüllenrechner* wird von der Computeranwendung erwartet, dass die Benutzer nach der Eingabe der Datengrundlage für Zaun und Wuchshülle eine konkrete Entscheidungsempfehlung erhalten. Das Programm berechnet die *Funktion der Kostengleichheit* und zeigt mittels Diagrammdarstellung, ob der jeweilige Schnittpunkt zwischen *Zaunlänge* auf der Abszissenachse und *Anzahl der Pflanzen* auf der Ordinatenachse unter- oder oberhalb der Funktionsgeraden liegt. Damit sollen die Anwender leicht erkennen, welcher Schutztyp günstiger ist. Welche tatsächlichen Kosten mit dem jeweiligen Schutz verbunden sind, soll vorerst nicht berechnet werden.

Es stellt sich die Frage, ob der Kalkulationsprozess in Excel oder in einer eigenen Fachanwendung abgebildet werden soll. Dies hängt von den Rahmenbedingungen in Form der Zielsetzungen des Projektes ab. Besonders wichtig und aussagekräftig scheinen hierbei der Aspekt der *variablen grafischen Darstellung*, der Wunsch nach der *Übersetzung in verschiedene Sprachen* und die *Plattformunabhängigkeit* der Anwendung zu sein. Die Art der grafischen Darstellung ist in Excel auch bei der Verwendung der Programmiersprache *Visual Basic for Applications* limitiert. Das heißt, dass die Benutzer nicht mit dem Ergebnis innerhalb der Grafik interagieren können. Ähnliche Limitierungen gibt es im Hinblick auf die Übersetzung in verschiedene Sprachen. In Anbetracht der Plattformunabhängigkeit muss bei Exceltabellen davon ausgegangen werden, dass diese häufig nur unter dem Betriebssystem Windows in Verbindung mit Microsoft Excel fehlerfrei unterstützt

werden. Bei der Verwendung anderer Betriebssysteme und dem entsprechenden Ersatz für Microsoft Excel, beispielsweise OpenOffice Calc, kann der volle Funktionsumfang nicht gewährleistet werden. Folglich wurde nach der Bewertung dieser Kriterien entschieden, den *Wuchshüllenrechner* als Fachanwendung in einer gebräuchlichen und plattformunabhängigen Programmiersprache zu entwickeln.

Bei der Entwicklung einer solchen Fachanwendung ist es insbesondere hilfreich, die Zielgruppe zu betrachten. Daraus ergeben sich zwei interessante Fragen. Zum einen ist es wichtig, welche Forstfach- und Computerkenntnisse von den Anwendern erwartet werden können. Zum anderen ist es nützlich, die betriebsübliche Software der Anwender zu kennen, um bei der Oberflächenentwicklung der Fachanwendung einen hohen Wiedererkennungsgrad von Softwarefunktionen zu erreichen. Dies ermöglicht den Benutzern einen schnelleren Einstieg in das Programm und erleichtert die Bedienung. Somit können sich die Anwender mit dem Programm identifizieren.

Ein zentrales Problem tritt beim Vergleich der Schutztypen *Zaun* und *Wuchshülle* auf, weil zwei grundverschiedene Datensätze erfasst werden müssen. Deshalb muss bei allen Überlegungen zum Programmaufbau die Tatsache der *ungleichen Datensatzstruktur* beachtet werden. Es entsteht also ein Spannungsfeld zwischen Datensatzstruktur und der jeweiligen Darstellung in einer Tabelle oder einem Diagramm. Hinsichtlich des hohen Wiedererkennungsgrades ist es eine Schwierigkeit dieser Abschlussarbeit, das angedeutete Spannungsfeld optimal zu lösen. Dabei ist die Festlegung nützlich, wie viel Spielraum den Benutzern zugestanden wird und welche Vorgaben durch das Programm gemacht werden sollen. Aus diesem Grund fließen die zu erwartenden Anwenderkenntnisse in die Softwareentwicklung mit ein.

2 Theoretische Grundlagen

2.1 Funktionsweise der Entscheidungshilfe

Die Funktionsweise des Werkzeuges beschreibt HAMMER in seiner Veröffentlichung sehr präzise. Dabei werden die Eingabedaten in die beiden Kategorien *flächenspezifische Parameter* und *Kostenstruktur des Forstbetriebes* unterteilt. Zu den flächenspezifischen Parametern zählen die *Zaunlänge in Metern* und die *Anzahl der Pflanzen*. Diese Zahlen können die Praktiker bereits vor Ort erfassen. Die zweite Kategorie beinhaltet die Kostenstruktur und dient daher zur Kostenkalkulation. Üblicherweise gelten die jeweiligen Werte für den gesamten Forstbetrieb und müssen entsprechend ermittelt werden. Aus der Kalkulation resultiert die *Funktion der Kostengleichheit* zwischen den Schutztypen Zaun und Wuchshülle. Wird diese Funktion beispielsweise in einem Liniendiagramm dargestellt, bewegt sich der Graph zwischen der Zaunlänge auf der Abszissenachse und der Anzahl der Pflanzen auf der Ordinatenachse. Praktisch bedeutet Kostengleichheit, dass bei gegebener Zaunlänge und Pflanzenzahl die Kosten für den Zaun und die Wuchshülle gleich sind.

Zur Herleitung der Berechnung gruppiert HAMMER die Kosten. Somit bilden alle Werte für den Schutztyp Zaun die *Kostengruppe A*. Folglich setzt sich die *Kostengruppe B* aus den Werten für den Schutztyp Wuchshülle zusammen.

Beschreibung der *Kostengruppe A*:

- A_1 : Stückkosten der Pflanze [EUR/St.]
- A_2 : Stückkosten der Kulturvorbereitung [EUR/St.]
- A_3 : Stückkosten der Pflanzung [EUR/St.]
- A_4 : Stückkosten der Kultursicherung in den ersten fünf Jahren [EUR/St.]
- A_5 : Zaunkosten (Aufbau, Unterhaltung und Abbau) je Laufmeter [EUR/Lfm.]

Kalkulationsformel für die *Kostengruppe A*:

$$(1) \quad K_1 = x \cdot A_5 + y \cdot (A_1 + A_2 + A_3 + A_4)$$

Beschreibung der *Kostengruppe B*:

- B_1 : Stückkosten der Pflanze [EUR/St.]
- B_2 : Stückkosten der Kulturvorbereitung [EUR/St.]
- B_3 : Stückkosten der Pflanzung [EUR/St.]
- B_4 : Stückkosten der Kultursicherung in den ersten fünf Jahren [EUR/St.]
- B_5 : Stückkosten der Wuchshülle sowie Zubehör (Stab) [EUR/St.]
- B_6 : Stückkosten (Aufbau, Unterhaltung und Abbau) je Wuchshülle [EUR/St.]

Kalkulationsformel für die *Kostengruppe B*:

$$(2) \quad K_2 = y \cdot (B_1 + B_2 + B_3 + B_4 + B_5 + B_6) \cdot C$$

Zusätzlich fließt ein waldbaulicher Aspekt in die Kostenkalkulation mit ein, da für den Einzelschutz in Form der Wuchshülle eine geringere Mortalitätsrate gegenüber dem Zaun angenommen werden kann. Um dies zu berücksichtigen, wird die Summe der *Kostengruppe B* mit dem Reduktionsfaktor C multipliziert. In seinem Fachbeitrag geht HAMMER davon aus, dass der Ausfall der Pflanzen innerhalb der Wuchshüllen um etwa 10 % reduziert ist. Das entspricht dem Reduktionsfaktor $C = 0,9$.

Für jede Kostengruppe wurde bereits eine eigene Kalkulationsformel erarbeitet, wobei die Funktionsvariablen x und y die *Zaunlänge* und die *Anzahl der Pflanzen* darstellen. Beide Formeln (1) und (2) werden nachfolgend im mathematischen Sinn gleichgesetzt und umgestellt. Als Ergebnis ist die *Funktion der Kostengleichheit* in der allgemeinen Form (3) für das erläuterte Entscheidungsproblem zwischen Zaun und Wuchshülle beschrieben. Damit dient sie gleichzeitig als Berechnungsgrundlage innerhalb der Anwendung *Wuchshüllenrechner*.

Gleichsetzen der Formeln (1) und (2):

$$\begin{aligned}
 K_1 &= K_2 \\
 x \cdot A_5 + y \cdot (A_1 + A_2 + A_3 + A_4) &= y \cdot (B_1 + B_2 + B_3 + B_4 + B_5 + B_6) \cdot C \\
 x \cdot A_5 &= y \cdot (B_1 + B_2 + B_3 + B_4 + B_5 + B_6) \cdot C - y \cdot (A_1 + A_2 + A_3 + A_4) \\
 x \cdot A_5 &= y \cdot [(B_1 + B_2 + B_3 + B_4 + B_5 + B_6) \cdot C - (A_1 + A_2 + A_3 + A_4)] \\
 (3) \quad y &= \frac{A_5}{(B_1 + B_2 + B_3 + B_4 + B_5 + B_6) \cdot C - (A_1 + A_2 + A_3 + A_4)} \cdot x
 \end{aligned}$$

Die in (3) hergeleitete Formel folgt dem allgemeinen Schema $y = m \cdot x + n$ einer linearen Funktion. Dabei enthält die Variable y die *Anzahl der Pflanzen* und wird in Abhängigkeit der Funktionsvariablen x , die ihrerseits die *Zaunlänge* abbildet, berechnet. Die Steigung der Funktion, die im Schema durch die Variable m repräsentiert wird, ergibt sich aus dem mathematischen Bruch. Hierbei werden die Kosten für den Zaun je laufender Meter (A_g) durch die Differenzkosten der Kultur geteilt. Ermittelt werden die Differenzkosten, indem die Summe der Kulturkosten für den Zaun von der um den Faktor C reduzierten Summe der Kulturkosten für die Wuchshülle subtrahiert wird. Der im Schema beschriebene Ordinatenabschnitt n fällt weg. Deshalb kann von einer *homogen linearen Funktion* gesprochen werden. Sie ist aus mathematischer Sicht proportional.

Bei genauer Betrachtung der Eingabedaten fällt auf, dass HAMMER keinen Flächenbezug in seiner Kalkulation herstellt. Er berücksichtigt die Zaunlänge und bezieht sich demnach auf den Umfang der Fläche. Das ist sinnvoll, weil in Abhängigkeit der geometrischen Form der Pflanzfläche bei gleicher Flächengröße der Umfang variiert. Deshalb wird aus betriebswirtschaftlicher Sicht immer der reale Kostenwert für den Schutztyp Zaun ermittelt. So wird die Entscheidungshilfe praxistauglich, da es vor Ort wesentlich unkomplizierter ist, den Umfang einer Pflanzfläche zu ermitteln als die tatsächliche Flächengröße. Aufgrund des Pflanzverbandes kann aber mithilfe der Pflanzenzahl ein indirekter Flächenbezug hergeleitet werden.

2.2 Überlegungen zum Programmaufbau

Bevor der Kalkulationsprozess in einer Anwendung abgebildet werden kann, sind in technischer Hinsicht einige Überlegungen notwendig. Diese sind jedoch vorerst unabhängig von der später verwendeten Programmiersprache und daher allgemeingültig.

Im ersten Schritt werden die Eingabedaten betrachtet. Hierzu wird festgelegt, welche Informationen tatsächlich von den Benutzern abgefragt werden müssen. Es ist das Ziel, aus einer möglichst übersichtlichen Anzahl von Benutzerabfragen eine optimale Kalkulation durchzuführen. Somit werden die Anwender nicht

durch unnötige Eingaben belästigt. Zusätzlich zu der bereits zuvor beschriebenen Kostenstruktur des Forstbetriebes sollen folgende allgemeinen Eingaben im Programm gemacht werden:

- Forstbetrieb, Forstrevier, Revierleiter und Waldort
- Mehrwertsteuer (Regelbesteuerung für Vorschlagswerte)
- Schutztyp (Zaun oder Wuchshülle) und Schutzbeschreibung
- Zaunlänge und Anzahl der Pflanzen

Der zweite Schritt beschreibt die Speicherung der abgefragten Informationen in der programminternen Struktur. Dazu wird auf ein etabliertes Konzept vieler Programmiersprachen zurückgegriffen. Gemeint ist die sogenannte *objektorientierte Programmierung*, abgekürzt OOP. Sie dient dazu, reale Probleme in kleinere Objekte mit den jeweils spezifischen Eigenschaften und Funktionen zu zerlegen. Beispielsweise kann die *Wurzel* eines Baumes als Objekt abgebildet werden. Drei spezifische Eigenschaften der *Wurzel* könnten durch die Variablen *Typ*, *Größe* und *Alter* gespeichert werden. Innerhalb der Variable *Typ* dürften die Benutzer ausschließlich zwischen Pfahl-, Senker- oder Herzwurzelsystem wählen. Auf diesem Weg wird jedes Objekt über seine Eigenschaften definiert. Hinzu kommen die Funktionen, die Methoden genannt werden. Für das Beispiel *Wurzel* könnte es eine Methode geben, welche die Wasser- und Nährstoffaufnahme aus dem Boden abbildet. Um ein solches Objekt in technischer Hinsicht zu erzeugen, ist eine *Klasse* notwendig. Die Klasse ist im weitesten Sinne eine Art Vorlage (vgl. WOLF 2006, S. 259 ff.).

Neben der objektorientierten Programmierung wird bei grafischen Programmen häufig das *Model-View-Controller-Entwurfsmuster*, kurz MVC, verwendet. Es unterstützt den Softwareentwickler bei der Programmierung der internen Programmläufe. Das *Model* hat die Aufgabe, alle Daten, auch in Form von Objekten, zu speichern und zu verwalten. Die *View* dient der grafischen Darstellung dieser Informationen in Tabellen oder Grafiken. Über den *Controller* kommunizieren das *Model* und die *View* miteinander. Er bestimmt beispielsweise, wie die Daten durch die *View* aufzubereiten sind (vgl. WOLMERINGER 2007, S. 327 f.).

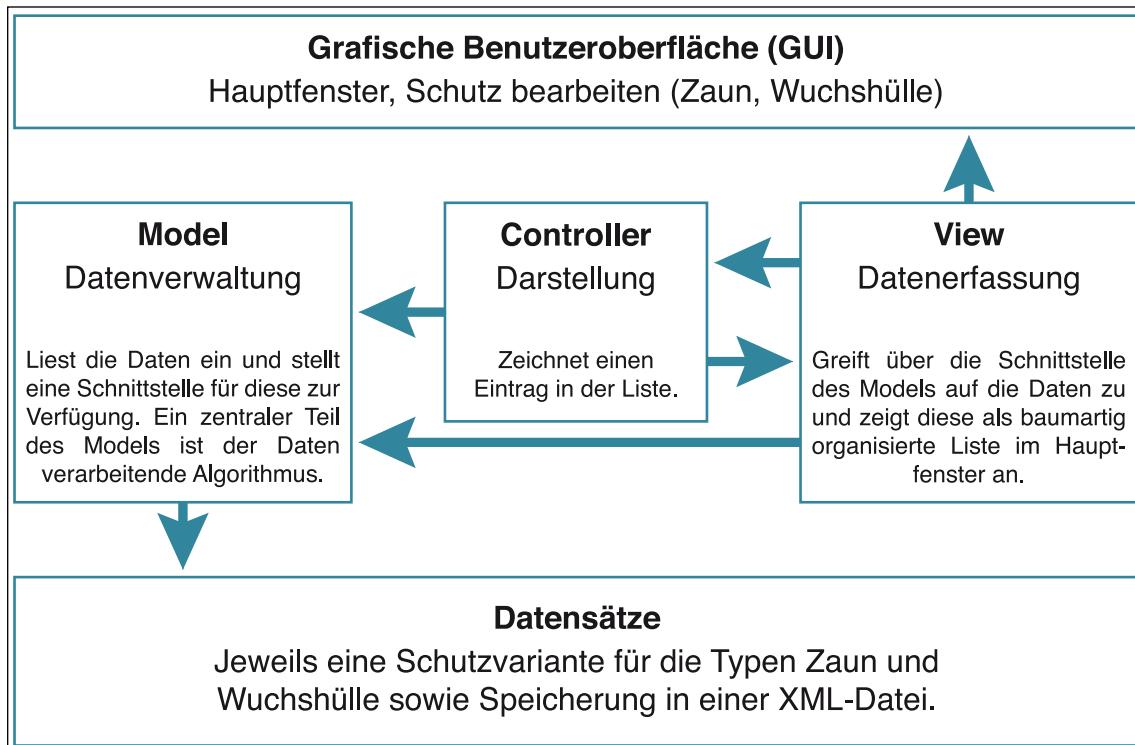


Abbildung 1: Model-View-Controller-Entwurfsmuster für den Wuchshüllenrechner.
Eigene Darstellung nach: ERNESTI; KAISER 2015, S. 872

Im nächsten Schritt ist zu überlegen, in welcher Art und Weise die gespeicherten Informationen weiterverarbeitet werden sollen. Oftmals wird hierfür ein Algorithmus entwickelt, der die Daten analysiert und verarbeitet. Im Fall des *Wuchshüllenrechners* soll der Algorithmus auf der Funktionsformel der Kostengleichheit basieren. Die Anwendung berechnet folglich beim Vergleich der Schutztypen Zaun und Wuchshülle den Steigungswert der Funktion. Das Ergebnis ist demnach ein einzelner Zahlenwert für jeden Vergleich, der ebenfalls durch das Model gespeichert werden muss.

Sobald endgültig sichergestellt ist, in welcher Form die Daten zu organisieren sind, kann die grafische Programmoberfläche entwickelt werden. In diesem Schritt wird die tatsächliche Schnittstelle zu den Anwendern geschaffen und die Benutzerabfragen werden grafisch aufbereitet. In jedem Fall benötigt eine solche Anwendung Fenster und Dialoge, welche die entsprechenden Eingabefelder enthalten. Für das Programm *Wuchshüllenrechner* sollen folgende Elemente erstellt werden:

- **Hauptfenster „Wuchshüllenrechner“:**

Das Hauptfenster beinhaltet alle zentralen Elemente. Es werden die Projektdaten erfasst und visuell mittels Liste und Diagramm dargestellt.

- **Dialog „Nützliche Hinweise“:**

Es wird auf die Berücksichtigung der Mehrwertsteuer sowie der Lohnnebenkosten hingewiesen.

- **Dialog „Schutz bearbeiten“:**

Je nach gewähltem Schutztyp wird eine unterschiedliche Datenerfassungsmaske für einen Zaun oder eine Wuchshülle angezeigt. Hier werden alle notwendigen Informationen von den Benutzern mithilfe von Eingabefeldern abgefragt. Zusätzlich müssen die Anwender eine Schutzbeschreibung eingeben, damit der Schutztyp später identifiziert werden kann.

- **Dialog „Über Wuchshüllenrechner“:**

Es werden Hinweise zur Programmversion und den Autoren bereitgestellt.

- **Dialog „Fachbeiträge“:**

Es werden die grundlegenden Fachbeiträge zur Thematik in einer Auswahl Liste zum Lesen angeboten.

Bei allen Benutzereingaben muss stets auf Plausibilität getestet werden. Deshalb ist oftmals eine große Menge Quelltext notwendig, um nur wenige Daten von den Anwendern abzufragen. Zusätzlich kann es hilfreich sein, den Anwendern bestimmte Schritte vorzugeben, indem manche Bereiche ausgegraut dargestellt werden. Ferner helfen kleine Hinweistexte in den Fenstern und Dialogen beim Verstehen der Programmoberfläche. Zusammengefasst müssen einige Aspekte der Benutzerführung durch den Softwareentwickler beachtet werden, wenn es um das Design der grafischen Oberfläche einer Anwendung geht. Damit zeigt sich wieder das Spannungsfeld, das zwischen den Datensatzstrukturen und der jeweiligen Darstellung besteht.

Sollten sich die Benutzer dennoch in der Applikation nicht zurechtfinden, bekommen sie üblicherweise Hilfe in der programmeigenen Dokumentation. Diese ist ebenfalls Teil der Benutzerführung und wird im letzten Schritt vor der Veröffentlichung erstellt. Sie sollte die Anwender problemlos durch das gesamte Programm führen und die jeweilige Funktion detailliert erklären. Auf Wunsch können die Anwender so die Programmschritte selbst in einer eigenen Kalkulation nachvollziehen.

2.3 Auswahl der Programmiersprache

In der Entscheidung, welche Programmiersprache für das Projekt *Wuchshüllenrechner* ausgewählt werden soll, werden viele technische Kriterien berücksichtigt. Dabei wurden einige dieser Kriterien bereits im Vorfeld der Überlegungen angedeutet. Beispielsweise soll die Computersprache *plattformunabhängig* sein und das Konzept der *objektorientierten Programmierung* unterstützen. Außerdem ist die Anbindung an ein häufig verwendetes Framework für grafische Benutzeroberflächen, in Englisch GUI für Graphical User Interface, hilfreich. Die folgende Entscheidungsmatrix beschreibt hierzu alle relevanten Auswahlkriterien:

Sprache Kriterium	Python 3	C	C++	Java	Visual Basic (.NET)
Plattformunabhängigkeit	uneingeschränkt	eingeschränkt	eingeschränkt	uneingeschränkt	nur Windows
objektorientierte Programmierung	unterstützt	nicht unterstützt	unterstützt	unterstützt	unterstützt
GUI Frameworks	TkInter, PyQt, wxPython	GTK+,	Qt, wxWidgets	AWT, Swing, SWT	Windows Forms (.NET)
Unterstützung für MVC	unterstützt	unterstützt	unterstützt	teilweise unterstützt	unterstützt
Übersetzung in andere Sprachen	unterstützt	unterstützt	unterstützt	unterstützt	unterstützt
Entwicklerfreundlichkeit	hoch	gering	gering	mittel	hoch
Geschwindigkeit	mittel	hoch	hoch	mittel	mittel

Tabelle 1: Entscheidungsmatrix für Programmiersprachen

Python 3 bietet eine breite Unterstützung der oben aufgeführten Auswahlkriterien und hat eine vergleichsweise einfache Syntax. Darüber hinaus können viele Probleme durch die bereits mitgelieferten Standardbibliotheken gelöst werden. Daher muss die Sprache nur in Einzelfällen um weitere Funktionen ergänzt werden. Die Speicherverwaltung ist automatisiert, weshalb sich der Entwickler keinerlei Gedanken um die Belegung und Freigabe von Arbeitsspeicher machen muss (vgl. ERNESTI; KAISER 2015, S. 33 f.). Außerdem kann *Python 3* an das Framework *PyQt 5* für grafische Oberflächen angebunden werden. Deshalb ist gewährleistet, dass die Anwendung in verschiedene Sprachen übersetzt werden kann.

Zusammen waren diese Auswahlkriterien bestimmend dafür, dass der *Wuchshülsenrechner* in der Programmiersprache *Python 3* entwickelt wurde.

Python 3 beinhaltet viele Sprachelemente einer Skriptsprache. Dennoch handelt es sich um eine sogenannte interpretierte Programmiersprache. Der Unterschied zwischen einer Skript- und einer Programmiersprache ist demnach der Compiler, der aus dem menschenlesbaren Quelltext einen für den Computer verständlichen Byte-Code erzeugt. *Python 3* verwendet hierzu üblicherweise die Dateiendung *py* für den Quellcode sowie *pyc* für den jeweiligen Byte-Code. Dieser Byte-Code wird vom Interpreter der Sprache ausgeführt. Folglich dient der Interpreter als Abstraktionsebene zwischen dem Betriebssystem und der Programmiersprache. Der Interpreter muss also speziell für das jeweilige Betriebssystem konzipiert werden, sodass die Programmiersprache plattformunabhängig verwendet werden kann (vgl. ERNESTI; KAISER 2015, S. 32 f.).

Damit eine Software in *Python 3* entwickelt werden kann, muss vorher eine Laufzeitumgebung mit dem entsprechenden Interpreter und den benötigten Bibliotheken für das jeweilige Betriebssystem eingerichtet werden. Die Einrichtung einer solchen Umgebung kann durchaus aufwendig sein und ist möglicherweise von Plattform zu Plattform verschieden. Welche Schritte für die Betriebssysteme Apple macOS und Microsoft Windows im Rahmen dieser Arbeit notwendig waren, kann dem Anhang entnommen werden.

Zusätzlich zur Laufzeitumgebung wird eine Entwicklungsumgebung benötigt. Sie dient dazu, den Quellcode zu schreiben und zu testen. Im einfachsten Fall besteht eine solche Entwicklungsumgebung aus einem Editor, der bestimmte Befehle der Programmiersprache farbig hervorhebt. Eine solche Funktion nennt sich Syntax-highlighting und hilft dem Programmierer wesentlich. Die Anwendung in Form des Quellcodes wird hierbei direkt durch Aufrufen des Interpreters durch den Entwickler ausgeführt. Oftmals kommt aber eine *Integrierte Entwicklungsumgebung*, englisch IDE für *Integrated Development Environment*, zum Einsatz. Sie bringt nicht nur einen Editor für die Programmiersprache mit, sondern auch einige Werkzeuge, die das Entwickeln einer Software enorm erleichtern. Der Autor dieser Arbeit bevorzugt jedoch im Allgemeinen den zuvor beschriebenen einfachen Editor.

Am Ende ist bei der Softwareentwicklung mit *Python 3* noch ein weiterer Arbeitsschritt notwendig, der bei Computersprachen wie *C* und *C++* entfällt. Bei der Verwendung solcher Sprachen erstellt der Compiler einen Byte-Code, der direkt vom Betriebssystem ausgeführt werden kann. Die Abstraktionsebene durch den Interpreter entfällt folglich. Damit eine Anwendung, die in *Python 3* geschrieben wurde, auch auf anderen Systemen funktioniert, ohne auf komplizierte Art und Weise eine Laufzeitumgebung einrichten zu müssen, sollte der Byte-Code zusammen mit dem Interpreter verpackt werden. Damit ist sichergestellt, dass die Applikation bei jedem Anwender problemlos läuft. Das Ergebnis ist das fertige Programm.

2.4 Festlegung der Softwarelizenz

Sobald eine Software entwickelt wird und die Entscheidung feststeht, dass die Anwendung anschließend veröffentlicht werden soll, wird eine Vereinbarung über die Lizenz getroffen. Bei dem Programm *Wuchshüllenrechner* wurde vom Autor dieser Arbeit festgelegt, dass es sich um eine *Open-Source-Software* handeln soll. *Open Source* bedeutet übersetzt, dass der Quellcode der Anwendung komplett offengelegt und frei zugänglich ist. Jeder kann das Programm unter Beachtung dieser Prämisse nach seiner Vorstellung weiterentwickeln. Rechtlich ist der Open-Source-Status mit der Lizenz GNU GENERAL PUBLIC LICENSE in der Version 3 vom 29. Juni 2007, abgekürzt GPLv3, festgehalten. Eine vollständige Kopie der GPLv3 liegt dem Programm *Wuchshüllenrechner* bei. Zudem wurde die Kurzform der Lizenz am Beginn jeder einzelnen Quelldatei eingefügt:

```
01 This file is part of Wuchshüllenrechner.
02
03 Wuchshüllenrechner is free software: you can redistribute it and/or modify
04 it under the terms of the GNU General Public License as published by
05 the Free Software Foundation, either version 3 of the License, or
06 (at your option) any later version.
07
08 Wuchshüllenrechner is distributed in the hope that it will be useful,
09 but WITHOUT ANY WARRANTY; without even the implied warranty of
10 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11 GNU General Public License for more details.
12
13 You should have received a copy of the GNU General Public License
14 along with this program. If not, see <http://www.gnu.org/licenses/>.
```

Listing 1: Lizenzhinweis am Beginn jeder Quelldatei

Zusätzlich zur Lizenz gilt für das Softwareprodukt ein Urheberrecht, welches nicht übertragen werden kann. Deshalb wurde jede einzelne Quelldatei zusätzlich mit folgendem Urheberrechtshinweis ausgestattet:

```
01 Copyright (C) 2016 Tobias Helfenstein <tobias.helfenstein@mailbox.org>
02 Copyright (C) 2016 Anton Hammer <hammer.anton@gmail.com>
03 Copyright (C) 2016 Sebastian Hein <hein@hs-rottenburg.de>
04 Copyright (C) 2016 University of Applied Forest Sciences Rottenburg
05      <hfr@hs-rottenburg.de>
```

Listing 2: Urheberrechtshinweis ebenfalls am Beginn jeder Quelldatei

3 Methodisches Vorgehen

3.1 Literaturanalyse

Die Literaturanalyse stellt ein Arbeitsmittel der Inhaltsanalyse dar und hat das Ziel, den aktuellen Stand der Forschung zu erfassen. Sie knüpft somit an grundlegendes Wissen an und manifestiert den Forschungsbedarf im ausgewählten Themengebiet sowie die Fragestellung der wissenschaftlichen Arbeit.

Grundsätzlich sind bei einer Inhaltsanalyse und folglich bei der Literaturanalyse die sogenannten Kommunikationsinhalte wie Texte, Bilder und Filme Gegenstandsbereich der Untersuchung (vgl. ATTESLANDER 2010, S. 195 f.). Das heißt, dass bereits vorhandene Theorien, Arbeitsergebnisse und Inhalte in Form von verfassten Werken analysiert werden. THEISEN unterscheidet die Literaturanalyse weiter in einen systematischen und pragmatischen Ansatz. Der systematische Weg einer Literaturrecherche beruht demnach auf den gedruckten und elektronischen Fundstellen wie Nachschlagewerken, Bibliothekskatalogen oder Periodika. Beim pragmatischen Vorgehen wird auf Suchmaschinen, Lehrbücher oder Literaturverzeichnisse in themenspezifischer Literatur zurückgegriffen (vgl. THEISEN 2013, S. 60). Als eine der wichtigsten Quellen schätzt THEISEN die Fachzeitschriften ein, die mit ihren aktuellen wissenschaftlichen Diskussionen und Literaturhinweisen viel Potenzial bieten (vgl. THEISEN 2013, S. 73 f.). Unabhängig vom fachlichen Teil der Abschlussarbeit wurden daher insbesondere zur Programmierung der Software folgende Dokumentationen herangezogen:

- Dokumentation zu Python 3 (gedruckt und online)
- Dokumentation zu Qt 5.5 und PyQt 5.5 (online)
- Dokumentation zu PyQtGraph (online)
- Dokumentation zu cx_Freeze (online)
- Quelltextbeispiele zu PyQt 5.5 (online)
- Forenbeiträge über Stack Overflow (online)

3.2 Entwicklung einer Software

Die tatsächliche Programmierung ist das Herzstück der Softwareentwicklung und dient der Realisierung einer Anwendung. Dennoch sind die Überlegungen im Vorfeld ein wichtiger Bestandteil, da sie sozusagen als Leitfaden für den Entwickler dienen. Das Programmieren erfordert im Allgemeinen viel Übung und Kreativität, sodass auch komplexe Bezüge schnell umgesetzt werden können. Dabei ist es durchaus angebracht, auf ein Handbuch oder eine Dokumentation der jeweiligen Programmiersprache zurückzugreifen.

3.2.1 Design des Programmablaufes

Die Programmierung beginnt mit der groben Abbildung des Programmablaufes. Bei Projekten, die in einer objektorientierten Sprache umgesetzt werden sollen, wird dazu oftmals die *Unified Modeling Language*, abgekürzt UML, eingesetzt. Mithilfe dieser Sprache können die einzelnen Klassen mit ihren Eigenschaften und Methoden rudimentär modelliert werden. Dabei ist es problemlos möglich, mehrere solcher Klassen untereinander zu verknüpfen, um ein sogenanntes Klassendiagramm zu erstellen (vgl. WOLMERINGER 2007, S. 204 f.). Die Abbildung 2 zeigt ein Beispiel eines Klassendiagrammes für einen Teilbereich dieser Arbeit.

Im dargestellten Klassendiagramm werden die vier Klassen *Plant*, *Fence*, *Tube* und *VariantItem* gezeigt. Sie bilden zusammen die Speicherstruktur für die Eingabedaten. Objekte, die beispielsweise von der Klasse *Plant* abgeleitet werden, sollen demnach alle Informationen zur Pflanze und zur Pflanzung enthalten. Ähnlich wird mittels der beiden Klassen *Fence* und *Tube* die Schutzvariante festgelegt. Letztlich sollen alle Daten innerhalb eines *VariantItems* gespeichert werden.

Der Vorteil dieser Darstellung besteht darin, dass der Entwickler einen Programmablaufplan erhält. Wurde dieser Plan sorgfältig und vollständig für die gesamte Anwendung erstellt, muss sich der Programmierer während der Entwicklung erheblich weniger Gedanken um den Aufbau der Klassen sowie deren Abhängigkeiten machen. Folglich kann ein Programm wesentlich effizienter erstellt werden. Außerdem können Problemstellungen wie beispielsweise die Bereitstellung von Datenschnittstellen innerhalb der Anwendung optimal gelöst werden. Zudem ist

es möglich, ein Klassendiagramm in die jeweilige Programmiersprache zu übersetzen. Damit wird mithilfe des Programmablaufplanes bereits konkreter Quelltext erarbeitet. Die Verwendung der UML ist dann besonders hilfreich, wenn an einem Projekt mehrere Softwareentwickler mitarbeiten. Somit kann das Projekt in mehrere übersichtliche Teilbereiche gegliedert werden.

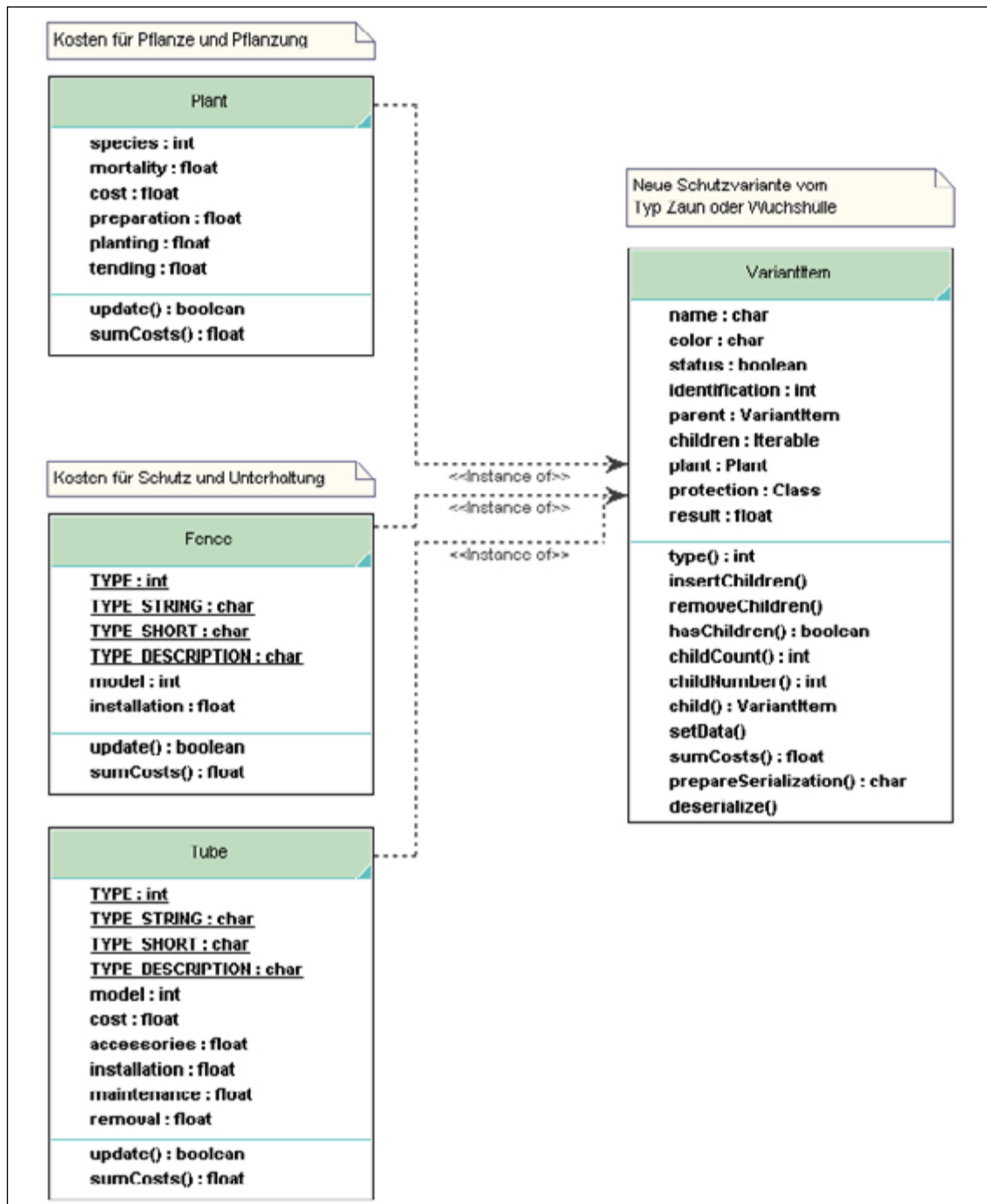


Abbildung 2: Klassendiagramm für *Plant*, *Fence*, *Tube* und *VariantItem*

Den genannten Vorteilen steht durchaus ein gewichtiger Nachteil gegenüber. Denn zur Modellierung des Klassendiagrammes muss viel Zeit investiert werden, um ein brauchbares Ergebnis zu erzielen. Daher musste aus Zeitgründen auf die Erarbeitung eines vollständigen Klassendiagrammes im Rahmen dieser Arbeit verzichtet werden. Als Ablaufschema dienten deshalb insbesondere die zuvor geschilderten Überlegungen zum Programmaufbau.

3.2.2 Nutzung von sprachentypischen Konventionen

Jede Computersprache verfügt über bestimmte sprachentypische Konventionen, die mit dem Dialekt einer menschlichen Sprache vergleichbar sind. Nach Möglichkeit sollte sich der Programmierer an die jeweiligen Konventionen halten, um den Quellcode lesbarer zu gestalten. So legt eine Konvention fest, welches Schema bei der Namensgebung von Variablen und Funktionen sowie Klassen und Modulen zu berücksichtigen ist (vgl. PYTHON DEVELOPER'S GUIDE 2016). Eine andere Konvention bestimmt, dass Funktionen einen einheitlichen Datentyp für Rückgabewerte haben müssen. Es ist nicht üblich, dass die gleiche Funktion an einer Stelle den Wahrheitswert *True* zurückgibt und an einem anderen Punkt den Textwert *Fehler* anstelle des Wahrheitswertes *False* meldet.

Im Allgemeinen beschreibt die jeweilige Computersprache häufig eine Kurzvariante, um bestimmte Abläufe technisch effizienter zu gestalten. Hierzu zählt die Kurzform für bedingte Ausdrücke.

```
01 if x == 1:  
02     var = 15  
03 else:  
04     var = 30
```

Listing 3: Langform der bedingten Anweisung (if)

```
01 var = (15 if x == 1 else 30)
```

Listing 4: Kurzform der bedingten Anweisung (if)

In Listing 3 ist die Langform eines bedingten Ausdrucks dargestellt. Zuerst wird geprüft, ob der Wert der Variablen *x* der Ganzzahl *1* entspricht. Falls das Ergebnis dieser Prüfung *wahr* ist, wird der Variablen *var* die Ganzzahl *15* zugewiesen. Ansonsten erhält die Variable *var* den Wert *30*. Die Kurzform dieser Bedingungsprüfung ist in Listing 4 abgebildet und in dieser Art typisch für die Programmier-

sprache *Python 3* (vgl. ERNESTI; KAISER 2015, S. 68). Solche Kurzformen können die Geschwindigkeit eines Programmes erhöhen, weil sie in der Regel intern optimiert wurden. Hierbei wird dann vom *idiomatic way* oder *pythonic way* gesprochen, weil die Befehlsabfolge sprachentypisch ist.

3.2.3 Festlegung einer Dateisystemstruktur

Unabhängig von den Konventionen sollte grundsätzlich für jedes Projekt eine Dateisystemstruktur festgelegt werden. Diese Festlegung gilt insbesondere für den erstellten Quellcode der Anwendung. In *Python 3* startet das Hauptprogramm aus der Quelldatei *main.py*. Sie lädt nach dem Start alle notwendigen Module, Klassen und Funktionen. In der Praxis gliedert sich der Programmordner häufig in Unterverzeichnisse, die verschiedene Programminhalte bündeln. So umfasst beispielsweise das Verzeichnis *gui* alle Module, welche für die Bereitstellung der grafischen Oberfläche notwendig sind. Das ist sinnvoll, weil der Entwickler somit einen Überblick über das Projekt behält. Selten befinden sich Programme ausschließlich in einer Datei. Das ist nur bei kleineren Anwendungen und Skripts gebräuchlich.

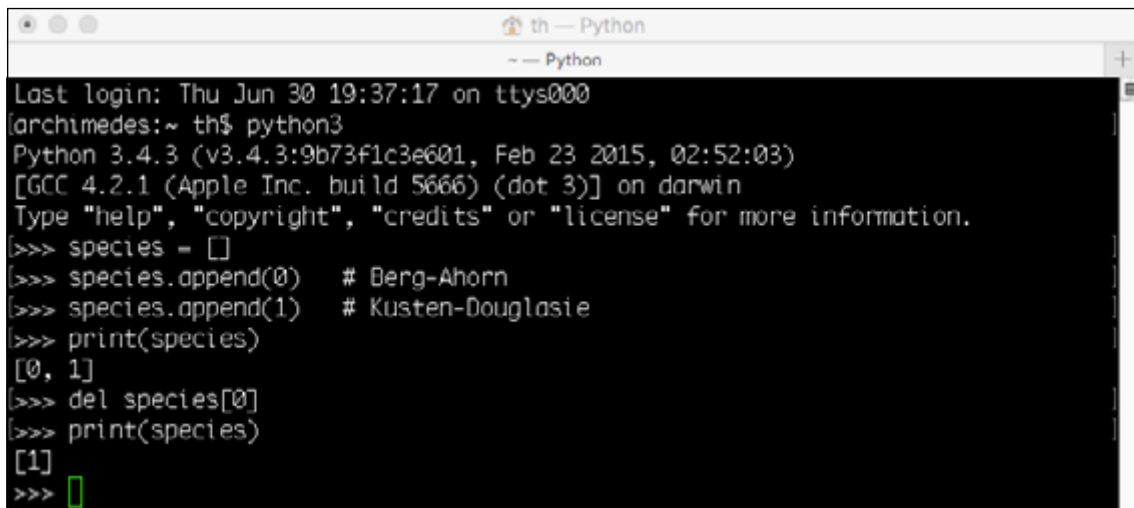
Abweichend von dieser Struktur stellt sich der Aufbau des plattformabhängigen Programmordners dar. Er beinhaltet die tatsächlich ausführbare Datei für das entsprechende Betriebssystem und wird mithilfe des Werkzeuges *cx_Freeze* erstellt. So wird für Microsoft Windows die Datei *wuchshüllenrechner.exe* mit ihren Abhängigkeiten erzeugt. Hierbei steuert *cx_Freeze* die Dateisystemstruktur, die deshalb vom Entwickler nur teilweise beeinflusst werden kann.

3.2.4 Probelauf mit einzelnen Befehlen

Oftmals fehlt bei der Verwendung einzelner Befehle die Kenntnis über die richtige Syntax oder den genauen Rückgabewert. Die Syntax beschreibt grundsätzlich, wie ein Befehl zu schreiben und zu verwenden ist. Mit dem Rückgabewert ist das Ergebnis des ausgeführten Befehles gemeint.

Daher ist es häufig notwendig, kleinere Befehlspassagen zu testen, bevor sie in der eigentlichen Anwendung eingesetzt werden. Hierzu wird regelmäßig die sogenannte Befehlszeile des Interpreters benutzt. Diese führt die Befehle direkt

nach der Eingabe aus und der Entwickler kann das Verhalten des Codes intensiv überprüfen. Erst danach wird die Befehlspassage in die Anwendung eingefügt. Es ist im Übrigen wesentlich aufwendiger, einen solchen Probelauf innerhalb der gesamten Anwendung durchzuführen, als in kleinen Abschnitten zu testen.



```
th — Python
~ — Python
Last login: Thu Jun 30 19:37:17 on ttys000
archimedes:~ th$ python3
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 23 2015, 02:52:03)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> species = []
>>> species.append(0) # Berg-Ahorn
>>> species.append(1) # Kusten-Douglasie
>>> print(species)
[0, 1]
>>> del species[0]
>>> print(species)
[1]
>>> 
```

Abbildung 3: Probelauf mit einer Liste für Baumarten

Manchmal kommt es dennoch vor, dass während der Laufzeit der Anwendung nicht sicher feststeht, welchen Inhalt eine Variable hat. Mithilfe eines solchen Probelaufes kann sich der Entwickler Schritt für Schritt im Befehlsablauf fortbewegen und so den Wert der Variablen ermitteln.

3.2.5 Programmierung

Der Quelltext einer Applikation besteht im Wesentlichen aus den jeweiligen Befehlen der Computersprache. Diese bauen meistens aufeinander auf und werden Zeile für Zeile in die Quelldatei geschrieben. Nach diesem Schema werden die Kommandos auch vom Compiler in Byte-Code umgewandelt und vom Interpreter gelesen. Werden etwa bedingte Ausdrücke verwendet, kann eine Zeile schnell sehr komplex werden. Es ist daher ratsam, mehrere Ausdrücke, die sich gegenseitig logisch bedingen, auf einem Papier zu gruppieren. Dieser Schritt unterstützt den Entwickler enorm und hilft dabei, unnötige Bedingungen zu eliminieren. Auch bei anderen komplexen Abläufen ist es sinnvoll, zuerst Papier und Stift als Hilfsmittel zu verwenden. Der jeweils benötigte Befehl ergibt sich dann aus den logischen Zusammenhängen oder findet sich in der Dokumentation der Computersprache.


```
01 import sys
02 from PyQt5.QtWidgets import QApplication
03 from gui.window_main import MainWindow
04
05
06 def main():
07     app = QApplication(sys.argv)
08
09     window = MainWindow()
10     window.show()
11
12     sys.exit(app.exec())
13
14 if __name__ == '__main__':
15     main()
```

Listing 5: Inhalt der Quelldatei *main.py* zum Starten des Wuchshüllenrechners

Bildet ein Kommando der Standardbibliotheken das gewünschte Verhalten nur teilweise ab, ist es durchaus möglich, dieses Kommando zu modifizieren. Oftmals werden hierfür eigene Funktionen erstellt, oder vorhandene Klassen funktional erweitert. Im Wesentlichen werden dazu Schleifen, bedingte Ausdrücke und bestimmte Grundbefehle für Ganzzahlen (Integer), Gleitkommazahlen (Float), Textwerte (String) und andere Datentypen in unterschiedlicher Kombination verwendet. Das Ergebnis ist der Quellcode der Anwendung.

Unterstützung und Übung erhält ein Softwareentwickler, indem er sich verschiedene Beispiele und Problemstellungen anschaut. Besonders im Internet lassen sich die unterschiedlichsten Lösungen für verschiedene Fragestellungen finden. Daraus kann immer ein Impuls für eigene Lösungsansätze gewonnen werden.

3.2.6 Entwicklung der Diagrammansicht

Zur Programmierung von Software gehört auch die Gestaltung von grafischen Oberflächen. Anhand der Diagrammansicht im Hauptfenster des *Wuchshüllenrechners* werden die Methodiken bei der Entwicklung einer solchen Oberfläche erklärt.

Die Entwicklung beginnt mit einem groben Entwurf der Oberfläche. Es ist sinnvoll, den Entwurf auf einem Papier als Bleistiftskizze entstehen zu lassen, weil er so einfacher verändert werden kann. Nach und nach werden dann in der Skizze die benötigten Steuerelemente für die Eingabefelder und das Diagramm einge-

zeichnet. Ein Beispiel einer solchen Skizze zur Beschreibung der Diagramm- und Ergebnisansicht ist nachfolgend in der Abbildung 4 gezeigt.

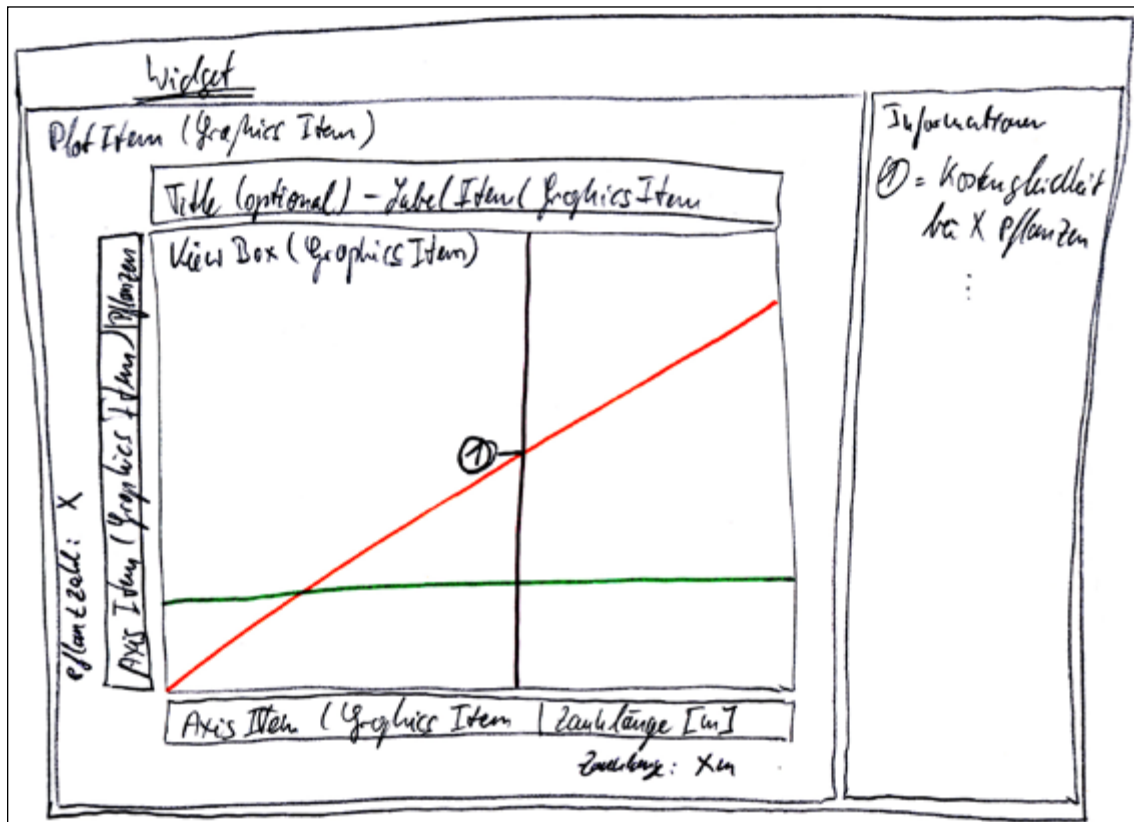


Abbildung 4: Skizze der Diagramm- und Ergebnisansicht

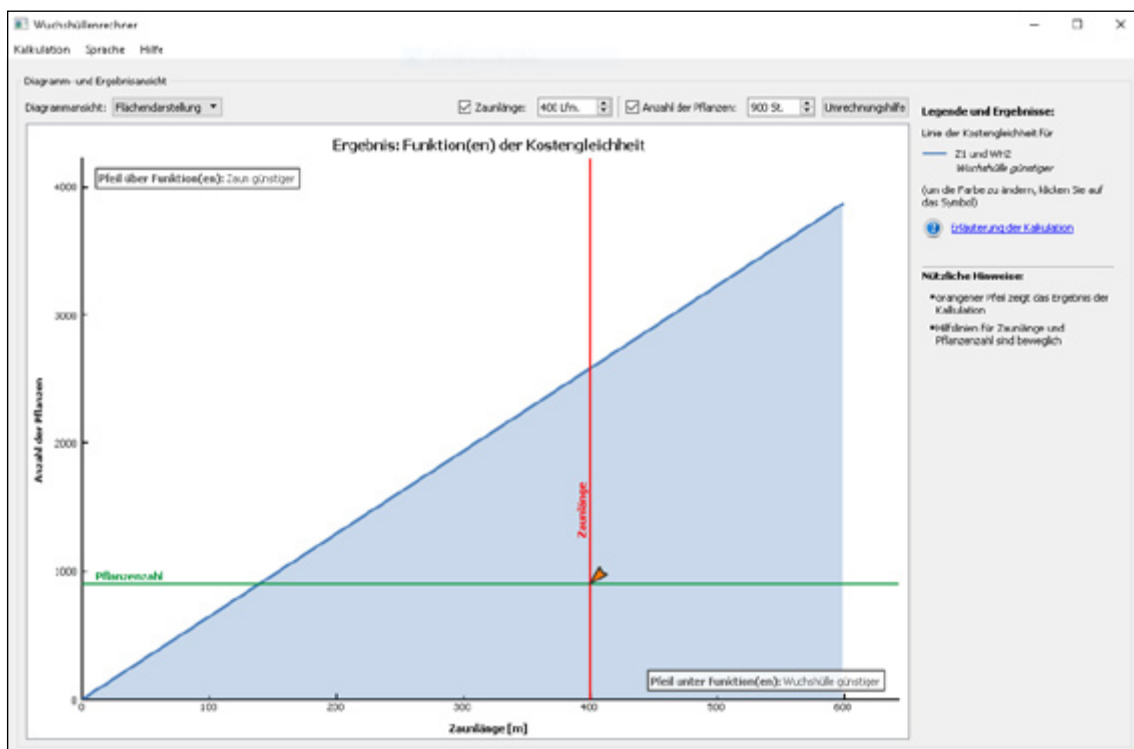


Abbildung 5: Die anhand der Skizze realisierte Diagramm- und Ergebnisansicht

Hier wurden zusätzlich die Elemente mit den jeweiligen Klassennamen beschriftet, damit beim Programmieren die Übersicht gewährleistet bleibt. Im Allgemeinen hilft eine solche Skizze dabei, sich die Programmoberfläche besser vorstellen zu können.

Im nächsten Schritt wird der Entwurf als Programmoberfläche am Computer umgesetzt, dargestellt in Abbildung 5. Die meisten Computersprachen und deren graphische Frameworks bringen hierzu entsprechende Werkzeuge mit, um dem Entwickler den Programmier- und Layoutaufwand zu ersparen. Bei der Verwendung von *PyQt 5* ist das Programm *Qt Designer* das Instrument der Wahl. Damit lassen sich bequem alle Steuerelemente und Schaltknöpfe einzeichnen und als Datei abspeichern. Diese Datei wird dann im Hauptprogramm an der entsprechenden Stelle automatisiert geladen und dargestellt.

Ein weiterer Weg zur Realisierung der Programmoberfläche ist mittels Programmierung möglich. Hierbei können die Elemente nicht per Mauszeiger eingezeichnet werden, sondern sie müssen im Quellcode penibel beschrieben werden. Der Effekt ist ein wesentlich höherer Aufwand. Dennoch ist der erzeugte Code deutlich schlanker als die automatisiert erstellte Datei.

Auf beiden Wegen wird somit die Programmoberfläche erzeugt, die beim Ausführen jedoch nur die Elemente darstellt. Aktionen in Form von Klicks sind dann noch nicht möglich. Deshalb müssen für die Oberfläche noch Aktionen definiert werden. Dies geschieht im Quellcode und ist entsprechend aufwendig. Beispielsweise wurden für die Diagrammansicht verschiedene Aktionen erstellt, die beim Bewegen der Linien die *Zaunlänge* und die *Anzahl der Pflanzen* permanent aktualisiert.

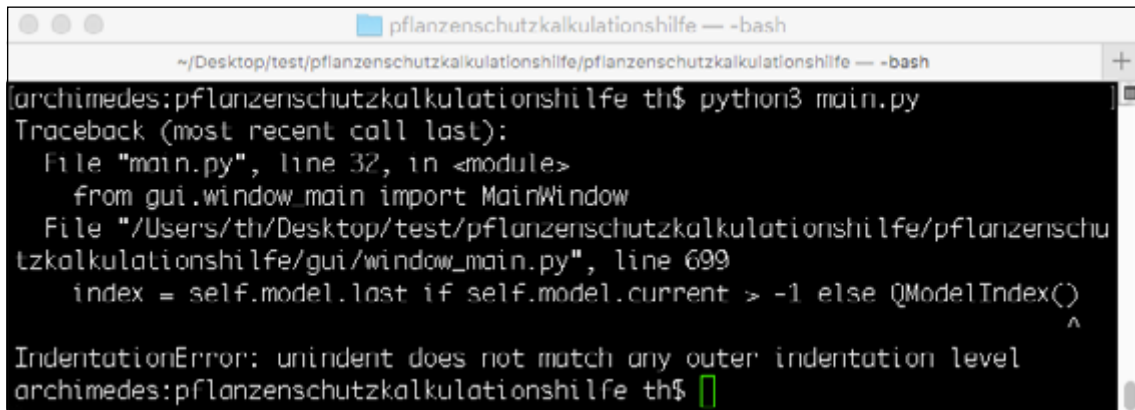
Im Ergebnis wird ein sogenanntes *Widget* erzeugt, das problemlos in anderen Programmoberflächen verwendet werden kann.

3.2.7 Evaluation durch den Entwickler

Jeder Programmabschnitt muss durch den Entwickler ausgiebig getestet werden. Das heißt im Detail, dass die Anwendung gestartet und die neue Funktion überprüft wird. Deshalb kann der Test je nach Funktionalität entsprechend aufwendig

sein. Insbesondere wird solange getestet, bis das gewünschte Verhalten sichergestellt ist und alle Fehler beseitigt sind.

Im einfachsten Fall wird eine Befehlspassage direkt implementiert und kann ohne Umweg ausgeführt werden. Dabei führen Syntaxfehler oder einfache Tippfehler zum Programmabbruch. Oftmals hilft dann der Interpreter mit einer aussagekräftigen Fehlermeldung, sodass die Korrektur schnell vorgenommen werden kann.



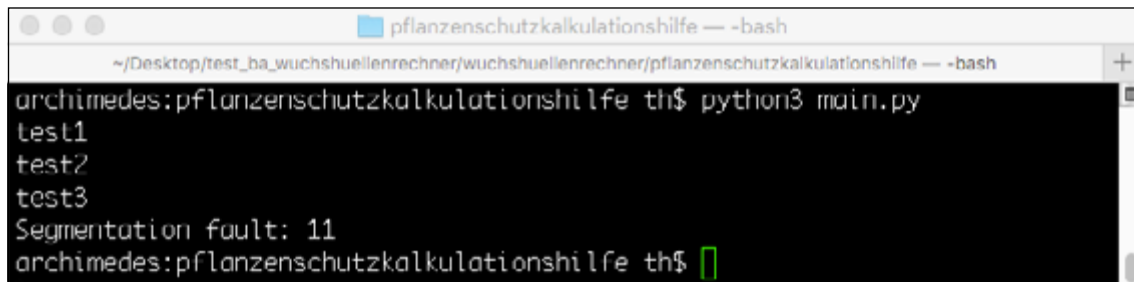
```
pflanzenschutzkalkulationshilfe -- -bash
~/Desktop/test/pflanzenschutzkalkulationshilfe/pflanzenschutzkalkulationshilfe -- -bash
archimedes:pflanzenschutzkalkulationshilfe th$ python3 main.py
Traceback (most recent call last):
  File "main.py", line 32, in <module>
    from gui.window_main import MainWindow
  File "/Users/th/Desktop/test/pflanzenschutzkalkulationshilfe/pflanzenschu
tzkalkulationshilfe/gui/window_main.py", line 699
    index = self.model.last if self.model.current > -1 else QModelIndex()
                                ^
IndentationError: unindent does not match any outer indentation level
archimedes:pflanzenschutzkalkulationshilfe th$
```

Abbildung 6: Syntaxfehler im Quelltext

Die Abbildung 6 zeigt einen solchen Syntaxfehler in der Zeile 699 der Datei *window_main.py*. Bei näherer Betrachtung des Quelltextes an der angegebenen Fehlerstelle ist verständlich, warum es zum Programmabbruch gekommen ist. Denn der Interpreter erklärt, dass diese Zeile falsch eingerückt wurde und so nicht erkannt wird.

Ein komplexes Fehlverhalten eines Programmes tritt häufig auf, sobald mehrere Funktionen voneinander abhängig sind. Zugegebenermaßen weist auch hier der Interpreter auf den vorhandenen Fehler hin, aber meistens kann dann nicht mehr anhand der Meldung auf die Problemursache geschlossen werden. Um das Fehlverhalten eingrenzen zu können, muss sich der Entwickler Schritt für Schritt an die Ursache herantasten. Zum einen kann es hilfreich sein, die jeweilige Befehlspassage gesondert im Interpreter zu testen (s. Kapitel 3.2.4). Ein anderer Weg ist mithilfe der Ausgabeanweisung `print()` möglich. Dazu wird an einigen Stellen rund um die vermutete Ursache beispielsweise fortlaufend zählend `print("test1")` im Quellcode eingefügt. Beim Ausführen der Anwendung wird vor dem Programmabsturz die jeweilige Ausgabe von `print()` mit zum Beispiel "test3" angezeigt. Im Quelltext bedeutet dies, dass alle Zeilen, die nach der Ausgabe von

"test3" stehen, das Fehlerverhalten verursachen können. Der Entwickler kann somit relativ genau den Fehler im Quelltext eingrenzen. Bei der Verwendung einer *Integrierten Entwicklungsumgebung* stehen anstelle dieser Methoden einige Werkzeuge zur Verfügung, die das Debugging, also die Fehlerbeseitigung, erheblich erleichtern.

A screenshot of a terminal window titled 'pflanzenschutzkalkulationshilfe — -bash'. The terminal shows the execution of a Python script 'main.py' which prints 'test1', 'test2', and 'test3'. After 'test3', a 'Segmentation fault: 11' occurs. The prompt returns to 'archimedes:pflanzenschutzkalkulationshilfe th\$'.

```
archimedes:pflanzenschutzkalkulationshilfe th$ python3 main.py
test1
test2
test3
Segmentation fault: 11
archimedes:pflanzenschutzkalkulationshilfe th$
```

Abbildung 7: Beispiel der Ausgabeanweisung `print()` im Umfeld eines Fehlers

Andere komplexe Fehler können auftreten, wenn das Programm ohne Absturz läuft, es aber dennoch zu einem Fehlverhalten kommt. Ist dann nicht sicher, welche Anweisungen im Quelltext zu diesem Verhalten führen, wird die Fehlersuche besonders zeitaufwändig. In gewissem Maße helfen hierbei ebenfalls die vorher beschriebenen Methodiken. Gerade, wenn in einem solchen Fall zuvor bestimmte Datensätze geladen werden müssen, wird die Fehlerbeseitigung sehr komplex.

Insgesamt ist es essenziell, dass der Entwickler seine Anwendung vor der Distribution und Weitergabe ausführlich testet. Nur so kann sichergestellt werden, dass die Anwendung fehlerfrei läuft. Viele Problemursachen können so bereits vor der Verteilung erkannt werden.

3.2.8 Evaluation durch einen ausgewählten Nutzerkreis

Neben den Testdurchläufen des Entwicklers ist auch die Evaluation durch einen ausgewählten Nutzerkreis besonders hilfreich. Dazu wurden bestimmte Personen mit unterschiedlichen Fachkenntnissen gebeten, die Software intensiv zu testen. Einige Tester haben gute Anwenderkenntnisse und Erfahrungen bezüglich Computeranwendungen, andere weisen eher betriebswirtschaftliche und waldbauliche Kenntnisse vor. Im Ergebnis konnte somit eine gute Rückmeldung zwischen der Benutzerfreundlichkeit und der fachlichen Programminhalte erreicht werden.

Es war vor allem das Ziel, die Verständnisfehler im Ablauf sowie innerhalb der Programmoberfläche aufzudecken. Hier konnten wertvolle Informationen gesammelt werden, wie der *Wuchshüllenrechner* einem größeren Nutzerkreis zugänglich gemacht werden kann. Außerdem benötigte der Entwickler weitere Sichtweisen, um die Verständnisproblematik lösen zu können. Darüber hinaus fanden die Tester häufig weitere Logikfehler im Programm, die vom Entwickler gar nicht betrachtet werden konnten, weil er voreingenommen ist. Zusätzlich konnten mithilfe der Tester durchaus neue Ideen und Gedanken gewonnen werden, um die Software zu einem späteren Zeitpunkt zu erweitern. Dies fällt unter den Punkt *Ausblick*.

Durchgeführt wurde die Evaluation, indem alle Tester einen gleichen und aussagekräftigen Entwicklungsstand des Programmes erhielten. Damit kann ein Bezug zu einem bestimmten Stand der Softwareentwicklung hergestellt werden. Dabei stand zentral auf einem Webserver die jeweilige Version des *Wuchshüllenrechners* (93b0eab) als komprimierter Dateiordner in Form des ZIP-Formates zum Herunterladen bereit. Zum Vergleich mit dem abschließenden Stand befindet sich die genannte Version ebenfalls auf der CD-ROM. Zusätzlich erhielt der ausgewählte Nutzerkreis einige Testdaten, um damit den Funktionsumfang der Anwendung zu überprüfen. Selbstverständlich sollten die Anwender das Programm auch mit eigenen Daten füttern. Der gesamte Testlauf war zeitlich auf etwa zwei Wochen beschränkt, sodass die Rückmeldungen zur Weiterentwicklung berücksichtigt werden konnten. Am 31. Mai 2016 war damit die offizielle Evaluation des *Wuchshüllenrechners* beendet und es wurden insgesamt 40 Testpersonen angeschrieben. Welche Informationen die Anwender erhielten und wie der Testdatensatz aussah, kann dem Anhang entnommen werden.

In dieser Hinsicht war die Evaluation durch einen ausgewählten Nutzerkreis von besonderer Bedeutung, um die Qualität des *Wuchshüllenrechners* zu sichern. Erst so wurde sichergestellt, dass die Anwendung in mehreren Betriebssystemumgebungen stabil läuft und damit ihren Einsatzzweck erfüllt.

4 Ergebnisse

4.1 Funktionsweise des Algorithmus

Das Herzstück des *Wuchshüllenrechners* ist der Algorithmus, welcher auf der Kalkulation von HAMMER basiert. Um damit die Berechnung durchführen zu können, waren verschiedene Vorüberlegungen notwendig. Speziell die Aufbereitung der Eingabedaten ist hierbei besonders bedeutsam. Am Beispiel der Klasse *Plant* wird deutlich, in welcher Speicherstruktur die Eingabedaten aus Kapitel 2.1 programmintern abgebildet werden.

```

01 class Plant(object):
02     def __init__(self, species=0, mortality=0.0, cost=0.0, preparation=0.0,
03         planting=0.0, tending=0.0):
04
05         # initializes plant specific values as integer
06         self.species = species
07
08         # initializes plant specific values as float
09         self.mortality = mortality
10
11         # initializes costs specific values as float
12         self.cost = cost
13         self.preparation = preparation
14         self.planting = planting
15         self.tending = tending
16
17     def update(self, dictionary):
18         # this method is less efficient then updating the class dictionary
19         # with self.__dict__.update(dictionary), but it's more secure
20         for key, value in dictionary.items():
21             try:
22                 self.__dict__[key] = value
23             except KeyError:
24                 return False
25
26         return True
27
28     def sumCosts(self):
29         return self.cost + self.preparation + self.planting + self.tending

```

Listing 6: Die Klasse *Plant* im Quelltext

Die Intention ist es, dass die Klasse *Plant* alle relevanten Informationen zur Baumart beinhaltet und zur Verfügung stellt. Deshalb wurden ab Zeile 5 bis Zeile 15 sechs Variablen definiert. Erstens ist das die Metainformation `self.species`, welche die Angabe zur Baumart enthält. Zweitens werden mit `self.cost`, `self.preparation`, `self.planting` und `self.tending` alle Werte für die Pflanzen-, Pflanz- sowie Kulturvorbereitungs- und -sicherungskosten als Gleitkommazahl mit einfacher Genauigkeit gesichert. In Zeile 9 wird zusätzlich die Mortalität mit `self.mortality` gespeichert. Diese Aktionen finden innerhalb des sogenannten *Konstruktors* statt, der in der Zeile 2 mit der Anweisung `def __init__()` beginnt. Auf alle Variablen des Objektes kann bei Bedarf jederzeit zugegriffen werden.

Ein wichtiger Bestandteil der Klasse *Plant* ist die Methode `sumCosts()` in Zeile 28. Sie summiert alle zugehörigen Kosten der jeweiligen Pflanze in Zeile 29. Im weiteren Programmablauf ist diese Methode deshalb besonders hilfreich. Insbesondere innerhalb des Algorithmus wird so der Gesamtkostenwert der Baumart auf einfache Weise bereitgestellt. Folglich hat die Methode einen Rückgabewert, der mithilfe der `return`-Anweisung eingeleitet wird.

In ähnlicher Art und Weise sind die beiden Klassen *Fence* und *Tube* aufgebaut. Analog zur Klasse *Plant* speichern sie die Kostenwerte und Metainformationen zum Zaun- und Wuchshüllenobjekt. Außerdem besitzen beide Klassen ebenfalls die Methode `sumCosts()`, um die entsprechenden Gesamtkosten bereitstellen zu können. Das hat den Vorteil, dass die Varianten, die sich sowohl aus einem Objekt *Pflanze* sowie aus dem Schutztyp *Zaun* oder *Wuchshülle* zusammensetzen, einheitlich behandelt werden können. Zur Kostenberechnung muss dann nicht mehr unterschieden werden, ob es sich um einen Zaun oder eine Wuchshülle handelt.

4.1.1 Sortierung nach Baumart

Die Angabe der Baumart hat eine besondere Bedeutung im Zuge der Kalkulation der Kostengleichheit zwischen Zaun und Wuchshülle. Denn der Rechenweg von HAMMER kann grundsätzlich unabhängig von der Baumart angewendet werden.

Demnach ist es rechnerisch möglich, eine Kultur bestehend aus Berg-Ahorn und geschützt durch Wuchshüllen mit einer Kultur bestehend aus Küsten-Douglasie und geschützt durch einen Zaun zu vergleichen. Das heißt, dass in der Kalkulation zwei bezüglich der Baumart unterschiedliche Kulturen verglichen werden können.

Ein solches Verhalten ist für den *Wuchshüllenrechner* nicht wünschenswert und für die Entscheidung nicht hilfreich. Deshalb wurde überlegt, wann es waldbaulich sinnvoll ist, die beiden Schutzvarianten miteinander zu vergleichen. Dabei fällt die Baumart als Unterscheidungskriterium ins Gewicht. Wird eine Kultur geschützt, soll die richtige Schutzvariante aufgrund der Kosten kalkuliert werden. Es macht keinen Sinn, wegen des Schutztyps die Baumart zu wechseln. Aus diesem Grund können für Zaun- und Wuchshüllenvarianten nur bei Eingabe der gleichen Baumart die Kosten gleichgesetzt werden. Folglich dient die Baumart als Sortierkriterium, wenn im *Wuchshüllenrechner* mehrere Zaun- und Wuchshüllenvarianten gleichzeitig verglichen werden sollen. Außerdem werden so keine unnötigen Berechnungen durchgeführt und die Anwender auf die fehlerhafte Eingabe hingewiesen.

Die Sortierung nach der Baumart führt zu dem Nebeneffekt, dass der gesamte Datensatz bestehend aus allen Varianten baumartig aufgebaut ist. Deshalb steht an der ersten Verzweigung immer eine Zaunvariante. Dieser werden alle Wuchshüllenvarianten mit der gleichen Baumart als Kindelemente zugeordnet. Es ist dann nicht mehr möglich, mehrere Zaunvarianten mit der gleichen Baumart anzulegen. Dennoch muss immer eine Variante vom Typ *Zaun* und vom Typ *Wuchshülle* angelegt werden, um überhaupt eine Berechnung durchführen zu können.

4.1.2 Kalkulation mittels Algorithmus

Die baumartig sortierten Varianten werden dann vom Algorithmus weiterverarbeitet. Seine Aufgabe ist es, die Steigung der Funktion der Kostengleichheit zwischen jeder Zaun- und Wuchshüllenvariante zu berechnen. In der Konsequenz basiert der Algorithmus auf dem grundlegenden Rechenweg von HAMMER. Das heißt, dass die folgende Berechnung durchgeführt und als Quellcode umgesetzt wird.

Allgemeine Kalkulationsformel nach dem Gleichsetzen:

$$y = \frac{A_5}{(B_1 + B_2 + B_3 + B_4 + B_5 + B_6) \cdot C - (A_1 + A_2 + A_3 + A_4)} \cdot x$$

Tatsächlich ist es vorerst ausreichend, nur den Steigungswert zu berechnen. Deshalb wird nur dieser Teil der homogen linearen Funktion betrachtet:

$$\frac{A_5}{(B_1 + B_2 + B_3 + B_4 + B_5 + B_6) \cdot C - (A_1 + A_2 + A_3 + A_4)}$$

Wie in Kapitel 2.1 beschrieben, entspricht die *Kostengruppe A* den Zaunkosten sowie die *Kostengruppe B* den Wuchshüllenkosten. Der Reduktionsfaktor *C* steht für die *Mortalitätsrate*. Im programminternen Ablauf stellt sich die Berechnung dann folgendermaßen dar:

```

01 def calculate(self):
02     shelters = []
03     for child in self.root.children:
04         if child.type == Fence.TYPE and child.hasChildren():
05             for shelter in child.children:
06                 # the following operations are based on the example calculation
07                 # of Dr. Anton Hammer <hammer.anton@gmail.com>
08                 slope = shelter.sumCosts() * (1 - shelter.plant.mortality) -
09                     child.sumCosts()
10                 result = child.protection.installation / slope
11
12                 # update result in model's item
13                 index = self.createIndex(shelter.childNumber(), 0, shelter)
14                 self.setData(index, result, TreeModel.ResultRole)
15             else:
16                 shelters.append(child.name)
17
18     # an empty list means that the calculation was successful
19     if not shelters:
20         self.calculated.emit()
21
22     return shelters

```

Listing 7: Der in Python 3 realisierte Algorithmus

Mit der Zeile `1 def calculate(self):` beginnt die Berechnungsmethode der Klasse `TreeModel`. Im Allgemeinen beinhaltet ein Objekt dieser Klasse alle Datensätze des *Wuchshüllenrechners*. Deshalb macht es Sinn, die Methode `calculate()` als Bestandteil von `TreeModel` zu entwickeln. Kennzeichnend für

ihre Funktion sind die beiden `for`-Anweisungen, die einen Schleifenaufruf ermöglichen. Der erste Schleifenaufruf in Zeile 3 wird solange ausgeführt, bis alle Zaunvarianten der Kalkulation durchlaufen sind. Bei jedem Aufruf wird temporär die aktuelle Zaunvariante in der Variablen `child` gespeichert. Sollte eine vermeintliche Zaunvariante vom Schutztyp Wuchshülle sein, wird der Name der Variante in die Liste `shelters` eingetragen. Die Benutzer erhalten später eine Rückmeldung, dass die jeweilige Variante nicht verglichen werden konnte. Im Gegensatz dazu bedeutet eine leere Liste, dass die Berechnung erfolgreich war.

Die zweite `for`-Schleife ist notwendig, um alle Kindelemente der Zaunvariante zu durchlaufen. Da der *Wuchshüllenrechner* an anderer Stelle die baumartige Organisation der Datensätze vornimmt, ist sichergestellt, dass es sich bei den Kindelementen ausschließlich um Varianten vom Typ *Wuchshülle* handelt. Folglich enthält die temporäre Variable `shelter` das aktuelle Wuchshüllenelement. Innerhalb dieser zweiten `for`-Anweisung wird dann erst die dargestellte Berechnung durchgeführt. Dazu wird in Zeile 8 der Wert des mathematischen Nenners (`slope`) ermittelt. Es werden alle Kosten der Wuchshülle (`shelter.sumCosts()`) mit der Mortalitätsrate (`1 - shelter.plant.mortality`) multipliziert. Vom Ergebnis wird der Wert der Gesamtkosten für die Pflanzung im Zaun (`child.sumCosts()`) abgezogen, wobei die Zaunkosten selbst vorerst unberücksichtigt bleiben. Um die entsprechenden Kostenwerte herzuleiten wird an dieser Stelle jeweils die Methode `sumCosts()` aufgerufen. In Zeile 10 wird schließlich die Steigung der Kostenvergleichsfunktion errechnet, indem die Zaunkosten (`child.protection.installation`) im mathematischen Zähler durch das zuvor erhaltene Ergebnis für den Nenner (`slope`) geteilt werden. Am Ende des Schleifendurchlaufes in Zeile 14 erfolgt die Speicherung der Steigung im jeweiligen Wuchshüllenelement.

Nachdem beide `for`-Anweisungen abgearbeitet sind, wird in Zeile 19 geprüft, ob die Liste `shelters` leer ist. In diesem Fall signalisiert die Methode `calculate()` in der folgenden Zeile mithilfe von `self.calculated.emit()`, dass die Berechnung tatsächlich durchgeführt wurde. Damit kann im nächsten Schritt das Diagramm gezeichnet werden. Zusätzlich wird der Inhalt von `shelters` mittels

`return` zurückgegeben. Ist die Liste nicht leer, enthält sie die Namen der fehlerhaften Varianten, die den Anwendern in einer Meldung angezeigt werden.

4.2 Die Programmoberfläche

Damit aus dem Algorithmus eine bequeme Entscheidungshilfe für jeden Anwender wird, ist eine grafische Programmoberfläche notwendig. Diese Oberfläche dient dazu, alle relevanten Informationen von den Benutzern abzufragen und die durch den Algorithmus erarbeiteten Ergebnisse ansprechend aufzubereiten. Daher war es eine essenzielle Aufgabe, die grafische Programmoberfläche des *Wuchshüllenrechners* zu entwickeln.

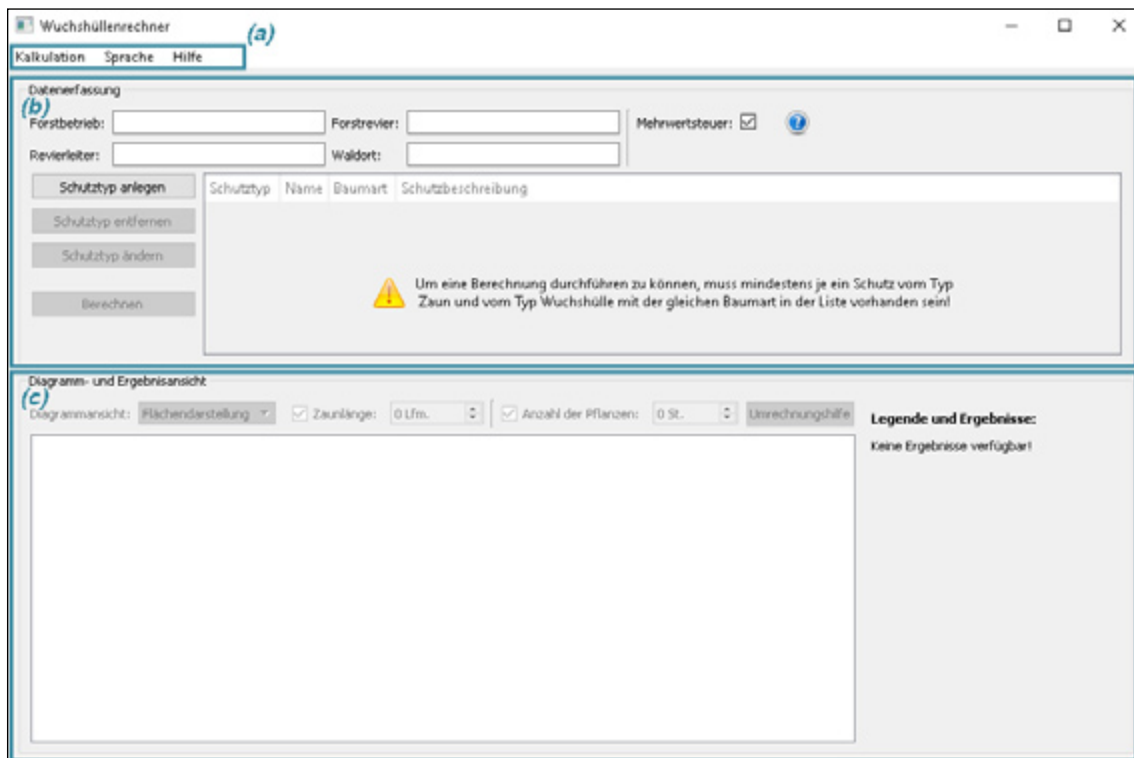


Abbildung 8: Der Aufbau des Hauptfensters

Sobald der *Wuchshüllenrechner* gestartet wird, zeigt sich das *Hauptfenster* der Anwendung. Dabei lässt sich dieses Fenster in drei wichtige Bereiche unterteilen. Demnach ist zuerst die *Menüleiste* (a) am oberen Rand des Hauptfensters dargestellt. Im zweiten Abschnitt folgt mit einer ganzen Gruppe von Eingabeelementen die *Datenerfassung* (b). Ferner befindet sich unterhalb dieser Gruppe die *Diagramm- und Ergebnisansicht* (c). Jeder Bereich erfüllt eine wichtige Aufgabe im Programmablauf und wird daher im Folgenden einzeln vorgestellt.

4.2.1 Die Menüleiste



Abbildung 9: Die Menüleiste am oberen Rand des Hauptfensters

Den ersten Bereich des Hauptfensters bildet die *Menüleiste* (a). Im Wesentlichen besteht sie aus den drei Haupteinträgen *Kalkulation*, *Sprache* und *Hilfe*. Mit einem Klick auf einen dieser Einträge öffnet sich jeweils ein weiteres Untermenü. Hierbei ist der Eintrag *Kalkulation* mit dem üblichen Menüpunkt *Datei* vergleichbar. Da der *Wuchshüllenrechner* jedoch nur Kostenkalkulationen durchführen soll, ist es sinnvoller, den Begriff *Kalkulation* zu verwenden. Im Übrigen besitzen einige Einträge die bekannten Tastenkombinationen, die entsprechend nach dem Eintrag stehen.

4.2.1.1 Der Menüeintrag Kalkulation

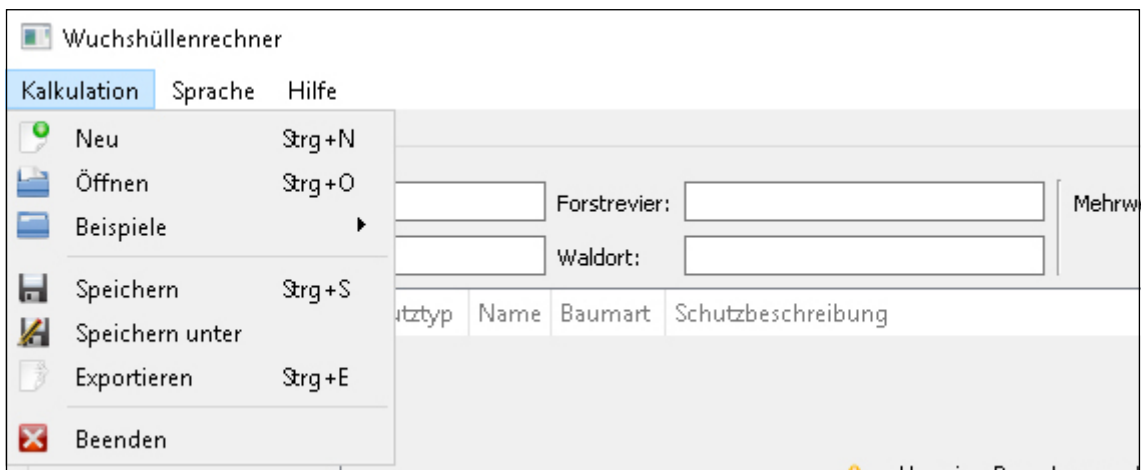


Abbildung 10: Alle Untermenüpunkte unter Kalkulation

Das Untermenü des Haupteintrages *Kalkulation* entspricht der Abbildung 10. Mit Hilfe von *Neu* können die Anwender eine leere Kalkulation anlegen, die auch dem Startzustand des *Wuchshüllenrechners* entspricht. Folglich sind noch keine Varianteneinträge enthalten. Falls vorher bereits mit einer Berechnung gearbeitet wurde und diese noch nicht gespeichert ist, erscheint ein Hinweis zum Speichern des erstellten oder veränderten Datensatzes. Analog verhält sich das Programm, sobald mittels *Öffnen* eine bereits vorhandene Berechnung geladen werden soll. Hierbei wird den Anwendern ein Dialog zur Auswahl einer Datei aus dem Datei-

system gezeigt. Es ist darauf zu achten, dass jeweils nur eine XML-Datei geöffnet werden kann. Zur Unterstützung filtert der *Wuchshüllenrechner* in der Voreinstellung alle Daten, sodass nur XML-Dateien ausgewählt werden können.

Außerdem können die Benutzer die angelegten Berechnungen manuell als XML-Datei speichern. Dazu stehen die beiden Menüeinträge *Speichern* und *Speichern unter* zur Verfügung. Üblicherweise kann mithilfe von *Speichern* eine bereits zuvor gesicherte Kalkulation in der gleichen Datei überschrieben werden. Es kommt zu keiner Rückfrage, wenn die Berechnung vorher einmal gespeichert wurde. Ansonsten fragt der *Wuchshüllenrechner* in einem Auswahldialog, wo die Daten im Dateisystem gesichert werden sollen. So verhält sich auch die Funktion *Speichern unter*, die jedoch die aktuelle Kalkulation immer in einer neuen Datei sichern möchte. Zusätzlich ist es nach der Berechnung möglich, die derzeitige Diagrammansicht als Grafik zu exportieren. Im Auswahldialog stehen dann die drei Dateiformate *JPEG*, *PNG* und *TIFF* zur Verfügung, sodass die Anwender das optimale Format selbst bestimmen können.

Letztlich können die Anwender durch den Menüeintrag *Beenden* das Programm vollständig schließen. Eine nicht gespeicherte Berechnung führt hier ebenfalls zur Rückfrage bei den Benutzern, ob der Datensatz gesichert werden soll.

4.2.1.2 Der Menüeintrag Sprache



Abbildung 11: Alle Untermenüpunkte unter Sprache

Das Untermenü des Eintrages *Sprache* ist in Abbildung 11 zu sehen. Es ist dafür zuständig, den Anwendern eine Sprachauswahl für die Benutzeroberfläche anzubieten. Im Falle des *Wuchshüllenrechners* sind das die beiden Sprachen *Deutsch* und *Englisch*. Mit dem Klick auf die jeweilige Sprache ist es daher möglich, manuell die Übersetzung der Programmoberfläche auszuwählen. Ansonsten wird die Sprachauswahl durch die betriebssystemspezifischen Einstellungen bestimmt.

4.2.1.3 Der Menüeintrag Hilfe

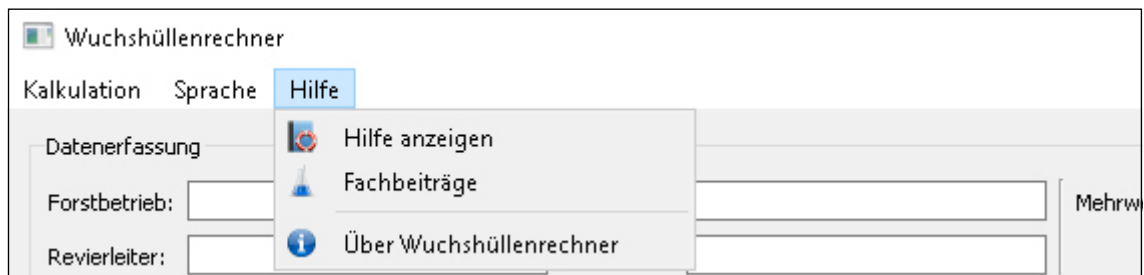


Abbildung 12: Alle Untermenüpunkte unter Hilfe

Der letzte Haupteintrag der Menüleiste ist der Punkt *Hilfe*, der in Abbildung 12 dargestellt ist. Er besteht aus den drei Einträgen *Hilfe anzeigen*, *Fachbeiträge* und *Über Wuchshüllenrechner*. Mittels *Hilfe anzeigen* können die Benutzer die Programmhilfe zum *Wuchshüllenrechner* öffnen, die einfachheitshalber aus einem PDF-Dokument besteht. Das Hilfedokument beschreibt alle Funktionen des *Wuchshüllenrechners*. Hinter dem Eintrag *Fachbeiträge* verbirgt sich ein Dialog mit grundlegender Fachliteratur zur Thematik. Und schließlich zeigt *Über Wuchshüllenrechner* einen Informationsdialog zur Anwendung, der die Programmversion, den Kontakt zu den Autoren sowie den Copyrighthinweis enthält.

4.2.2 Die Gruppe Datenerfassung

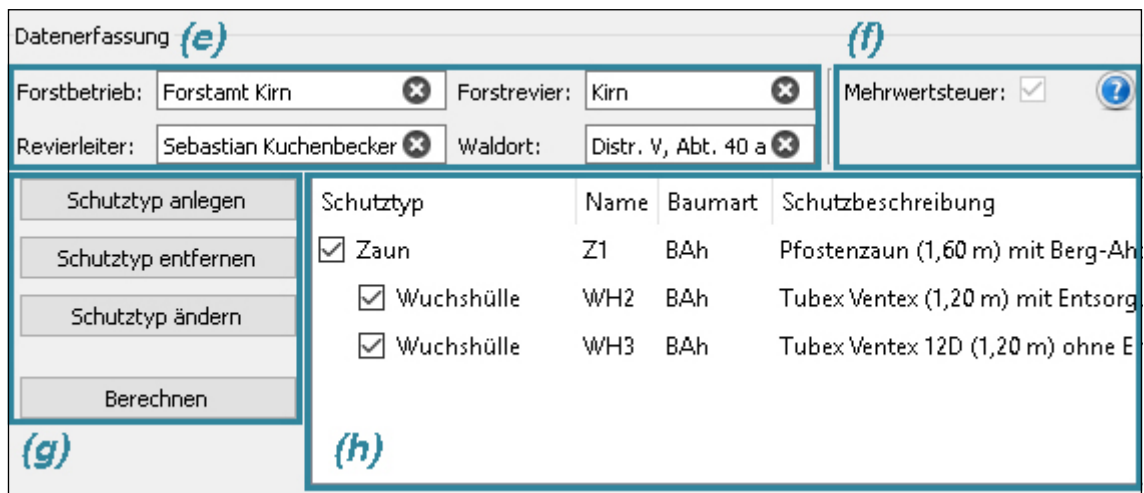


Abbildung 13: Alle Steuerelemente der Gruppe *Datenerfassung*

Da der *Wuchshüllenrechner* mit einer leeren Kalkulation startet, beginnen die Anwender bei einer neuen Berechnung mit der *Datenerfassung* (b). Diese Gruppe enthält unterschiedliche Eingabeelemente. Zum einen befinden sich hier die Textfelder *Forstbetrieb* und *-revier* sowie *Revierleiter* und *Waldort* (e), die nicht

zwingend ausgefüllt werden müssen. Vielmehr dienen sie der Zuordnung der Kalkulation zu einer Organisationseinheit. Außerdem bilden sie beim Ausdrucken eine Kopfzeile mit den entsprechenden Informationen, wobei leere Felder unberücksichtigt bleiben. Daneben können die Anwender festlegen, ob die *Mehrwertsteuer* (f) in ihrer Kalkulation berücksichtigt werden soll. Solange noch kein Schutztyp angelegt wurde, kann das Häkchen gesetzt oder entfernt werden, wobei kein Häkchen bedeutet, dass die Mehrwertsteuer nicht berücksichtigt wird. Dieses Verhalten hat insbesondere Auswirkungen auf die vom *Wuchshüllenrechner* in der Eingabemaske vorgeschlagenen Kostenwerte.

Unterhalb dieser Eingabefelder befinden sich die vier Schaltflächen *Schutztyp anlegen*, *Schutztyp entfernen*, *Schutztyp ändern* und *Berechnen* (g) sowie die *baumartig organisierte Liste* (h) für die angelegten Varianten. Die Schaltfläche *Schutztyp anlegen* ist dazu gedacht, die Eingabemaske für einen neuen Schutztyp zu öffnen. Der sich dann öffnende Dialog *Schutz bearbeiten* bildet alle Eingabefelder für Zaun- und Wuchshüllenelemente ab. Nur auf diesem Weg können Schutzvarianten für die Kalkulation angelegt werden. Es ist aber zu beachten, dass der jeweils angelegte Schutztyp nicht mehr nachträglich verändert werden kann. Das heißt, dass es beim Verändern einer Variante nicht mehr möglich ist, zwischen Zaun oder Wuchshülle zu wählen. Im Gegensatz dazu kann mit dem Knopf *Schutztyp entfernen* eine bereits vorhandene Variante gelöscht werden. Damit der Schalter *Schutztyp entfernen* aktiv dargestellt wird, muss jedoch mindestens eine Schutzvariante in der Liste vorhanden sein. Darüber hinaus werden die Benutzer vor dem Löschvorgang gefragt, ob das ausgewählte Element wirklich gelöscht werden soll.

Die Schaltfläche *Schutztyp ändern* hingegen öffnet wieder den Dialog *Schutz bearbeiten* für das jeweils ausgewählte Element. Damit ist es möglich, die existierenden Eingaben zu überarbeiten. Eine Ausnahme gibt es hierbei dennoch, da der Schutztyp *Zaun* respektive *Wuchshülle* nicht mehr nachträglich geändert werden darf. Auch der Knopf *Schutztyp Ändern* wird erst aktiv erscheinen, sobald mindestens eine Schutzvariante in der Liste vorhanden ist.

Der letzte Schalter dieser Steuerelementgruppe ist die Aktion *Berechnen*. Grundsätzlich wird diese Schaltfläche erst aktiviert, sobald mindestens jeweils eine Schutzvariante vom Typ *Zaun* und vom Typ *Wuchshülle* mit der gleichen Baumart angelegt wurde. Zusätzlich erscheint dieser Hinweistext im *Listenfeld* (h). Damit soll der fehlerhafte Vergleich von Varianten bereits mit der Eingabe verhindert werden. Der Klick auf den Knopf *Berechnen* führt programmintern den Algorithmus aus, der die Schutzvarianten miteinander vergleicht. Das Ergebnis der Kalkulation wird daraufhin in der *Diagramm- und Ergebnisansicht* (c) veranschaulicht. Da nicht immer alle Eingabefehler abgefangen werden können, kann es dennoch passieren, dass der Algorithmus nicht alle Varianten verarbeiten kann. In diesem Fall erhalten dann die Benutzer eine Fehlermeldung wie sie beispielsweise in Abbildung 14 gezeigt ist.

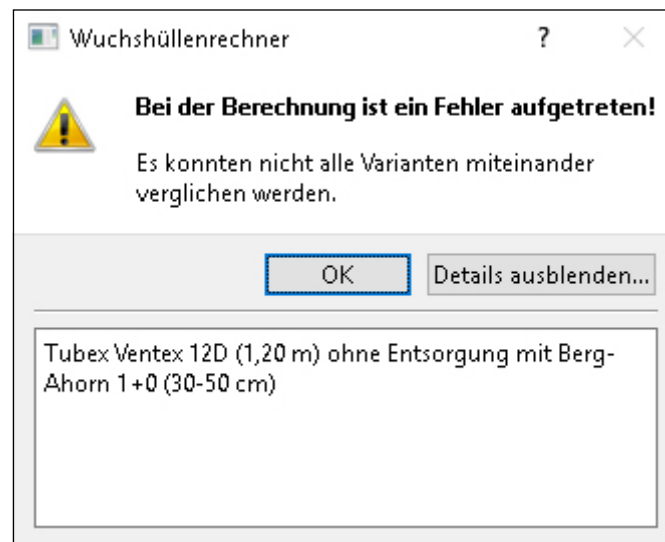


Abbildung 14: Eine Schutzvariante konnte nicht berechnet werden

Die grauen Schaltflächen sind beim Programmstart absichtlich inaktiv, damit die Anwender durch die Programmoberfläche geführt werden können. Als einzige Aktion bleibt somit nur *Schutztyp anlegen*. Durch die Erfüllung der zuvor beschriebenen Bedingungen werden die betreffenden Schaltflächen dann nach und nach aktiv.

4.2.2.1 Die Listenansicht

Alle angelegten Varianten werden der *baumartig organisierten Listenansicht* (h) hinzugefügt. Nebenbei ist die *Listenansicht* gleichzeitig die Legende aller Kostengleichheitsfunktionen. Beim Programmstart ist diese ebenfalls mit der Farbe *grau*

inaktiv dargestellt. Einzig der Hinweistext, dass mindestens jeweils eine Schutzvariante vom Typ *Zaun* und *Wuchshülle* mit der gleichen Baumart angelegt werden soll, erscheint. Hierbei verhält sich die *Listenansicht* analog der Schaltfläche *Berechnen* und bleibt solange inaktiv, bis ein Schutztyp angelegt wurde. Erst mit der Erfüllung der im Hinweis beschriebenen Bedingung wird dieser Text ausgeblendet.

Schutztyp	Name	Baumart	Schutzbeschreibung
<input checked="" type="checkbox"/> Zaun	Z1	BAh	Pfostenzaun (1,60 m) mit Berg-Ahorn 1+1 (50-80 cm)
<input checked="" type="checkbox"/> Wuchshülle	WH2	BAh	Tubex/Ventex (1,20 m) mit Entsorgung und Berg-Ahorn 1+0 (30
<input checked="" type="checkbox"/> Wuchshülle	WH3	BAh	Tubex/Ventex 12D (1,20 m) ohne Entsorgung mit Berg-Ahorn 1-

Abbildung 15: Detaillierte Darstellung der baumartig organisierten Listenansicht (h)

Die *Listenansicht* besitzt eine Kopfzeile, welche die vier Spalten beschriftet. Die erste Spalte zeigt für jede angelegte Variante den zugehörigen *Schutztyp*. Daneben wird ein kleines Kontrollkästchen eingeblendet, das mittels Häkchen geschaltet werden kann. Ein gesetztes Häkchen bedeutet demnach, dass in der *Diagrammansicht* das Ergebnis dieser Variante als Kostengleichheitsfunktion abgebildet wird. Sobald das Häkchen entfernt wird, ist das Ergebnis folglich nicht mehr sichtbar. Das Kontrollkästchen hat aber keinen Einfluss auf die Datenverarbeitung durch den Algorithmus, weshalb unabhängig von der Sichtbarkeit der Variante immer eine Vergleichskalkulation durchgeführt wird. Bei Zaunvarianten muss auf eine Besonderheit hingewiesen werden, da das Entfernen des Häkchens automatisch alle untergeordneten Wuchshüllenvarianten unsichtbar schaltet.

Die Spalte *Name* enthält für jeden Schutztyp eine eindeutige Bezeichnung, um die Variante in der zusätzlichen Legende im Informationsbereich identifizieren zu können. Diese Bezeichnung wird vom *Wuchshüllenrechner* automatisch für jedes Zaun- und Wuchshüllenelement vergeben. Außerdem enthält die Spalte *Baumart* die Kurzform der ausgewählten Baumart, um auf die Sortierung hinzuweisen.

In der letzten Spalte befindet sich die *Schutzbeschreibung* der Variante. Sie wird ebenfalls im Dialog *Schutz bearbeiten* eingegeben und hilft den Benutzern, das Schutzelement eindeutig zu identifizieren. Deshalb sollte für die *Schutzbeschreibung* ein für die Anwender sinnvoller Text eingegeben werden.

Obwohl es wünschenswert wäre, weitere Spalten mit Variantendaten einzufügen, wurde dies aus praktischen Gründen unterlassen. Zum einen müsste dann ein horizontaler Scrollbalken eingeblendet werden, um die Fülle an Informationen anzeigen zu können, da hierfür die Breite der *Listenansicht* nicht genügt. Zum anderen besitzen die Zaun- und Wuchshüllenelemente die eingangs erwähnte *ungleiche Datensatzstruktur*. Praktisch bedeutet dies, dass für ein Zaunelement andere Spalten benötigt werden als für ein Wuchshüllenelement, da die Informationen nicht immer in einer Spalte gebündelt werden können. Das würde die *Listenansicht* zusätzlich unnötig verbreitern.

4.2.3 Der Dialog Schutz bearbeiten

Sobald die Anwender auf eine der beiden Schaltflächen *Schutztyp anlegen* oder *Schutztyp ändern* klicken, öffnet sich der Dialog *Schutz bearbeiten*. Bei einer neuen Variante sind die Eingabefelder entweder leer oder der jeweilige Zahlen- und Kostenwert beträgt 0. Anders als beim Ändern eines vorhandenen Listenelementes ist es bei einer neuen Variante möglich, zwischen den Variantentypen *Zaun* und *Wuchshülle* zu wählen. Zudem werden beim Klick auf die Schaltfläche *Schutztyp ändern* die Eingabefelder mit den gespeicherten Werten der betreffenden Variante befüllt.

Im Allgemeinen ist der Dialog *Schutz bearbeiten*, welcher in der folgenden Abbildung 16 zu sehen ist, in drei Bereiche unterteilt. In der ersten Gruppe (a) wird die *Auswahl des Schutztyps* behandelt sowie zum Beschreiben des Schutzes aufgefordert. Derzeit kann im Feld *Schutztyp* einzig zwischen *Zaun* und *Wuchshülle* gewählt werden, wobei eine Änderung des Typs nachträglich nicht mehr erlaubt ist. Darauf weist auch der Hinweistext im Dialog hin.

Schutz bearbeiten

(a) Auswahl des Schutztyps

Schutztyp: ?

Schutzbeschreibung: ?

(b) Kosten für Pflanze und Pflanzung

Baumart: ?

Stückkosten*: Umrechnungshilfe

Kulturvorbereitungskosten*: ? Umrechnungshilfe

Pflanzungskosten*: ? Umrechnungshilfe

Kultursicherungskosten (5 Jahre)*: ? Umrechnungshilfe

Geringere Mortalität gegenüber Zaun: ?

(c) Kosten für Zaun und Unterhaltung

Zauntyp: ?

Aufbaukosten*: ? Umrechnungshilfe

Unterhaltungskosten*: ?

Abbaukosten*: ?

*) Bitte beachten Sie, dass alle Werte einheitlich die Mehrwertsteuer enthalten oder nicht enthalten müssen.

OK Abbrechen Hilfe

Abbildung 16: Der Dialog *Schutz bearbeiten* in der Voreinstellung

Darüber hinaus dient die *Schutzbeschreibung* dazu, das Listenelement später schnell identifizieren zu können. Es ist daher wichtig, eine sinnvolle Beschreibung zu wählen. Aus diesem Grund wird beim Klick auf die Schaltfläche *OK* überprüft, ob die Benutzer eine Eingabe im Textfeld *Schutzbeschreibung* gemacht haben. Falls diese vergessen wurde, erhalten die Anwender eine Meldung mit dem Hinweis, unbedingt eine *Schutzbeschreibung* einzugeben.

Der zweite Eingabebereich (b) wird unabhängig vom Variantentyp durch die *Kosten für die Pflanze und die Pflanzung* bestimmt. Hierfür sollten die Benutzer über das Listenfeld eine *Baumart* auswählen. Jedoch wird aufgrund der Baumart kein Vorschlagswert für die Kosten angeboten. Auf der Grundlage der Vorüberlegungen zum *Wuchshüllenrechner* werden in diesem Eingabebereich dann die Stückkosten der *Pflanze*, die *Pflanzungs*-, *Kulturvorbereitungs*- und *-sicherungskosten* jeweils in Euro je Stück [EUR/St.] erfasst. Anwender, die den jeweiligen Stückpreis

nicht kennen, unterstützt der *Wuchshüllenrechner* übrigens mittels der Schaltfläche *Umrechnungshilfe*.

Der letzte Bereich (c) des Dialoges *Schutz bearbeiten* ist variabel, weil die angezeigten Eingabefelder vom Schutztyp abhängen. Haben sich die Benutzer für den Typ *Zaun* entschieden, werden hier die *Kosten für den Zaun und seine Unterhaltung* abgefragt. Umgekehrt führt die Auswahl des *Wuchshüllentyps* zur Abfrage der *Kosten für die Wuchshülle und ihre Unterhaltung*.

Im Eingabebereich für die Zaunkosten können die Anwender nach der Auswahl der Schutztyps *Zaun* einen *Zauntyp* selektieren. Dies führt dazu, dass die Felder für *Aufbau-*, *Unterhaltungs-* und *Abbaukosten* mit einem Vorschlagswert belegt werden. Dabei wird der Gesamtkostenwert des Zaunes je Laufmeter nach der 1/3-Regelung auf die drei Kostenfelder aufgeteilt, weil sich in der Praxis häufig jeweils ein Anteil an den Gesamtkosten von rund 30 % ergibt (vgl. KOHNLE; KÄNDLER; WOHNHAS 2016). Findet sich kein passender *Zauntyp* im Listefeld, bietet sich die Auswahl von *Anderer Zauntyp* an. In jedem Fall können die Anwender die Vorschlagskosten überschreiben, da diese nur als grober Richtwert anzusehen sind. Der *Wuchshüllenrechner* erwartet jedoch die Eingabe der Kosten in Euro je Laufmeter [EUR/Lfm.]. Auch hier unterstützt die Anwendung die Benutzer mit einer *Umrechnungshilfe*, sofern die Kosten je laufender Meter nicht bekannt sind.

Ähnlich können die Kosten für die *Wuchshülle und ihre Unterhaltung* eingegeben werden, sobald der Variantentyp *Wuchshülle* ausgewählt wurde. Analog zum *Zaun* fragt das Programm dann den Typ der *Wuchshülle* ab. Hierbei führt die Selektion im Listefeld dazu, dass die Kostenfelder für *Stück- und Zubehörkosten* vorbelegt werden. Mit dem Begriff *Zubehör* ist speziell der Befestigungspfahl der Hülle gemeint. Sollte sich kein passender *Wuchshüllentyp* finden, können die Benutzer *Anderer Wuchshüllentyp* auswählen. Auch hier dürfen alle Vorschlagswerte überschrieben werden und sind als grobe Richtwerte anzusehen. Zusätzlich müssen noch Angaben zu den *Aufbau-*, *Unterhaltungs-* und *Abbaukosten* gemacht werden, wobei diese Kostenfelder ebenfalls Vorschlagswerte enthalten. Dabei müssen alle händisch eingetragenen Kostenwerte in Euro je

Stück [EUR/St.] erfasst werden. An dieser Stelle hilft der *Wuchshüllenrechner* den Anwendern wiederum mit einer *Umrechnungshilfe*.

Allgemein müssen die Benutzer darauf achten, alle Kosten einheitlich inklusive respektive exklusive Mehrwertsteuer zu erfassen. Daran erinnert die Applikation die Anwender durch einen Asterisk (*) an jedem betreffenden Eingabefeld. Das Kontrollkästchen *Mehrwertsteuer* im Hauptfenster bewirkt nur, dass die Vorschlagswerte mit dem richtigen Steuersatz (19 %) berechnet werden.

Nachdem alle Eingaben getätigt wurden, kann der Schutztyp abschließend mit einem Klick auf die Schaltfläche *OK* zur baumartig organisierten Liste im Hauptfenster hinzugefügt werden.

4.2.3.1 Die Eingabe der Baumart

Im Dialog *Schutz bearbeiten* verlangt der *Wuchshüllenrechner* im Bereich *Kosten für die Pflanze und die Pflanzung* die Auswahl einer Baumart aus dem Listefeld. Dies ist zum einen notwendig, um die Varianten in der baumartig organisierten Liste zu sortieren. Zum anderen stellt die Baumart speziell beim Ausdruck eine Information für die Benutzer dar.

Aber die Anwender können für jeden neuen Schutztyp nur eine Baumart nennen. Eine Mehrfachauswahl ist dagegen nicht möglich, weshalb das waldbauliche Szenario einer Mischkultur nicht abgebildet werden kann. Sollen beispielsweise auf einer Fläche die Stiel-Eiche mit der dienenden Baumart Hainbuche und beigemischte Vogel-Kirschen gepflanzt werden, würden hierzu pro Baumart jeweils eine Zaun- und eine Wuchshüllenvariante zur Kalkulation benötigt, sofern auch die dienende Baumart berücksichtigt werden soll. In diesem Fall, legen die Anwender aber nur jeweils eine Schutzvariante vom Typ *Zaun* und vom Typ *Wuchshülle* an und setzen die Kostenwerte der Hauptbaumart Stiel-Eiche ein. Jedoch wird die Gesamtzahl der Pflanzen sowie die gesamte Lauflänge des Zaunes berücksichtigt. Um ein genaueres Ergebnis zu erhalten, ist es möglich, für die Stück- und Pflanzungskosten einen Durchschnittswert zu ermitteln, indem diese Kosten der betreffenden Baumarten durch ein gewichtetes Mittel errechnet werden.

4.2.3.2 Die Vorschlagswerte für Zaun und Wuchshülle

Der *Wuchshüllenrechner* wurde so entwickelt, dass für die beiden Schutztypen *Zaun* und *Wuchshülle* im Dialog *Schutz bearbeiten* bereits Kosten vorgeschlagen werden. Diese sogenannten Vorschlagswerte beziehen sich bei einem Zaun auf die drei Eingabefelder *Aufbau-*, *Unterhaltungs-* und *Abbaukosten*. Sobald die Benutzer einen Zauntyp aus dem Listefeld wählen, werden die Felder mit den betreffenden Werten gefüllt. Alle Kostenwerte für die genannten Zauntypen stammen sowohl aus dem kwf-Merkblatt „Schutzmaßnahmen gegen Wildschäden im Wald“ (vgl. KWF 2012, S. 10 ff.) als auch aus der aktuellen Ausgabe AFZ-DerWald (vgl. KWF 2016, S. 20-22). Da die berechneten Kostenwerte aktuell sind, wurde auf eine eigene Kalkulationen verzichtet.

Die Vorschlagswerte für die genannten Wuchshüllentypen wurden aus dem Onlineshop des Anbieters *Grube* entnommen (GRUBE KG 2015). Bei der Auswahl einer Wuchshülle werden alle Felder befüllt, wobei sich die Stückkosten in der Regel auf eine Größenordnung von 500 Exemplaren beziehen, um einen brauchbaren Mittelwert bieten zu können. Mit dem Zubehör ist der Befestigungsstab der Hülle gemeint. Da sich die Kostenwerte aufgrund der Höhe des Stabes unterscheiden, wird anhand der Wuchshüllenhöhe automatisch die richtige Stabhöhe ermittelt und dementsprechend der Kostenwert im Eingabefeld eingesetzt.

Es ist wichtig darauf hinzuweisen, dass diese Vorschlagswerte nur grobe Richtwerte sind. Ein Grund hierfür sind die je nach Forstbetrieb unterschiedlichen Material- und Betriebskosten, weil jeder Betrieb die Schutzmaterialien zu anderen Konditionen erhält und beispielsweise die Arbeitsleistung sowie die Löhne variieren. Außerdem verändern sich mit der Zeit diese Kosten, weshalb die Vorschlagswerte nicht immer aktuell sein können. Aus diesen Gründen empfiehlt es sich, auf die betriebseigene Kostenstruktur zurückzugreifen, um genauere Ergebnisse mittels der Entscheidungshilfe zu erzielen.

Darüber hinaus wird die Mehrwertsteuer in den Vorschlagswerten berücksichtigt. Hierfür wird von der *Regelbesteuerung* des Forstbetriebes ausgegangen und damit der Steuersatz 19 % angenommen, da die jeweiligen Schutzmaterialien nicht

in der *Anlage 2* zum Umsatzsteuergesetz aufgeführt sind und damit kein ermäßigter Steuersatz gewährt wird (vgl. ANLAGE 2 USTG). Bei nach Durchschnittssätzen besteuerten Betrieben wird empfohlen, die Kalkulation ohne Mehrwertsteuer durchzuführen oder auf die betriebseigene Kostenstruktur zurückzugreifen. Der Steuersatz bleibt unberücksichtigt, wenn die Benutzer im Hauptfenster des *Wuchshüllenrechners* das Häkchen aus dem Feld *Mehrwertsteuer* entfernt haben. Das heißt, dass die steuerfreien Vorschlagswerte in die Kalkulation einfließen.

4.2.4 Die Diagramm- und Ergebnisansicht

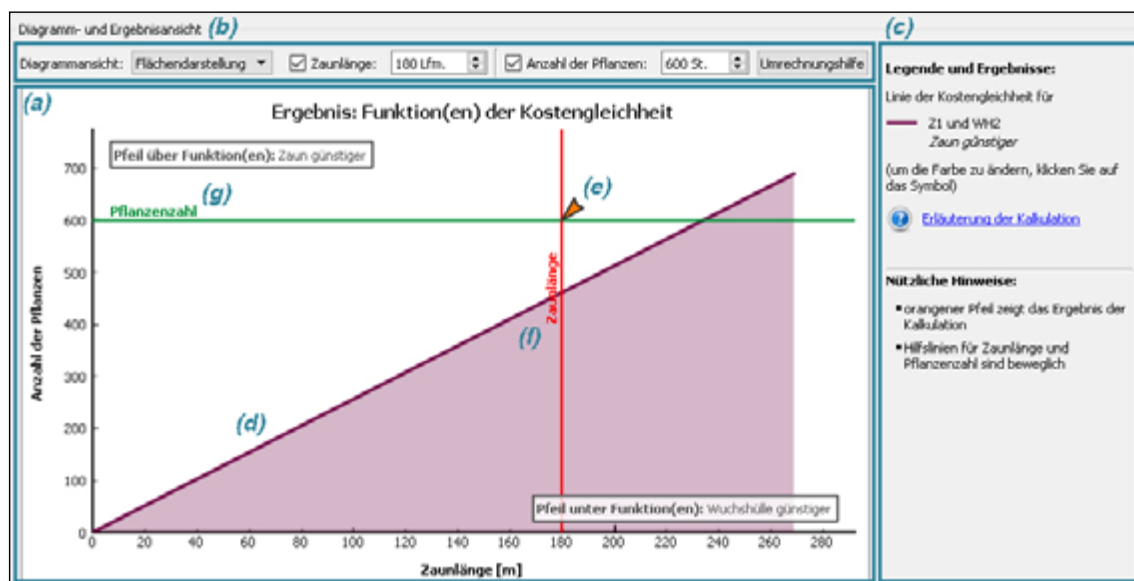


Abbildung 17: Detaillierte Diagramm- und Ergebnisansicht mit Flächendarstellung

Der dritte Abschnitt im Hauptfenster besteht aus der *Diagramm- und Ergebnisansicht* (c), wobei auch dieser Bereich in drei Bestandteile gegliedert werden kann. Zentral befindet sich die *Diagrammansicht* (a), die beim Starten des Wuchshüllenrechners aus einer weißen Fläche besteht und vorerst keine weiteren Elemente zeigt. Darüber befindet sich die sogenannte *Kontrollleiste* (b). Im gesamten rechten Teilbereich erhalten die Anwender eine Übersicht zu den kalkulierten Ergebnissen sowie einige Hinweise, weshalb vom *Informationsbereich* (c) gesprochen wird.

Die *Diagrammansicht* (a) verändert sich erst, wenn die Anwender auf die Schaltfläche *Berechnen* innerhalb der Gruppe *Datenerfassung* geklickt haben. Dann wird vom Algorithmus der Vergleich der Varianten ausgeführt und das betref-

fende Ergebnis als *Funktionsgerade* im Diagramm eingezeichnet. Dabei ist auf der Abszissenachse des Diagramms die *Zaunlänge in Metern* abgetragen. Die Ordinatenachse hingegen besteht aus der *Anzahl der Pflanzen*. Somit wird der Graph der *Kostengleichheitsfunktion* (d) zwischen der Zaunlänge und der Anzahl der Pflanzen gezeichnet. Dass der Funktionsgraph tatsächlich im Diagramm dargestellt wird, hängt davon ab, ob die jeweilige Variante im Bereich *Datenerfassung* (b) durch das Kontrollkästchen in der Spalte *Schutztyp* der *Listensicht* (h) aktiviert wurde. Nur die dort aktivierten Varianten werden eingezeichnet.

4.2.4.1 Das Ergebnis

Damit eine Entscheidungsempfehlung mit dem *Wuchshüllenrechner* möglich ist, muss der *Schnittpunkt* zwischen der *Zaunlänge* und der *Pflanzenzahl* gefunden werden. Deshalb schlägt die Anwendung nach der ersten Berechnung innerhalb einer Kalkulation den Schnittpunkt bei einer Zaunlänge von 400 Metern und 900 Pflanzen vor. Dieser Punkt ist eindeutig durch den *orangenen Pfeil* (e) markiert. Dort trifft sich die *vertikale rote Hilfslinie* für die Länge des Zaunes (f) mit der *horizontalen grünen Hilfslinie* für die Anzahl der Pflanzen (g). Im Diagramm wird nur eine Hilfslinie für die *Zaunlänge* und die *Anzahl der Pflanzen* eingeblendet.

Mithilfe dieser beiden Linien kann der Pfeil verschoben werden, um den passenden Schnittpunkt für die jeweilige Kalkulation zu finden. Hierzu verschiebt die *rote Hilfslinie* den Pfeil entlang der Abszissenachse durch eine Links- oder Rechtsbewegung. Die Bewegung der *grünen Hilfslinie* nach oben oder nach unten führt dazu, dass der Pfeil entlang der Ordinatenachse verschoben wird. Folglich können die Benutzer mit diesen Werkzeugen ihre tatsächlichen Werte für die Zaunlänge und die Pflanzenzahl festlegen.

Die Entscheidungsempfehlung, ob nun der Zaun oder die Wuchshülle günstiger ist, hängt von der Lage des Schnittpunktes ab. Grundsätzlich bedeutet nach HAMMER ein unterhalb der Kostengleichheitsgeraden liegender Pfeil, dass die Wuchshüllenvariante günstiger ist. Dies gilt prinzipiell für die gesamte Fläche unter der Geraden. Im Gegensatz dazu meint ein oberhalb der Funktionsgeraden liegender

Pfeil, dass die Zaunvariante günstiger ist. Dies gilt entsprechend für die gesamte Fläche über der Geraden. Liegt der Schnittpunkt genau auf dem Graphen der Funktion, sind beide Varianten bezüglich der Kosten gleichwertig. Zum besseren Verständnis wurden für die Anwender Hinweise im Diagramm eingefügt, die sich jeweils in der oberen linken und unteren rechten Ecke befinden.

4.2.4.2 Die Kontrollleiste

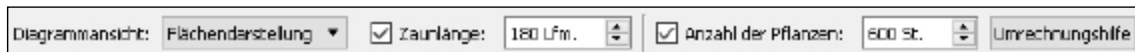


Abbildung 18: Detaillierte Darstellung der Kontrollleiste

Weitere Eingabemöglichkeiten, um das Diagramm zu steuern, erhalten die Anwender durch die *Kontrollleiste* (b), welche die drei Felder *Diagrammansicht*, *Zaunlänge* und *Anzahl der Pflanzen* enthält. Sie befindet sich direkt über dem Diagramm und ist nach dem Start der Anwendung inaktiv. Die Steuerelemente werden erst durch den Klick auf die Schaltfläche *Berechnen* innerhalb der Gruppe *Datenerfassung* aktiviert.

Auf der linken Seite der Kontrollleiste ist das Listefeld *Diagrammansicht* angeordnet. Hierbei bietet die Liste die Auswahl zwischen der *Flächendarstellung* und der *Liniendarstellung*, wobei sich das Diagramm je nach gewählter Darstellung verändert. Die *Flächendarstellung* wird vom *Wuchshüllenrechner* automatisch bei jeder neuen Berechnung gewählt, da sie sehr aussagekräftig erscheint (Abbildung 17). Dabei wird zusätzlich zur Funktionsgeraden der Kostengleichheit die gesamte Fläche unterhalb des Graphen mit dessen Farbe eingefärbt. Die Fläche oberhalb des Graphen bleibt weiß. Durch die Färbung soll es für die Anwender einfacher sein, das abgebildete Ergebnis zu verstehen, da die Kolorierung die Wuchshüllenvariante eindeutig gegenüber der Zaunvariante hervorhebt. Das heißt, es wird zusätzlich grafisch veranschaulicht, in welcher Fläche sich der Schnittpunkt zwischen der Zaunlänge und der Pflanzenzahl befindet. In dieser Darstellung kann jedoch aus Gründen der Übersicht nur eine Funktionsgerade abgebildet werden. Deshalb müssen die Anwender manuell in der *Datenerfassung* (b) durch das Kontrollkästchen in der Spalte *Schutztyp* der *Listenansicht* (h) nur eine Zaun- und Wuchshüllenvariante mit der gleichen Baumart aktivieren. Ansonsten erscheint

eine Meldung mit dem Hinweis, dass zu wenige oder zu viele Varianten aktiv geschaltet wurden und in der Flächendarstellung nicht angezeigt werden können.

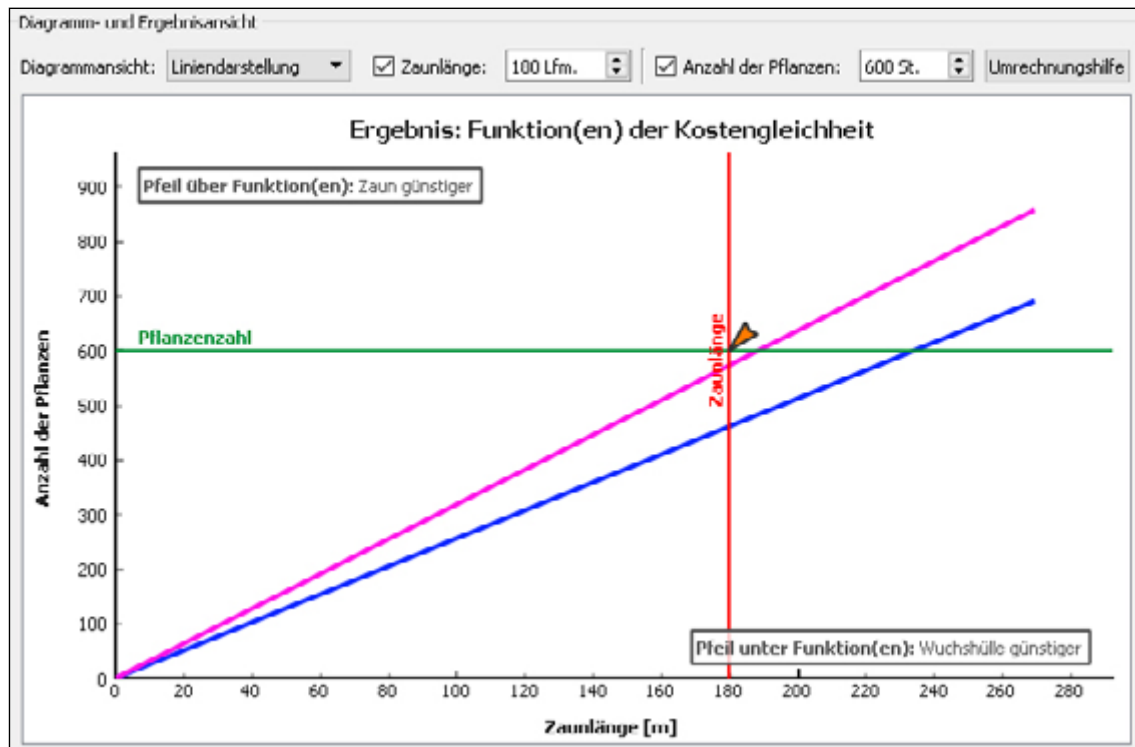


Abbildung 19: Die Liniendarstellung mit zwei Wuchshüllenvarianten

Die *Liniendarstellung* ist die vereinfachte Form der Flächendarstellung. Ist sie ausgewählt, wird ausschließlich der Funktionsgraph ohne zusätzliche Färbung einer Fläche angezeigt. Dabei bleiben alle weiteren Elemente im Diagramm unbeeinflusst. In der Liniendarstellung ist es demnach möglich, mehrere Funktionsgeraden abzubilden, wobei auch hier die Steuerung bezüglich der Sichtbarkeit durch das Kontrollkästchen in der *Listenansicht* (h) stattfindet. Die Auswahl dieser Darstellung ist dann sinnvoll, wenn zwei im Bezug auf die Kosten unterschiedliche Wuchshüllenvarianten mit einem Zaunelement verglichen werden sollen. Dies ist beispielsweise der Fall, wenn eine Wuchshüllenvariante Entsorgungskosten enthält, weil sie nach dem Schutzzweck abgebaut werden muss. Die zweite Wuchshüllenvariante soll aber ohne Entsorgungskosten auskommen, da sich die Hülle nach dem Schutzzweck selbst abbaut. In diesem Beispiel in Abbildung 19 zeigt der *Wuchshüllenrechner* in der Liniendarstellung nicht nur, wann sich ein Zaun lohnt, sondern ebenfalls den Unterschied der beiden Wuchshüllenvarianten in Relation zum Zaun.

Die Kontrollleiste bietet außerdem in der Mitte ein Steuerelement für die Eingabe der Zaunlänge. Es weist ein Kontrollkästchen sowie ein Eingabefeld auf. Wird das Häkchen aus dem Kontrollkästchen entfernt, erscheint das gesamte Steuerelement inaktiv. Darüber hinaus wird die *vertikale rote Linie* zur Zaunlänge im Diagramm ausgeblendet. Somit ist es nicht mehr möglich, die Zaunlänge einzugeben. Erst mit dem Setzen des Häkchens im Kontrollkästchen werden das Steuerelement und die *vertikale rote Linie* wieder aktiviert.

Das Eingabefeld bietet zwei Funktionen. Zum einen können die Benutzer hier eine Zaunlänge eingeben, um damit die *rote Linie* und folglich den *Schnittpunkt* entlang der Abszissenachse punktgenau zu verschieben. Zum anderen zeigt das Eingabefeld die aktuelle Zaunlänge, sobald die *rote Linie* im Diagramm verschoben wird.

Neben der Zaunlänge können die Anwender ebenfalls die *Anzahl der Pflanzen* im gleichnamigen Steuerelement festlegen. Das Verhalten ist analog zum bereits erklärten Eingabeelement *Zaunlänge*. Im Gegensatz dazu verschiebt sich jedoch die *horizontale grüne Linie* entlang der Ordinatenachse, die beim Entfernen des Häkchens im Kontrollkästchen ausgeblendet wird. Zudem bietet dieses Steuerelement mit der Schaltfläche *Umrechnungshilfe* eine Unterstützung zur Ermittlung der genauen Pflanzenzahl.

4.2.4.3 Der Informationsbereich

Der rechte Teilbereich der Diagramm- und Ergebnisansicht ist der *Informationsbereich* (c), der im Detail in Abbildung 20 zu sehen. Dort wird einerseits kurz erläutert, welche Empfehlung die Benutzer durch den *Wuchshüllenrechner* erhalten. Das heißt, es wird eindeutig beschrieben, welcher Schutztyp günstiger ist. Zudem können die Anwender durch einen Klick auf den Link *Erläuterung der Kalkulation* theoretisch nachvollziehen, welche Rechenschritte der Algorithmus durchführt. Dazu wird auf das entsprechende Kapitel in der Dokumentation verlinkt.

Andererseits zeigt das Programm im Informationsbereich einige Hinweise für die Anwender. Demnach ist verständlich erklärt, dass der *orange Pfeil* am Schnittpunkt das Ergebnis der Kalkulation darstellt. Ebenso werden die Benutzer dazu

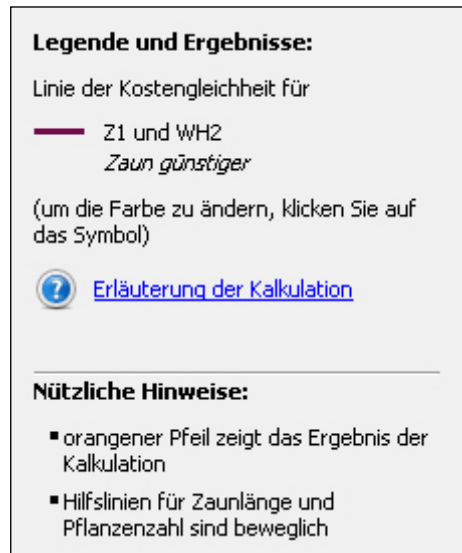


Abbildung 20: Detailansicht des Informationsbereiches

aufgefordert, die *vertikale rote Linie* für die Zaunlänge sowie die *horizontale grüne Linie* für die Pflanzenzahl zu bewegen.

Darüber hinaus ist es mithilfe eines Klicks auf das farbige Symbol in der Legende möglich, die Kolorierung der jeweiligen Kostengleichheitsfunktion zu ändern. Dazu öffnet sich anschließend der betriebssystemübliche Auswahldialog für Farben. Nach dem Klick auf *OK*, wird dann die neue Färbung automatisch übernommen.

4.3 Beispielkalkulation mit Berg-Ahorn

Um den Rechenweg besser nachvollziehen zu können, wurde im Folgenden eine Beispielberechnung anhand einer Berg-Ahorn-Monokultur durchgeführt. Dabei handelt es sich um eine Kulturfläche von 1.800 m², wobei die Pflanzen in einem Verband von 2 x 1,5 Metern ausgebracht werden sollen. Folglich werden 600 Berg-Ahorne benötigt. Bei einer Breite von 30 Metern und eine Länge von 60 Metern beträgt der Umfang und damit die Länge des Zaunes 180 Laufmeter. Dieser Flächenschutz soll in Form eines Pfostenzaunes mit einer Bauhöhe von 1,60 Metern stattfinden. Bei den Wuchshüllen soll das Produkt *Ventex* mit einer Höhe von 1,20 Metern der Firma *Tubex* verwendet und anschließend entsorgt werden. Alle weiteren Kostenangaben und relevanten Zahlen sind der folgenden Tabelle 2 zu entnehmen.

Kostenwerte	Zaunschutz	Einzelschutz
Berg-Ahorn	1,05 EUR/St. (600 St.)	0,40 EUR/St. (600 St.)
<i>Sortiment</i>	<i>1+1; 50 bis 80 cm</i>	<i>1+0; 30 bis 50 cm</i>
Kulturvorbereitung	0,00 EUR/St.	0,00 EUR/St.
Pflanzung	1,17 EUR/St.	0,74 EUR/St.
Kultursicherung (5 Jahre)	0,30 EUR/St.	0,15 EUR/St.
geringere Mortalität		ca. 10 %
Pfostenzaun (1,60 m)	11,94 EUR/Lfm. (180 Lfm.)	
Tubex Ventex (1,20 m) + Stab		3,29 EUR/St. (600 St.)
Aufbau	(im Gesamtpreis enthalten)	1,62 EUR/St.
Unterhaltung	(im Gesamtpreis enthalten)	0,00 EUR/St.
Abbau inklusive Entsorgung	(im Gesamtpreis enthalten)	1,78 EUR/St.

Tabelle 2: Kostenstruktur der Berg-Ahorn-Monokultur

Alle Stückkosten der Pflanze stammen aus dem aktuellen Katalog der *Forstbauschule Stingel* und wurden mit 7 % Mehrwertsteuer kalkuliert. Die Arbeitskosten für die Pflanzung, die Kultursicherung sowie für den Aufbau der Wuchshülle entsprechen den Beispielwerten von HAMMER. Bei den Abbaukosten der Hülle wurden zusätzlich zum Richtwert von 1,25 EUR/St. die Transport- und Entsorgungskosten von 0,53 EUR/St. addiert (vgl. GÖCKEL; KOPP; WICHT-LÜCKGE 2012, S. 28-29). Die Gesamtkosten für den Zaun entstammen den Kostenbeispielen des KWF (vgl. KWF 2016, S. 22). Erwähnenswert ist, dass diese Kostenwerte alle mit 19 % Mehrwertsteuer berechnet wurden.

Nachdem alle Kostenwerte im *Wuchshüllenrechner* eingegeben wurden, werden programmintern die folgenden Berechnungen durchgeführt. Dazu werden die Zaunkosten anfangs in die Kalkulationsformel für den Zaun eingesetzt:

$$K_1 = x \cdot \frac{11,94 \text{ EUR}}{m} + y \cdot \left(\frac{1,05 \text{ EUR}}{\text{St.}} + \frac{1,17 \text{ EUR}}{\text{St.}} + \frac{0,30 \text{ EUR}}{\text{St.}} \right)$$

Danach müssen die Wuchshüllenkosten in die Kalkulationsformel für die Wuchshülle eingesetzt werden:

$$K_2 = y \cdot \left(\frac{0,40 \text{ EUR}}{\text{St.}} + \frac{0,74 \text{ EUR}}{\text{St.}} + \frac{0,15 \text{ EUR}}{\text{St.}} + \frac{3,29 \text{ EUR}}{\text{St.}} + \frac{3,40 \text{ EUR}}{\text{St.}} \right) \cdot 0,9$$

Nach dem Gleichsetzen der beiden Formeln ergibt sich das bereits bekannte Schema der allgemeinen Kalkulationsformel. Mit den eingesetzten Kostenwerten stellt sich dies wie folgt dar:

$$y = \frac{\frac{11,94 \text{ EUR}}{m}}{\left(\frac{7,98 \text{ EUR}}{\text{St.}}\right) \cdot 0,9 - \left(\frac{2,52 \text{ EUR}}{\text{St.}}\right)} \cdot x$$

$$y = \frac{\frac{11,94 \text{ EUR}}{m}}{\frac{4,662 \text{ EUR}}{\text{St.}}} \cdot x$$

$$y = \frac{2,561 \text{ St.}}{m} \cdot x$$

Das Ergebnis ist die Steigung von 2,561. Dieser Wert wird ebenfalls vom *Wuchshüllenrechner* ermittelt. Nun muss die Zaunlänge als Funktionswert x in die lineare Funktion eingesetzt werden, um als Funktionsergebnis y den Punkt der Kostengleichheit zu erhalten. In diesem Beispiel liegt dieser Punkt bei 461 Pflanzen (abgerundet), was bedeutet, dass der Zaun und die Wuchshülle bezüglich der Kosten gleichwertig sind.

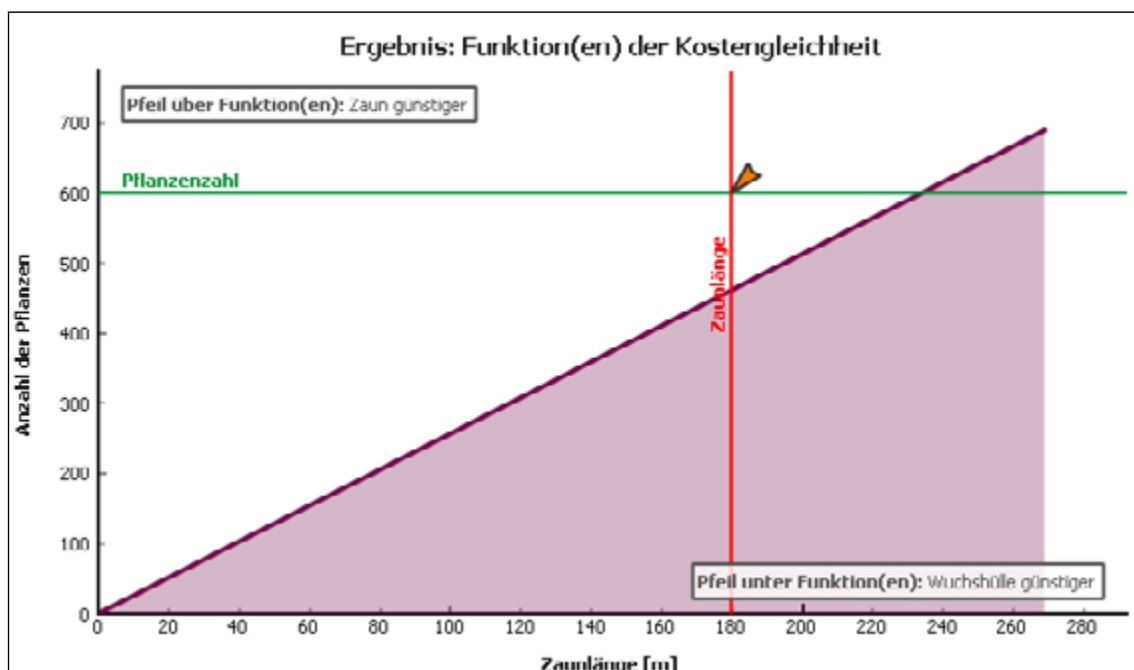


Abbildung 21: Veranschaulichung der Beispielkalkulation

Nun beträgt die tatsächliche Pflanzzahl in der Fläche jedoch 600 Stück. Da dieser Wert größer ist, als der errechnete Punkt der Kostengleichheit, lohnt sich die

Zaunvariate mehr. Um dies zu veranschaulichen, wurde die selbe Kalkulation im Wuchshüllenrechner durchgeführt. Das Ergebnis zeigt die Abbildung 21.

4.4 Ergebnis der Evaluation

Von den 40 angeschriebenen Testpersonen gab es 10 Rückmeldungen zur Software. Der Inhalt ist im Folgenden stichpunktartig zusammengefasst, da es oftmals ähnliche Rückmeldungen gab:

- Start des *Wuchshüllenrechners* durch den Virenschutz verhindert
- Funktion *Speichern unter* fehlt
- Kosten für *Kulturvorbereitung* berücksichtigen
- Mortalitätsrate unverständlich und an störender Position
- Zaunlänge und Pflanzenzahl einheitlich abfragen und nicht für jede Variante getrennt
- bessere Erläuterung der Diagramme (Fläche, Schnittpunkt) und des Ergebnisses gewünscht
- Verwirrung durch *Anderer Zauntyp* und *Anderer Wuchshüllentyp*
- unklare Benutzerführung (Benennung der Felder mit *Variante*)
- Vergleich anderer Schutzverfahren
- Hinweis beim Löschen einer Variante gewünscht

In der weiteren Entwicklung des *Wuchshüllenrechners* wurde versucht, einige dieser Punkte abzuarbeiten und damit die Anwendung zu verbessern. Aufgrund des engen Zeitrahmens war es nicht möglich, alle Punkte zu berücksichtigen, sodass diese teilweise in den Ausblick übernommen wurden.

5 Diskussion des Ergebnisses

5.1 Berücksichtigung von Mischkulturen

Bei der Verwendung des *Wuchshüllenrechners* fällt auf, dass es nicht möglich ist, beim Anlegen einer neuen Schutzvariante mehrere Baumarten auszuwählen. Stattdessen kann innerhalb des Listenfeldes im Dialog *Schutz bearbeiten* nur eine Baumart festgelegt werden. Aus diesem Grund stellt sich die Frage, in welcher Art und Weise *Mischkulturen* mit verschiedenen Baumarten in der Kalkulation berücksichtigt werden können?

Um eine solche Mischkultur im *Wuchshüllenrechner* abbilden zu können, wurde anfangs ein Lösungsweg erarbeitet. Dabei werden die Baumarten anhand ihres ideellen Flächenanteils in der Berechnung berücksichtigt. Deshalb sollten die Anwender den jeweiligen Anteil der Baumart an der Gesamtfläche der Kultur eingeben. Mithilfe des entsprechend festgelegten Prozentsatzes kann die Anwendung intern die Kosten gewichten. Unabhängig von diesem gewichteten Mittel bleibt jedoch die Zaunlänge. Für diesen Wert sollten die Benutzer die tatsächliche Zaunlänge der Kulturfläche eingeben. Die Anwendung muss jedoch zwischen einer gewichteten Kalkulation mit mehreren Baumarten und einer einfachen Kalkulation mit nur einer Baumart unterscheiden. Um dies zu realisieren, sollten die Anwender bei einer neuen Kalkulation gefragt werden, ob es sich um eine Monokultur oder um eine Mischkultur handelt.

Dieser Lösungsweg ist nur eine Idee und wurde so nicht im *Wuchshüllenrechner* umgesetzt. Insbesondere ist der gewählte Ansatz ausgesprochen kompliziert. Er stellt dennoch eine kalkulatorisch richtige Lösung dar, Mischkulturen zu berücksichtigen. Es zeigte sich jedoch im Laufe der Programmentwicklung, dass die Betrachtung von Mischkulturen vorerst nicht notwendig erscheint, da der resultierende Fehler gering bleibt und die Entscheidungsempfehlung kaum beeinflusst. Das beweist auch die im Folgenden modifizierte Beispielkalkulation aus Kapitel 4.3.

Kostenwerte	Zaunschutz	Einzelerschutz
Berg-Ahorn	1,31 EUR/St. (480 St.)	0,50 EUR/St. (480 St.)
<i>Sortiment</i>	<i>1+1; 50 bis 80 cm</i>	<i>1+0; 30 bis 50 cm</i>
Vogel-Kirsche	1,61 EUR/St. (120 St.)	0,74 EUR/St. (120 St.)
<i>Sortiment</i>	<i>1+1; 50 bis 80 cm</i>	<i>1+0; 30 bis 50 cm</i>
Kulturvorbereitung	0,00 EUR/St.	0,00 EUR/St.
Pflanzung	1,17 EUR/St.	0,74 EUR/St.
Kultursicherung (5 Jahre)	0,30 EUR/St.	0,15 EUR/St.
geringere Mortalität		ca. 10 %
Pfostenzaun (1,60 m)	11,94 EUR/Lfm. (180 Lfm.)	
Tubex Ventex (1,20 m) + Stab		3,29 EUR/St. (600 St.)
Aufbau	<i>(im Gesamtpreis enthalten)</i>	1,62 EUR/St.
Unterhaltung	<i>(im Gesamtpreis enthalten)</i>	0,00 EUR/St.
Abbau inklusive Entsorgung	<i>(im Gesamtpreis enthalten)</i>	1,78 EUR/St.

Tabelle 3: Veränderte Kostenstruktur für das Beispiel einer Mischkultur

Grundsätzlich bleiben dabei die Fläche mit 1.800 m² und der Pflanzverband mit 2 x 1,5 Metern gleich. Deshalb sollen insgesamt wieder 600 junge Pflanzen ausgebracht werden, die alle aus der *Forstbaumschule Stingel* stammen. In diesem Szenario sollen aber 480 Berg-Ahorne mit 120 Vogel-Kirschen als Mischkultur gepflanzt werden. Die genauen Kulturkosten sind der Tabelle 3 zu entnehmen. Zusätzlich hat die Pflanzfläche eine Breite von 30 Metern sowie eine Länge von 60 Metern, was dem Umfang von 180 Laufmetern entspricht.

Kostenwerte	Preisdifferenz <i>(groß - klein)</i>	Preisdifferenz <i>(groß: Kir - BAh)</i>	Preisdifferenz <i>(klein: Kir - BAh)</i>
Berg-Ahorn	0,81 EUR <i>(1,31 - 0,50)</i>	0,30 EUR <i>(1,61 - 1,31)</i>	0,24 EUR <i>(0,74 - 0,50)</i>
Vogel-Kirsche	0,87 EUR <i>(1,61 - 0,74)</i>		

Tabelle 4: Preisdifferenzen zwischen den Baumarten

Da es sich um eine geringe Abnahmemenge handelt, musste mit dem Preis für die Verpackungseinheit von 100 Pflanzen gerechnet werden (vgl. FORSTBAUMSCHULE STINGEL 2015). Entscheidend für die vorliegende Kalkulation ist aber nicht nur der Preisunterschied zwischen den Baumarten, sondern speziell die

Preisdifferenz zwischen dem größeren und damit teureren Sortiment für die Zaunvariante und dem kleineren und folglich günstigeren Sortiment für die Wuchshüllenvariante. Alle Preisunterschiede wurden hierzu in Tabelle 4 aufgedröselte. Damit lässt sich folglich ableiten, dass die relative Preisdifferenz zwischen den Sortimenten für Vogel-Kirsche größer ist, als es bei den Sortimenten für Berg-Ahorn der Fall ist.

Schutztyp	Name	Baumart	Schutzbeschreibung
<input checked="" type="checkbox"/> Zaun	Z5	sLb	Pfostenzaun (1,60 m) mit 480 Berg-Ahorn und 120 Vogel-Kirschen 1+1 (50-80 cm)
<input checked="" type="checkbox"/> Wuchshülle	WH6	sLb	Tubex Ventex (1,20 m) mit Entsorgung und 480 Berg-Ahorn und 120 Vogel-Kirschen 1+0 (30-50 cm)
<input checked="" type="checkbox"/> Zaun	Z3	Kir	Pfostenzaun (1,60 m) mit 600 Vogel-Kirschen 1+1 (50-80 cm)
<input checked="" type="checkbox"/> Wuchshülle	WH4	Kir	Tubex Ventex (1,20 m) mit Entsorgung und 600 Vogel-Kirschen 1+0 (30-50 cm)
<input checked="" type="checkbox"/> Zaun	Z1	BAh	Pfostenzaun (1,60 m) mit 600 Berg-Ahorn 1+1 (50-80 cm)
<input checked="" type="checkbox"/> Wuchshülle	WH2	BAh	Tubex Ventex (1,20 m) mit Entsorgung und 600 Berg-Ahorn 1+0 (30-50 cm)

Abbildung 22: Die Datenliste der Mischkultur im Wuchshüllenrechner

Mithilfe dieser Grunddaten für eine Kostenkalkulation einer Kultur soll gezeigt werden, dass es nicht notwendig ist, mehrere Baumarten mit unterschiedlichen Pflanzenpreisen im *Wuchshüllenrechner* einzugeben, um eine plausible Entscheidungsempfehlung zu erhalten. Dafür wurden im Programm drei Variantenpaare und damit insgesamt sechs Schutzvarianten verglichen. Das erste Schutzpaar bilden ein Zaun sowie eine Wuchshülle mit den angegebenen Kostensätzen für Berg-Ahorn. Da keine Mischkultur abgebildet werden kann, geht das Programm davon aus, dass es sich bei den insgesamt 600 Pflanzen um die gleiche Baumart handelt. Beim zweiten Variantenpaar handelt es sich ebenfalls um einen Zaun sowie um eine Wuchshülle, wobei die angegebenen Kostensätze für Vogel-Kirsche eingegeben wurden. In diesem Fall bestünde die gesamte Kultur aus der Baumart Vogel-Kirsche. Weil die Pflanzenkosten für Berg-Ahorn gegenüber der Vogel-Kirsche geringer sind, wurden mit dem ersten Vergleichspaar die minimalen Kosten und mit dem zweiten Vergleichspaar die maximalen Kosten für die Kultur abgebildet. Daher steht hier bereits fest, dass sich die Entscheidung im Kostenrahmen zwischen diesen zwei Funktionsgeraden bewegen wird.

Darüber hinaus wurde ein drittes Schutzpaar angelegt, das auch aus einem Zaun und einem Wuchshüllenelement besteht. Es wurden hier jedoch die Kosten gewichtet, indem jeweils die Anzahl der Pflanzen einer Baumart berücksichtigt wur-

den. Das heißt, dass in diesem Fall die realen Kulturkosten für 480 Berg-Ahorne und 120 Vogel-Kirschen eingegeben wurden. Folglich ist verständlich, warum die Funktion des dritten Variantenpaares zwischen den Funktionsgraphen der ersten beiden Paare liegt. Es soll an dieser Stelle jedoch darauf hingewiesen werden, dass die Berechnung des gewichteten Mittels aus den angegebenen Kosten außerhalb des *Wuchshüllenrechners* stattgefunden hat. Außerdem wird programmiert bei den insgesamt 600 Pflanzen davon ausgegangen, dass es sich weiterhin um die gleiche Baumart handelt.

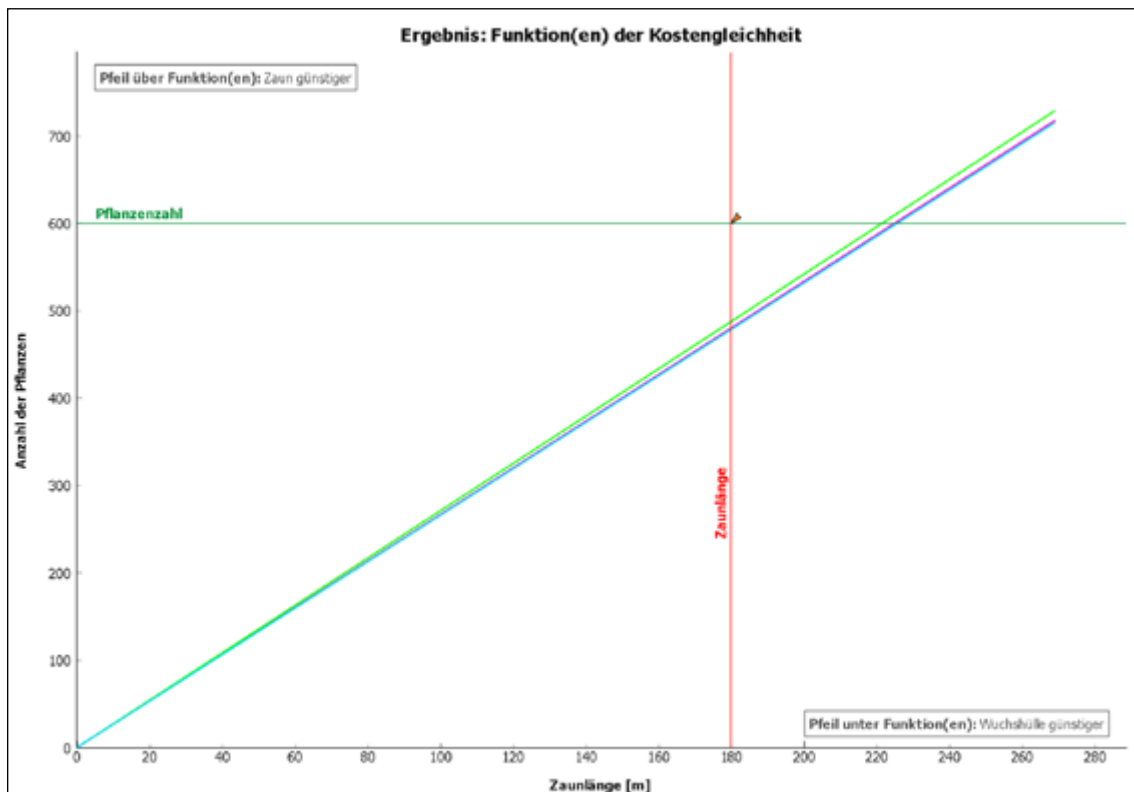


Abbildung 23: Mittels *Exportieren* erstellte Ergebnisgrafik des Vergleichs

Die Abbildung 23 zeigt das Ergebnis aus dem Vergleich der beschriebenen Variantenpaare als Funktionen der Kostengleichheit in der Liniendarstellung. Die Abszissenachse ist bis zu einer maximalen Zaunlänge von 280 Metern skaliert, wohingegen die Ordinatenachse bis zu einer Anzahl von 700 Pflanzen skaliert. Aus den jeweils drei eingegebenen Varianten für Zaun und Wuchshülle wurden drei Ergebnisse berechnet, die als Funktionsgraphen dargestellt sind. Der *untere blaue Graph* zeigt das Ergebnis für die Berg-Ahorn-Monokultur und analog zeigt der *obere grüne Graph* das Ergebnis für die Vogel-Kirschen-Monokultur. Dazwischen ist der Funktionsgraph für das Ergebnis der Kalkulation aus den manuell gewich-

teten Kostenwerten der Mischkultur in *pinkter Farbe* dargestellt. Zudem wurden die Zaunlänge bei 180 Metern sowie die Pflanzenzahl bei 600 Stück eingezeichnet, weshalb der resultierende Schnittpunkt, der durch den Pfeil gezeigt wird, oberhalb aller Funktionsgraphen liegt. Daher empfiehlt der *Wuchshüllenrechner* für die im Szenario beschriebene Beispielkultur in jedem Fall die Verwendung des Zaunes.

Entscheidend ist aber die Lage der Funktionsgraphen zueinander. Bei okularer Betrachtung des Diagramms liegen anfangs alle Graphen eng zusammen. Erst mit zunehmender Zaunlänge und steigender Pflanzenzahl zeigt sich die geringe Abweichung zueinander. Ausgehend von der *blauen Funktionsgeraden* für die Berg-Ahorn-Monokultur, die in dieser Konstellation eine untere Grenze durch eine geringere positive Preisdifferenz zwischen den beiden Berg-Ahorn-Sortimenten in Verbindung mit dem gegenüber der Vogel-Kirsche günstigeren Preis im kleineren Sortiment als *Startwert* bildet, liegen die beiden anderen Funktionsgraphen darüber. Daher verschiebt sich in diesem Fall durch die insgesamt teureren Vogel-Kirschen mit dem *oberen grünen Graphen* das Verhältnis zugunsten der Wuchshülle, denn je größer die Fläche unterhalb der Geraden ist, desto größer ist der betriebswirtschaftliche Anwendungsbereich der Wuchshülle. Dieses Verhältnis wird beispielsweise beeinflusst, sobald die Pflanzenkosten in der Wuchshülle (*Startwert*) gegenüber gleich bleibenden Pflanzenkosten im Zaun günstiger werden und somit die positive Preisdifferenz zunimmt. Auch dann verschiebt sich das Verhältnis zugunsten der Wuchshülle. Es muss darauf hingewiesen werden, dass die Preisdifferenz eine entscheidende Rolle spielt und viele weitere Konstellationen möglich sind, die an dieser Stelle nicht weiter beleuchtet werden konnten. Insbesondere wurden hier einzig die Pflanzenkosten bei gleichem Pflanzverband betrachtet. Selbstverständlich sind die Materialkosten und beispielsweise der Abbau und die Entsorgung der Wuchshülle ebenso zu betrachten.

Dass die realen Kostenwerte und damit die *pinke Funktionsgerade* näher an der Berg-Ahorn-Variante liegen, ist so zu erklären, dass die 480 Berg-Ahorne im gewichteten Kostenwert einen größeren Anteil haben als die 120 Vogel-Kirschen. Wäre der Anteil der Vogel-Kirschen höher, würde sich die Gerade in Richtung des *grünen Graphen* verschieben.

Allein aus dem Diagramm kann abgelesen werden, dass die Entscheidungsempfehlung im vorliegenden Beispiel aufgrund der eng zueinander liegenden Funktionsgraphen nur minimal durch den Kostenunterschied zwischen Berg-Ahorn und Vogel-Kirsche beeinflusst wird. Erst bei größeren Flächen, kann der aufgedeckte Fehler ins Gewicht fallen. Was dies nun in Zahlen bedeutet, beschreiben die Tabelle 5, Tabelle 6 und die Tabelle 7.

Kulturpflanze	Steigung	Pflanzenzahl (180) <i>(tatsächlich: 180 Lfm.)</i>	Pflanzenzahl (280) <i>(max. 280 Lfm.)</i>
Berg-Ahorn	2,658	478	744
Vogel-Kirsche	2,709	487	758
Mischkultur	2,667	480	746

Tabelle 5: Absolute Zahlenwerte für die Steigung sowie für die Pflanzenzahlen bei 180 Lfm. (tatsächlich gesucht) und bei 280 Lfm. in Bezug auf die Baumarten Berg-Ahorn und Vogel-Kirsche und eine entsprechende Mischkultur

Kulturpflanze	Steigung <i>(Differenz zur Mischkultur)</i>	Pflanzenzahl (180) <i>(Differenz zur Mischkultur)</i>	Prozent (Pflanzen) <i>(Differenz zur Mischkultur)</i>
Berg-Ahorn	- 0,009	- 2	- 0,42 %
Vogel-Kirsche	+ 0,042	+ 7	+ 1,46 %

Tabelle 6: Differenzwerte von Berg-Ahorn und Vogel-Kirsche bezogen auf die Mischkultur

Kulturpflanze <i>(Differenz zwischen)</i>	Steigung <i>(Differenz)</i>	Pflanzenzahl (180) <i>(Differenz)</i>	Prozent (Pflanzen) <i>(Differenz zur Mischkultur)</i>
Kir - BAh	0,051	9	1,88 %

Tabelle 7: Differenzwerte zwischen Berg-Ahorn und Vogel-Kirsche in Relation zur Mischkultur

Die Differenzen der Steigungswerte zeigen damit, dass die Funktionen relativ eng zueinander liegen. Interessant ist der Unterschied bei der gesuchten Pflanzenzahl für 180 Meter Zaunlänge. Bei der Berg-Ahorn-Variante liegt die Pflanzenzahl für diese Zaunlänge bei 478 (abgerundet) Stück, wohingegen die Vogel-Kirschen-Variante eine Pflanzenzahl von 487 (abgerundet) Stück aufweist. Die Differenz entspricht 9 Individuen bei einer Zaunlänge von 180 Metern und ist damit in Anbetracht der großen Pflanzenzahl verhältnismäßig gering. Ausgehend von der realen Mischkultur liegt der Punkt der Kostengleichheit bei 180 Metern Zaunlänge und 480 Pflanzen (abgerundet). Prozentual bezogen auf 480 Pflanzen befindet sich die gesamte Differenz zwischen beiden Funktionsgeraden mit 9 Stück im Be-

reich von 1,88 % (aufgerundet). Absolut gesehen, nimmt jedoch die Differenz der Pflanzen bei großen Zaunlängen zu, weshalb die Entscheidungsempfehlung nicht mehr genau sein wird. Im Rahmen dieser Arbeit wird der Fehler dennoch als gering eingeschätzt, weil der Aufwand zur Implementierung einer Lösung für genaue Entscheidungsempfehlungen in Mischkulturen im Verhältnis zu hoch ist. Hinzu kommt, dass in der Beispielkalkulation von 600 (480/120) Pflanzen ausgegangen wurde und so der Preis für die Verpackungseinheit von 100 Stück angenommen werden musste. Bei einer höheren Anzahl von Individuen kann mit der Verpackungseinheit von 1.000 Stück gerechnet werden und somit ist ein günstigeres Preisverhältnis möglich. In der Folge liegen die Funktionen vermutlich etwas enger zusammen, weil auch die absoluten Preisdifferenzen geringer sind.

5.2 Verständnisproblematik

5.2.1 Trennung von Zaun und Wuchshülle

Schutztyp	Name	Baumart	Schutzbeschreibung
<input checked="" type="checkbox"/> Zaun	Z5	sLb	Pfostenzaun (1,60 m) mit 480 Berg-Ahorn und 120 Vogel-Kirschen 1+1 (50-80 cm)
<input checked="" type="checkbox"/> Wuchshülle	WH6	sLb	Tubex Ventex (1,20 m) mit Entsorgung und 480 Berg-Ahorn und 120 Vogel-Kirschen 1+0 (30-50 cm)
<input checked="" type="checkbox"/> Zaun	Z3	Kir	Pfostenzaun (1,60 m) mit 600 Vogel-Kirschen 1+1 (50-80 cm)
<input checked="" type="checkbox"/> Wuchshülle	WH4	Kir	Tubex Ventex (1,20 m) mit Entsorgung und 600 Vogel-Kirschen 1+0 (30-50 cm)
<input checked="" type="checkbox"/> Zaun	Z1	BAh	Pfostenzaun (1,60 m) mit 600 Berg-Ahorn 1+1 (50-80 cm)
<input checked="" type="checkbox"/> Wuchshülle	WH2	BAh	Tubex Ventex (1,20 m) mit Entsorgung und 600 Berg-Ahorn 1+0 (30-50 cm)

Abbildung 24: Beispiel für die getrennte Auflistung von Zaun- und Wuchshüllenelementen

Bereits bei der Konzeption der Anwendung wurde festgestellt, dass ein Spannungsfeld aufgrund der *ungleichen Datensätze* für Zaun- und Wuchshüllenelemente besteht. Daher wurde im Programmablauf versucht, die Daten der jeweiligen Elemente möglichst einheitlich abzufragen. Bei unzähligen Gesprächen mit Forstpraktikern stellte sich heraus, dass es nicht immer ganz verständlich ist, warum die Zaunvariante getrennt von der Wuchshüllenvariante angelegt wird. Zudem gab es einige Vorschläge, die Kostenwerte der Pflanze aus dem jeweils zuvor angelegten Schutzelement zu übernehmen.

Dass die Zaunvariante vom Schutz mit der Wuchshülle getrennt eingegeben werden muss, hat verschiedene Hintergründe. Das Programm sieht grundsätzlich vor, dass mehrere Wuchshüllenelemente mit einem Zaunschutz der gleichen Baumart verglichen werden können. Dies bietet beispielsweise den Vorteil, eine Wuchshüllenvariante mit Entsorgungskosten sowie eine zweite Wuchshüllenvariante ohne Entsorgungskosten mit einem Zaun zu vergleichen. Indirekt werden so auch die Wuchshüllen selbst verglichen. Würde jedoch die gleichzeitige Abfrage der Kostenwerte für den Zaun- und den Wuchshüllenschutz in einem Dialog zusammengefasst werden, wären beide Schutztypen miteinander verbunden und es gäbe keine Möglichkeit, mehrere Wuchshüllenelemente anzulegen. Deshalb bietet die getrennte Erfassung der Schutzvarianten für die Anwender wesentlich mehr Flexibilität. Hinzu kommt, dass es im Programmablauf einfacher erscheint, die Wuchshüllen getrennt vom Zaun zu betrachten. Ebenfalls bietet die Oberfläche der Anwendung wesentlich mehr Spielraum, wenn es um das Ändern oder Entfernen einer Variante geht. Haben die Benutzer beispielsweise in der *Listensicht* der Datenerfassung eine Wuchshüllenvariante ausgewählt und möchten diese entfernen, hat dies keinen Einfluss auf das Zaunelement, da die Hülle ein untergeordnetes Element darstellt. Umgekehrt bedeutet jedoch das Entfernen einer übergeordneten Zaunvariante, dass ebenfalls alle untergeordneten Elemente gelöscht werden.

5.2.2 Keine Übernahme der Kostenwerte

Außerdem ist die Anwendung so konzipiert, dass beim Anlegen eines Wuchshüllenschutzes nicht automatisch die Kostenwerte eines gegebenenfalls zuvor erstellten Zaunelementes übernommen werden. Der Gedanke hierbei ist, dass bei den beiden Schutzmöglichkeiten verschiedene Sortimente der gleichen Baumart verwendet werden. Folglich müssen die Stückkosten sowie mindestens die Pflanzungskosten jeweils unterschiedlich sein. Darüber hinaus müssen je nach Schutztyp andere Kulturvorbereitungs- und -sicherungskosten berücksichtigt werden. Würde das Programm die Kostenwerte automatisch als Vorschlagswerte für die nächste Variante mit der gleichen Baumart übernehmen, hätten die Anwender weniger Aufwand bei der Eingabe. Dies könnte jedoch zu dem Nachteil führen,

Abbildung 25: Abgefragte Kostenwerte für die Pflanze und die Pflanzung

dass sich die Benutzer nicht mehr mit den *Kosten für die Pflanze und die Pflanzung* beschäftigt. Die automatische Übernahme der Kostenwerte wäre nur dann sinnvoll integriert, wenn die Anwender gefragt würden, ob sie wirklich fortfahren möchten, sobald die eingegebenen Werte für mehrere Schutzvarianten gleich sind. Doch im Rahmen der Bachelorarbeit ist es nicht möglich, diese Plausibilitätsprüfung zu implementieren. Daher soll die bisherige Dateneingabe zusätzlich als Gedankenstütze für die Benutzer dienen. Zudem ist sie flexibler.

5.2.3 Einheiten der Kostenwerte und Eingabehilfen

Im Bereich des Erfassungsfensters *Schutz bearbeiten* werden einige Werte als Stückkosten erfasst, so wie es in Abbildung 25 zu sehen ist. Es ist jedoch häufig realistisch, dass die Anwender die stückbezogenen Kosten gar nicht kennen und deshalb Probleme bei der Eingabe haben. Oftmals muss zumindest von Kostenwerten je Hektar ausgegangen werden und andere Anwender kennen womöglich nur die Gesamtkosten und die absolute Kulturfläche sowie die konkrete Zaunlänge und Pflanzenzahl. In solchen Fällen ist die Eingabe im Erfassungsfenster *Schutz bearbeiten* auf den ersten Blick schwierig oder gar unmöglich. Es wäre jedoch keinesfalls sinnvoll, alle Alternativeingaben im Dialog *Schutz bearbeiten* einzufügen. Damit die Anwender die gewünschten Eingabewerte trotzdem nicht händisch errechnen müssen, wurde im Programm für bestimmte Felder eine *Umrechnungshilfe* hinterlegt, die jeweils mithilfe der gleichnamigen Schaltfläche erreichbar ist. Diese Umrechnungshilfe bietet sicher in einigen Fällen einen Vorteil. Beispielweise kann für das Eingabefeld *Anzahl der Pflanzen* schnell der passen-

de Wert aus der Flächengröße und dem festgelegten Pflanzverband berechnet werden. Andererseits muss jedoch darauf hingewiesen werden, dass die entsprechenden Dialoge zur Umrechnung rudimentär implementiert wurden und deshalb nur einen Teil der üblichen Kalkulationen abdecken können. Es war hier ebenfalls im Rahmen der Bachelorarbeit nicht möglich, alle forstlich relevanten Umrechnungen abzubilden.

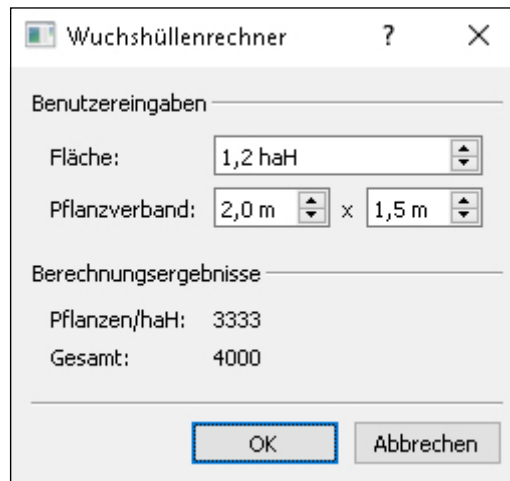


Abbildung 26: Umrechnungshilfe für die *Anzahl der Pflanzen* in Abhängigkeit der Flächengröße und des Pflanzverbandes

Bei vielen Eingabefeldern werden die Anwender zusätzlich vom Programm unterstützt. Bereits beim Feldnamen wurde darauf geachtet, einen prägnanten Text zu wählen. Grundsätzlich hängt es jedoch vom Verständnis der Benutzer ab, ob die Absichten des Entwicklers einleuchtend sind. Aus diesem Grund wurden bei einigen Eingabefeldern zusätzlich die sogenannten *ToolTips* in Form eines Informationssymbols installiert. Bewegen die Anwender den Cursor der Maus über ein solches Symbol, erfahren sie eine kurze Beschreibung und Hilfe zur Dateneingabe im entsprechenden Feld. Das heißt, dass diese Hilfestellung speziell dazu dient, den Benutzern einen Hinweis zu geben, welche Eingaben von ihnen erwartet werden. Diese ToolTips wurden zu allen Eingabefeldern hinzugefügt, sobald der Autor dieser Arbeit die Eingabe als schwierig erachtete.

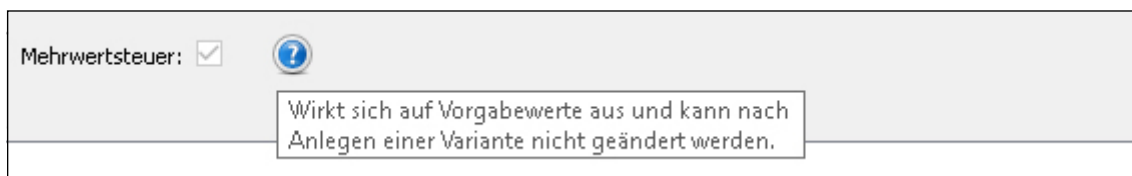


Abbildung 27: Berücksichtigung der Mehrwertsteuer im Hauptfenster mit aktivem ToolTip am Informationssymbol

5.2.4 Darstellung des Ergebnisses

Geht es um die Verständnisfrage, erweist sich speziell die Diagramm- und Ergebnisansicht als schwierig. Offensichtlich wird der *Wuchshüllenrechner* oftmals nicht als Entscheidungshilfe verstanden, sondern es wird vielmehr eine absolute Kostenkalkulation erwartet. In erster Linie ist eine reale Kostenkalkulation nicht die Intention des *Wuchshüllenrechners*, obwohl dies aufgrund der Eingabedaten durchaus möglich ist. Dennoch gab es bereits bei der Konzeption des Programmes die Überlegung, neben der Flächen- und Liniendarstellung die sogenannte *Kostendarstellung* einzuführen. Hier könnte ein Vergleich der absoluten Kosten in Abhängigkeit der Variablen *Zaunlänge* und *Pflanzenzahl* in einem Balkendiagramm durchgeführt werden. Doch auch hier ist es im Rahmen der Bachelorarbeit nicht möglich, eine solche Kostendarstellung *fehlerfrei* zu implementieren. Somit wurde davon abgesehen.

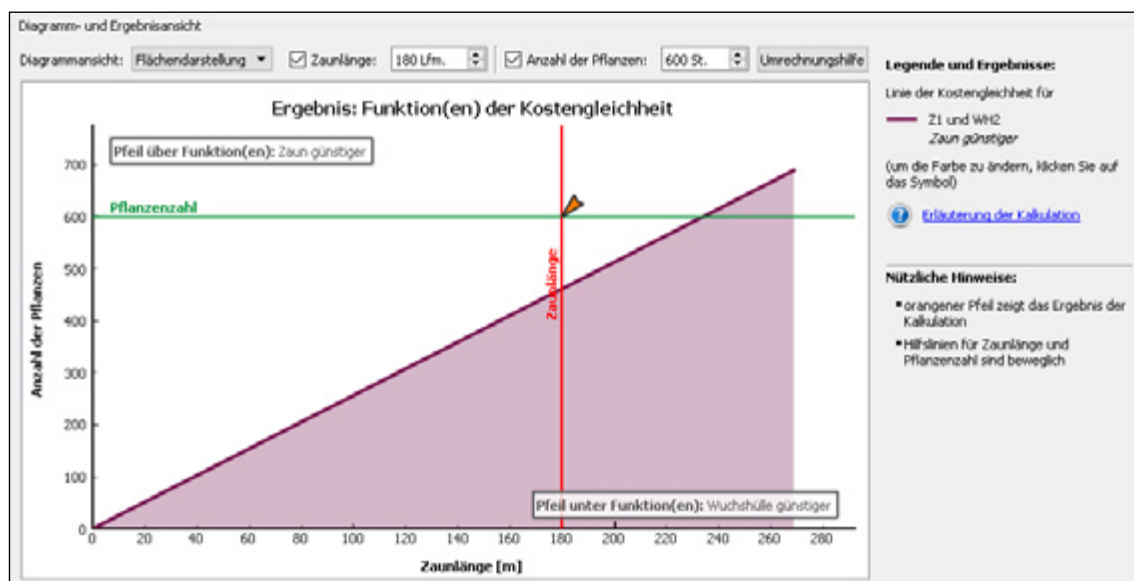


Abbildung 28: Aufbereitete Diagramm- und Ergebnisansicht

Weiterhin ist gerade die Abbildung der Kostengleichheitsfunktionen schwierig. Dabei fällt auf, dass es den Anwendern nicht verständlich ist, dass jeweils eine Wuchshüllenvariante mit einem Zaunelement verglichen wird. Daher wird die Bedeutung der Geraden als Funktion der Kostengleichheit nicht erkannt. Folglich verstehen viele Benutzer die Abbildung nicht sofort. Dies kann einerseits dazu führen, dass sich der interessierte Anwender intensiver mit dem dargestellten Ergebnis beschäftigt und damit die Entscheidung eher hinterfragt. Andererseits

nutzen weniger interessierte Anwender den Wuchshüllenrechner möglicherweise gar nicht. Hinzu kommt, dass der Schnittpunkt aus der *Zaunlänge* und der *Pflanzenzahl*, der durch die beiden Hilfslinien dargestellt wird, nicht als Ergebnis der Entscheidungsempfehlung betrachtet wird.

Daher wurden einige zusätzliche Elemente in der Diagrammansicht installiert, um nicht zuletzt eine Beschreibung für die einzelnen Programmfunktionen zu bieten. Dahinter steht die Absicht, die Benutzer beim Verstehen der Darstellung zu unterstützen. Bereits in der Überschrift findet sich der Hinweis, dass alle Ergebnisse der Kalkulation als Funktionen der Kostengleichheit dargestellt werden. Dies gilt für die Flächen- und Liniendarstellung. Daneben wird jeweils darauf hingewiesen, dass sich die Zaunvariante lohnt, wenn der Schnittpunkt oberhalb der Funktion liegt und analog lohnt sich die Wuchshüllenvariante, wenn der Schnittpunkt unterhalb der Funktion liegt. Allein diese Hinweise führen zu einem deutlich besseren Verständnis. Zusätzlich wurde der Schnittpunkt mit einem *orangenen Pfeil* markiert, um möglichst auffällig zu wirken. Damit ist weiterhin ein besserer Bezug verbunden, da nicht alle Anwender etwas mit dem Begriff *Schnittpunkt* anfangen konnten. Der Pfeil ist umso eindeutiger. Und schließlich entstand die Legende im Infobereich, die anhand der Linienfarbe erklärt, dass ein Vergleich einer Zaunvariante mit dem betreffenden Wuchshüllenelement stattgefunden hat. Innerhalb der Legende wird außerdem das Ergebnis in Textform eingeblendet, um unabhängig vom Schnittpunkt das Ergebnis benutzerfreundlich und eindeutig anzuzeigen.

Außerdem ist es in der Diagrammansicht sinnvoll, zuerst die Flächendarstellung zu wählen. Zum einen kann sie nur eine Funktion abbilden und bleibt somit übersichtlich. Zum anderen soll sich mit der eingefärbten Fläche unterhalb des Graphen der betriebswirtschaftliche Einsatzbereich der Wuchshülle vom Anwendungsbereich des Zaunes abheben. Demgegenüber zeigt die Liniensicht ihre Stärke, sobald es um die Zeichnung mehrere Funktionsgraphen geht. Ein Vergleich zwischen den einzelnen Varianten ist folglich einfacher möglich, was auch die Beispielrechnung im Kapitel 5.1 anschaulich beweist.

Fraglich bleibt, ob es den Benutzern möglich sein soll, die Hilfslinien für die *Zaunlänge* und die *Pflanzenzahl* im Diagramm zu verschieben. Rein waldbau-

lich betrachtet, muss davon ausgegangen werden, dass sowohl die *Zaunlänge* als auch die *Anzahl der Pflanzen* fixe Werte darstellen, die sich nicht mehr ändern. Die Kulturläche wird nicht plötzlich größer oder kleiner werden, weshalb sich der Umfang nicht verändert. Ähnlich unrealistisch scheint es, dass ein schon festgelegter Pflanzverband geändert wird, nur um einen bevorzugten Schutztyp betriebswirtschaftlich rechtfertigen zu können. Dies wäre nicht im Sinne der Entscheidungshilfe nach HAMMER. Dennoch bietet das Verschieben der Hilfslinien einige *Analysemöglichkeiten*, insbesondere in Grenzfällen, wenn sich der Pfeil bei gegebener Zaunlänge nah am Punkt der Kostengleichheit bewegt. Besonders in Einzelfallentscheidungen kann die beschriebene Programmfunktion hilfreich sein. Und abschließend helfen die beiden Linien beim *spielerischen Finden* des richtigen Schnittpunktes. Damit erhalten die Anwender größtmögliche Flexibilität.

Insgesamt wurde besonders im Bereich der Diagramm- und Ergebnisansicht viel Zeit investiert, um eine optimale und zugleich ansprechende Ergebnisdarstellung zu gewährleisten. Ob dies nun die Anwender letztlich tatsächlich unterstützt, konnte im Rahmen der Bachelorarbeit nur zum Teil evaluiert werden und wird sich weiterhin zeigen.

5.3 Weitere Schutzverfahren

Das Ziel der Arbeit ist die digitale Umsetzung der Entscheidungshilfe nach dem Beispiel des Fachbeitrages von HAMMER. Deshalb unterstützt der *Wuchshüllenrechner* in seiner derzeitigen Implementierung ausschließlich den Vergleich zwischen Zaun und Einzelschutz in Form der Wuchshülle. Letztlich findet dieser Vergleich statt, indem die Kosten der jeweiligen Schutzvarianten mathematisch miteinander gleichgesetzt werden, sodass in Abhängigkeit der Zaunlänge sowie der Pflanzenzahl die Funktion der Kostengleichheit entsteht.

Wie bereits aus der Evaluation hervorgegangen ist, wäre es durchaus wünschenswert, andere Schutzverfahren zu vergleichen. Es wäre dabei denkbar, beispielsweise *chemische Verfahren* in der Auswahl zu berücksichtigen. Grundsätzlich wurde die Anwendung so entwickelt, dass die Liste der bestehenden Verfahren auf einfache Art und Weise erweitert werden kann. Der Algorithmus sieht jedoch

in der aktuellen Implementierung vor, dass alle Einzelschutzverfahren mit einem Zaunelement mathematisch gleichgesetzt werden müssen. Auch in der *baumartig organisierten Liste* der Datenerfassung wird den Benutzern diese Tatsache suggeriert, da alle Wuchshüllenelemente immer einem Zaunschutzes untergeordnet werden müssen. Insofern würde mit dieser Implementierung kein Vergleich unter den Einzelschutzvarianten stattfinden. Für eine solche Kalkulation müsste der *Wuchshüllenrechner* entsprechend technisch erweitert werden. An diesem Punkt tritt zusätzlich die Benutzerfreundlichkeit ins Bewusstsein. Je mehr Funktionalität die Anwendung erhält, desto schneller verschiebt sich die Tendenz in Richtung *Unübersichtlichkeit* und *Überladung* der Programmoberfläche.

Im Hinblick auf die Erweiterbarkeit des *Wuchshüllenrechners* kann zusammenfassend festgestellt werden, dass die Weiterentwicklung und Integration von weiteren Schutzverfahren in einem gewissen Rahmen problemlos möglich ist, da dieser Punkt an vielen Stellen bereits programmintern berücksichtigt wurde. Dieser Vorteil wird erst möglich, weil im Quelltext sehr viele dynamische und wenige statische Elemente vorhanden sind.

5.4 Waldbauliche Faktoren

Auf die Entscheidung, ob ein Zaun gebaut oder ein Einzelschutz verwendet werden soll, haben viele Faktoren einen Einfluss. Das Computerprogramm *Wuchshüllenrechner* arbeitet speziell die ökonomische Fragestellung dieses Konfliktes ab, indem betriebswirtschaftliche Kostensätze Eingang in die Vergleichskalkulation finden. Daher fällt es auf, dass bestimmte waldbauliche Faktoren unbetrachtet bleiben.

Oftmals spielt die *Wilddichte* auf der Fläche eine weitere Rolle im Bezug darauf, ob ein Zaunschutzes gebaut werden soll, weil in der Folge die für das Wild zur Verfügung stehende Äsungsfläche verkleinert wird. Im Prinzip ist schutzbedingt und unabhängig von der Einzelpflanze die gesamte Kulturfläche unzugänglich. Beim Schutz durch Wuchshüllen und allgemein bei Einzelschutzverfahren ist die Situation anders. Denn tatsächlich wird nur die einzelne, verbissgefährdete Pflanze behandelt, was dazu führt, dass sich das Wild weiterhin auf einem Großteil der

Fläche bewegen kann. Folglich verteilt sich der Verbissdruck auf den Flächen und wird nicht durch die strikte Abgrenzung der Kultur in anderen Beständen erhöht.

Sicherlich muss der geschilderte Umstand in die Entscheidung mit einfließen. Da jedoch bereits im Fachbeitrag von HAMMER und somit auch beim *Wuchshüllenrechner* der Schwerpunkt auf einen betriebswirtschaftlichen Vergleich gelegt wurde, wird dieser Einflussfaktor nicht abgefragt oder betrachtet. Einerseits ist es fraglich, ob es überhaupt Sinn macht, diesen Faktor in die Kalkulation mit einfließen zu lassen, denn an dieser Stelle ist in Anbetracht der oben beschriebenen Situation vielmehr das waldbauliche Geschick gefragt, um die passende Entscheidung zu treffen. Darüber hinaus könnte das Kriterium *Wilddichte* nur geschätzt werden und entsprechend ungenau die Berechnung beeinflussen. Damit würde der durch den *Wuchshüllenrechner* kalkulierte Empfehlung der Vorteil als Grundlage für eine rationale Entscheidung entzogen.



Abbildung 29: Ein nicht schutzbedürftiger Fichten-Naturverjüngungsvorrat
Bildquelle: Landesforsten.RLP.de / Hansen/Lamour

In ähnlicher Weise muss mit dem Einflussfaktor *Naturverjüngung* verfahren werden. Hier ist es besonders interessant, welche Baumart auf der Fläche bereits vorhanden ist. Denn wird beispielsweise eine nicht schutzbedürftige Baumart durch eine weitere Art ergänzt, die dringend geschützt werden muss, hängt es von der Anzahl der *eingebrachten* Pflanzen ab, welcher Schutz verwendet werden soll. Die Situation ist anders, wenn die verjüngte Baumart ebenfalls schutzbedürftig ist. In diesem Fall kommt es zusätzlich speziell auf den Verjüngungsvorrat an, um die richtige Entscheidung zu treffen. Damit wird es jedoch schwierig, die tatsächliche Situation vor Ort genau einzuschätzen, um diese im Programm richtig darstellen zu können. Vielmehr ist es eine komplexe Thematik, die nicht einfach abgebildet werden kann.

Allgemein ist der *Wuchshüllenrechner* dazu gedacht, eine Empfehlung für eine rationale Entscheidung zu geben, wobei das Kalkulationsergebnis auf ermittelbaren betriebswirtschaftlichen Kennzahlen beruht. Deshalb wurden die beiden oben exemplarisch genannten waldbaulichen Kriterien nicht betrachtet.

5.5 Verwendung von Python 3

Dass die Applikation schließlich in der Programmiersprache *Python 3* geschrieben wurde, bietet einige Vorteile gegenüber einer in Excel mithilfe von VBA entwickelten Software. In bestimmten, Sicherheitsrichtlinien abhängigen Konstellationen des Betriebssystems kann eine native Anwendung, wie es beim *Wuchshüllenrechner* der Fall ist, aber auch zu Problemen führen. Dies konnte so auch bei einigen Testern festgestellt werden, die aufgrund des Virenschutzes das Programm gar nicht erst starten konnten.

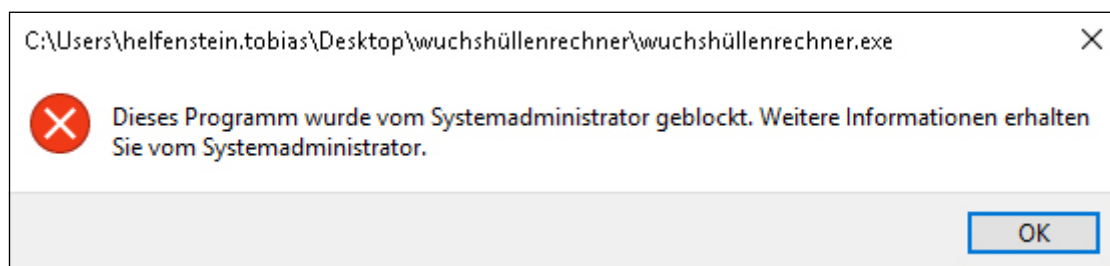


Abbildung 30: Durch eine Sicherheitsrichtlinie des Betriebssystems wird der Start des Wuchshüllenrechners verhindert

Viele Unternehmen sind wegen der angespannten Sicherheitssituation im Bereich der Informationstechnik entsprechend sensibilisiert, weshalb die Liste der ausführbaren Programme oftmals recht restriktiv gewählt ist. Soll in solchen Umgebungen nun der *Wuchshüllenrechner* gestartet werden, kommt es zur Fehlermeldung, dass es nicht erlaubt sei, das Programm auszuführen. Dann muss der Administrator die Anwendung in jedem Fall erst freigeben.

Im Rahmen dieser Arbeit und speziell bezogen auf die Evaluation des Projektes führt das beschriebene Verhalten des Betriebssystems natürlich nicht zum Erfolg. Tester, die sich den *Wuchshüllenrechner* anschauen wollten, sind im besten Fall auf ein privates System ausgewichen. In einigen Fällen gab es dann jedoch keine weitere Rückmeldung mehr. Es muss dennoch darauf hingewiesen werden, dass es möglich und gleichermaßen verständlich ist, dass unbekannte Anwendungen wie der *Wuchshüllenrechner* vom System blockiert werden. Ähnliche Phänomene können bei Exceldateien auftreten, die in VBA programmierte Makros enthalten. Dort ist ebenfalls die Sicherheitsrichtlinie häufig so restriktiv, dass nur zugelassene Exceldateien überhaupt geladen werden dürfen. Auf einigen Mailservern dürfen diese Exceldateien teilweise nicht verschickt und gar nicht empfangen werden.

Daher ist es problematisch, den *Wuchshüllenrechner* als ausführbare Datei zu versenden. Abhilfe schafft ein komprimierter Ordner, das sogenannte *ZIP-Format*, welches auch bei der Evaluation in dieser Arbeit eingesetzt wurde. Es muss jedoch vor dem Start der Anwendung erst der gesamte Ordner auf die Festplatte entpackt werden, womit ein zusätzlicher Arbeitsschritt für die Tester notwendig war. Das Entpacken des ZIP komprimierten Verzeichnisses ist nicht immer ganz einfach und hängt von der bei den Anwendern installierten Software ab. Eine andere Möglichkeit bietet die Installationsdatei *setup.exe*, welche die gesamten Programminhalte enthält. Mit dem Ausführen werden die Inhalte automatisch an den vorgegebenen Ort kopiert und die Anwendung ist startbereit. In diesem Zusammenhang sind die Restriktionen in vielen Fällen noch größer und der *allgemeine Benutzer* kann ohne Rechte des Administrators den Installationsprozess überhaupt nicht anstoßen.

Dies zeigt, welche Konflikte entstehen, wenn Software auf modernen Systemen zum Testen verteilt werden soll. Aus diesen Gründen darf der *Wuchshüllenrechner* nicht abgewertet werden, da es von einem Programm vollkommen unabhängig sein kann, ob die Sicherheitsrichtlinien greifen oder nicht. Oft hat das Umgehen dieser Richtlinien eher eine gegenteilige Wirkung. Dennoch konnte der *Wuchshüllenrechner* trotz dieser vielen Schwierigkeiten auf den meisten Systemen anstandslos gestartet und getestet werden.

6 Schlussfolgerungen und Ausblick

6.1 Abschließende Bewertung

Am Ende dieser Arbeit lässt sich schlussfolgern, dass aus der ursprünglichen Idee von HAMMER eine solide Computeranwendung geworden ist. In seinem Fachbeitrag war es insbesondere die Intention, den Forstpraktikern ein simples Werkzeug an die Hand zu geben, um eine rationale Entscheidung zwischen Zaun- oder Einzelschutz treffen zu können. Speziell die Auswahl des Schutzverfahrens nach Gefühl und Ermessen soll damit hinterfragt und die tatsächliche Entscheidung mit betriebswirtschaftlichen Fakten untermauert werden. War es anfangs noch so gedacht, dass sich die Anwender eine eigene Kalkulation auf der Grundlage des Artikels entweder händisch oder mithilfe von Excel erarbeiten, so bietet der *Wuchshüllenrechner* nun eine allgemeingültige und computerbasierte Lösung für die beschriebene, komplexe ökonomische Fragestellung. Schließlich wurde der Ansatz von HAMMER digitalisiert und für einen breiten Benutzerkreis zugänglich gemacht.

Kern der Anwendung ist der auf dem wissenschaftlichen Fachbeitrag basierende Rechenweg, welcher in einem Baumarten abhängigen Algorithmus realisiert wurde. Dabei werden viele Eingangsdaten von HAMMER als Eingabefelder im Programm übernommen und nur an einigen Stellen wurde die Entscheidungshilfe erweitert. Beispielsweise berücksichtigt die Anwendung zusätzlich den Kostenwert der *Kulturvorbereitung* in der Kalkulation. An anderer Stelle setzt der *Wuchshüllenrechner* zwingend voraus, dass nur Schutzverfahren mit der gleichen Baumart miteinander verglichen werden können. Dies ist eine wichtige Erweiterung, die bei einer händischen Berechnung wegfallen würde, in einem Programm aber abgefangen werden muss. Insgesamt führen die vielen Sortieraktionen sowie Interaktionen mit den Anwendern zu einer mächtigen Programmlogik.

Im Allgemeinen konnten die wesentlichen Funktionen für den Vergleich der Schutzverfahren im Programm abgebildet werden. Neben der getrennten Datenerfassung von Zaun- und Wuchshüllenschutz werden die Ergebnisse der Kalkulation und die davon abhängige Entscheidungsempfehlung in einer variablen sowie

flexiblen Diagrammansicht dargestellt. Darüber hinaus erlaubt die Diagrammansicht die Auswahl einer Flächen- und Liniendarstellung, um die Funktionen der Kostengleichheit optimal grafisch aufzubereiten. Zudem wird die Entscheidungsempfehlung in schriftlicher Form im Ergebnisbereich beschrieben. Abschließend können die Benutzer das gesamte Diagramm mit den aktuellen Werten problemlos als Grafik exportieren. Außerdem enthält der *Wuchshüllenrechner* noch eine ganze Reihe weiterer Anwendungsmöglichkeiten, die bereits ausführlich erläutert wurden. Deshalb soll die folgende Liste noch einmal den gesamten Funktionsumfang der Applikation zusammenfassen:

- Speichern und Öffnen von Projektmappen im XML-Dateiformat
- Kalkulationsbeispiele für Berg-Ahorn und Küsten-Douglasie
- Export der Diagrammansicht in ein Grafikformat (JPEG, PNG oder TIFF)
- Laufzeitübersetzung für die deutsche und englische Sprache
- Hilfedokument im PDF-Dateiformat in deutscher Sprache
- übersichtliche Darstellung der wissenschaftlichen Fachbeiträge
- Erfassung der Kopfdaten für Forstbetrieb, -revier, Revierleiter und Waldort
- Berücksichtigung der Mehrwertsteuer für regelbesteuerte Betriebe (19 %)
- getrennte Datenerfassung von Zaun- und Einzelschutz
- Umrechnungshilfen für verschiedene Eingabefelder
- Diagrammansicht mit Flächen- und Liniendarstellung
- Diagrammlegende mit Ergebnisbeschreibung
- variable *Zaunlänge* und *Pflanzenzahl* durch bewegliche Hilfslinien
- Plattformunabhängigkeit durch die Verwendung von *Python 3* und *PyQt 5*

Mit den aufgezählten Programmfunktionen wurden die in dieser Arbeit einleitend skizzierten Ziele vollumfänglich erreicht. Daneben konnten mit der *Plattformunabhängigkeit*, den Dateisystemfunktionen wie *Speichern*, *Öffnen* und *Exportieren* sowie der übersichtlichen Aufbereitung der *wissenschaftlichen Fachbeiträge* auch einige der optionalen Ziele berücksichtigt werden. Insofern ist mit dem *Wuchshüllenrechner* trotz des zeitlichen Rahmens eine sinnvolle Anwendung entstanden.

Schließlich richtet sich die Applikation besonders an Betriebsleitungen und Analysten in größeren Forstbetrieben, deren Verantwortung im Bereich des Controllings liegt. Dies lässt sich damit begründen, dass zum besseren Verständnis durchaus betriebswirtschaftliche Kenntnisse erforderlich sind, weil die geschilderte ökonometrische Fragestellung, welche HAMMER in seinem Fachbeitrag bearbeitet hat, zu einer komplexen Software führt. Trotzdem soll der *Wuchshüllenrechner* auch für private Waldbesitzer oder beispielsweise einzelne Revierleiter attraktiv sein, weshalb die grafische Oberfläche nach Möglichkeit benutzerfreundlich und selbsterklärend gestaltet wurde. Ob sich ein Anwenderkreis nach der Distribution der Software entwickeln wird, bleibt abzuwarten.

6.2 Bewertung der Evaluation

Aufgrund des zeitlich engen Rahmens machte die Evaluation der Anwendung nur einen kleinen Teil der Arbeit aus. Dennoch muss dieses Instrument besonders hervorgehoben werden, da viele Programmfunktionen erst in der Folge verbessert werden konnten und somit deutlich verständlicher geworden sind. Damit ist dieser Arbeitsabschnitt ein wichtiger Schritt gewesen, um den *Wuchshüllenrechner* weiterentwickeln zu können. Es geht besonders aus Kapitel 5.2 hervor, dass die Evaluation im Hinblick auf die Verständnisproblematik hilfreich war. Dabei ist es natürlich nicht immer möglich, alle Bedenken und Rückmeldungen der ausgewählten Testpersonen umzusetzen. Denn zum einen hat jeder Anwender durchaus andere Vorstellungen von einem Programm und speziell von einer Entscheidungshilfe, weshalb nicht alle Gedanken zielführend sind. Zum anderen mussten viele neue Ideen in den *Ausblick* übernommen werden, da eine Realisation im Rahmen der Arbeit nicht mehr möglich war. Dass der *Wuchshüllenrechner* deshalb keine komplette Kostenkalkulation für eine neue Kulturanlage mit unterschiedlichen Schutzverfahren bieten kann, ist daher verständlich und war nicht das Ziel dieser Arbeit.

Abschließend darf gesagt werden, dass die Rückmeldungen der ausgewählten Tester sehr informativ waren und den Autor bei dieser Arbeit enorm unterstützt haben. Einige haben den *Wuchshüllenrechner* kritisch hinterfragt und damit auf

eine andere Sichtweise der Problematik hingewiesen. Insbesondere in persönlichen Gesprächen konnte dabei auf viel Erfahrung der Forstpraktiker zurückgegriffen werden.

6.3 Ausblick

Bereits bei der Planung und auch während der Entwicklung des *Wuchshüllenrechners* gab es immer neue Gedanken, um das Programm zu erweitern. Insbesondere innerhalb der Evaluation konnten einige neue Ideen gewonnen werden. Doch, wie es bereits schon häufig angedeutet wurde, war es oftmals aus Zeitgründen nicht mehr möglich, diese Gedanken und Ideen im Rahmen der Arbeit zu realisieren. Es kommt hinzu, dass die vorhandene Funktionalität sicher nicht immer die optimale Problemlösung bietet, weshalb eine Weiterentwicklung der Software sinnvoll ist. Hierbei zeigt sich der Vorteil von *Open Source*, da jeder das Programm frei einsetzen und weiterentwickeln darf. Vielmehr ist es sogar gewünscht, die vorhandenen Funktionen zu erweitern. Damit ist die Zukunft der Software nicht zwingend an den Autor dieser Arbeit gebunden.

Einen Ausblick und einige Anregungen, welche Erweiterungen grundsätzlich noch integriert werden sollten, finden sich in der folgenden Liste. Diese wurde in zwei Kategorien unterteilt, wobei erstere die *Weiterentwicklung der vorhandenen Funktionen* beinhaltet. In der zweiten Kategorie werden grundsätzlich komplett *neue Funktionalitäten* gruppiert. Dabei gibt die Reihenfolge einen Hinweis zur Priorität. Die Art und Weise der Gruppierung steht darüber hinaus für den Zeitbedarf, da neue Funktionen einen entsprechenden höheren Aufwand bei der Implementierung bedeuten.

Erweiterung vorhandener Funktionen

- weitere Plausibilitätsprüfungen im Dialog *Schutz bearbeiten*
- Ergebnisfeil sollte ausgeblendet werden können
- Verschieben des Ergebnisfeiles führt zum Ändern der *Zaunlänge* sowie der *Pflanzenzahl*
- Bei veränderter Skalierung: Rückfrage, ob rückgängig

- Export in PDF-Datei mit Datentabelle und Diagrammansicht
- Export nach Excel, mindestens nach CSV (Comma-separated values)
- schreibgeschützte Dateien, Änderung nicht mehr möglich
- Aktualisierung der Bedienungsanleitung
- in Listenansicht ist automatisch nur eine Wuchshülle aktiv geschaltet
- Ergänzung der Einzelschutzverfahren
- Ergänzung weiterer Inhalte im Dialog *Grundlegende Fachbeiträge*
- zuletzt geöffnete Dateien
- gegebenenfalls Verbesserung der *ToolTips* und Hinweise
- Übersetzung in andere Sprachen – beispielsweise Japanisch
- Aktualisierung der vorhandenen Kostenstruktur für die Vorschlagswerte

Neue Funktionen

- Bereitstellung eines Installationsprogrammes (*setup.exe*)
- Druckfunktion für DIN A4 mit Datentabelle und Diagrammansicht
- eigenes Programmsymbol zur Wiedererkennung
- korrekte Berücksichtigung von Mischkulturen, besonders für absolute Kostenberechnung
- Balkendiagramm zur absoluten Kostenberechnung
- eigenes Datenformat: WRX (*.wrx) für XML-Dateien und WRB (*.wrb) für Binärdateien
- Öffnen per *Drag & Drop* aus dem Dateisystem (*Öffnen mit...*)
- Vergleich der Einzelschutzverfahren untereinander

Die Möglichkeit der Übersetzung in andere Sprachen soll an diesem Punkt explizit aufgegriffen werden. Mit den beiden Sprachen *Deutsch* und *Englisch* wurde die Anwendung bereits für zwei Sprachregionen zugänglich gemacht. Dabei wurde der Wunsch nach der Übersetzung schon mit dem Beginn der Programmierung für die Beschreibungen und Texte der Steuerelemente konsequent berücksichtigt.

Demzufolge ist es wünschenswert, den *Wuchshüllenrechner* in weiteren Sprachen zu veröffentlichen.

Außerdem sollte der *Wuchshüllenrechner* mithilfe einer Internetplattform einfach zugänglich sein, sodass Interessierte die Anwendung problemlos herunterladen und installieren können. Dazu dient in jedem Fall die Bereitstellung einer Installationsdatei in Form eines ausführbaren Setups. Ein Beispiel für eine ausgewählte Internetplattform stellt die Homepage der Hochschule selbst dar. Dort könnte auf einer Unterseite die Anwendung mit den betreffenden Hinweisen bereitgestellt werden. Eine weitere Plattform zur Distribution bietet beispielsweise *GitHub*. Diese Seite ist speziell dazu gedacht, den Quelltext eines Programmes öffentlich anzubieten und die Versionen mit ihren Änderungen zu verwalten. Für Projekte, die unter dem Label *Open Source* laufen, fallen somit auch keine Gebühren oder anderweitige Kosten an. Zudem können ausführbare Programminhalte wie eine Installationsdatei direkt zum Herunterladen angeboten werden. Der Nachteil von *GitHub* ist die etwas umständlichere Darstellung, die gewisse Computerkenntnisse der Anwender erfordert. Außerdem gäbe es keinen direkten Bezug des Inhaltes zur Hochschule mehr. Andererseits wäre es denkbar, dass vom Internetauftritt der Hochschule direkt zum jeweiligen Bereich auf *GitHub* verlinkt wird.

6.4 Hard- und Software der Entwicklungsumgebung

Auf einen wichtigen Bestandteil dieser Arbeit, der bisher noch nicht erwähnt wurde, soll nun zum Schluss kurz eingegangen werden. Beim Entwickeln der Software kommt es natürlich auf die Mithilfe der Testpersonen und die Informationen aus anderen Quellen an. Vor allem ist das Verständnis für die zu realisierende Problematik von großer Bedeutung. Dennoch kann eine solche Anwendung, wie es beim *Wuchshüllenrechner* der Fall ist, nicht einfach geschrieben werden. In erster Linie ist dazu natürlich die Softwaresprache mit ihren Abhängigkeiten und Bibliotheken notwendig. Gerade dieser Punkt hat einen besonders technischen Charakter. Andererseits werden unterschiedliche Betriebssystemumgebungen benötigt und folglich eine leistungsfähige Hardware. Da es sich hierbei um einen essenziellen Punkt dieser Arbeit handelt, der bisher jedoch bewusst nicht im Teil

Material respektive Grundlagen betrachtet wurde, erfolgt ein kleiner Überblick hinsichtlich der verwendeten Hard- und Softwareumgebungen.

Primär wurde die Anwendung auf den Betriebssystemen *OS X El Capitan (10.11)* und *OS X Yosemite (10.10)*, die beide von Apple stammen, entwickelt. Dazu standen ein MacBook Pro sowie ein wesentlich leistungsfähigerer Mac Pro zur Verfügung. Damit lässt sich jedoch unabhängig von der Programmiersprache nur Software für die Apple eigene Umgebung entwickeln, die zudem rein auf einer *64-Bit-Architektur* basiert. Um die Anwendung zumindest für das Microsoft Betriebssystem Windows bereitstellen zu können, wurde eine weitere leistungsfähige Workstation mit *Windows Server 2008 R2* ausgestattet. Damit war es möglich, den plattformunabhängigen Quelltext des *Wuchshüllenrechners* in ein ausführbares Programm für Windows basierte Computer zu übersetzen. Dabei stellte das native Serverbetriebssystem eine Umgebung für die 64-Bit-Architektur bereit. Zusätzlich musste mittels eines virtuellen Servers eine Umgebung für die *32-Bit-Architektur* geschaffen werden. Denn erst 32-Bit-Anwendungen können tatsächlich auch auf älteren Betriebssystemen (teilweise auch *Microsoft Windows 7*) ausgeführt werden. Gleichzeitig stand diese Workstation weiterhin zur Softwareentwicklung zur Verfügung.

Da sich sowohl der Mac Pro als auch die Windows Server basierte Workstation nicht in Rottenburg am Neckar befanden, musste ein weiterer Server bestimmte Dateidienste bereitstellen, um Daten zwischen den unterschiedlichen Systemen problemlos und weltweit austauschen zu können. Auf diesem üblicherweise Linux basierten Betriebssystem wurde ebenfalls die notwendige und äußerst praktische Versionsverwaltungssoftware für das Projekt installiert. Somit konnte auf alle Revisionen der Software zu jedem Zeitpunkt an allen Orten zugegriffen werden. Zusätzlich ermöglicht ein weiteres unabhängiges Backupsystem die Bereitstellung von Speicherplatz für die vollautomatisierte Sicherung der Datenstände.

Insgesamt war es damit immer völlig anstandslos und zuverlässig möglich, den *Wuchshüllenrechner* zu entwickeln. Dass dieser Überblick im Kapitel *Schlussfolgerungen* erwähnt wird, soll der gesamten, zur Verfügung stehenden Umgebung ein besonderes Gewicht verleihen. Außerdem steht ein Großteil der Systeme und

Dienste unabhängig im privaten Heimnetzwerk zur Verfügung und war damit direkt greifbar. Das heißt auch, dass einige Bestandteile dieser Umgebung für die Weiterentwicklung nicht notwendig sind. Daher bilden diese keine allgemeine Verständnisgrundlage für die Entwicklung des *Wuchshüllenrechners*.

7 Quellenverzeichnis

7.1 Literatur

- **ATTESLANDER, P. (2010):** Methoden der empirischen Sozialforschung.
13. Auflage. Erich Schmidt Verlag, Berlin, 387 S.
- **DIEDRICH, O. (2016):** Die passende Programmiersprache finden.
c't Programmieren 2016, S. 6-9
- **ERNESTI, J.; KAISER, P. (2015):** Python 3. Das umfassende Handbuch.
4. Auflage. Rheinwerk Verlag, Bonn, 1032 S.
- **FORSTLICHE BILDUNGSSTÄTTEN [Hrsg.] (2011):** Der Forstwirt.
5. Auflage. Ulmer, Stuttgart, S. 162-176
- **HAMMER, A. (2012):** Entscheidungshilfen zu: Zaun oder Einzelschutz mit
Wuchshüllen. AFZ-DerWald Nr. 23/2012, S. 19-21
- **HEIN, S. (2012): Wuchshüllen: ein Rundum-sorglos-Paket?**
AFZ-DerWald Nr. 16/2012, S. 19
- **HEIN, S.; SPANGENBERG, S. (2012):** Wuchshüllen: Ziele, Funktionen,
Entwicklungen. AFZ-DerWald Nr. 16/2012, S. 20-21
- **KOPP, C.; GÖCKEL, C.; WICHT-LÜCKGE, G. (2012):** Wuchshüllen und
alternative Wildschutzmaßnahmen im Kostenvergleich.
AFZ-DerWald Nr. 16/2012, S. 28-29
- **KURATORIUM FÜR WALDARBEIT UND FORSTTECHNIK (2012):**
Schutzmaßnahmen gegen Wildschäden. Merkblatt Nr. 16/2012
- **KURATORIUM FÜR WALDARBEIT UND FORSTTECHNIK (2016):**
Flächige Schutzverfahren zur Verhinderung von Wildschäden.
AFZ-DerWald Nr. 10-11/2016, S. 20-22
- **LANDESBETRIEB FORST BADEN-WÜRTTEMBERG [Hrsg.] (2011):**
Wuchshüllen: Praxis-Infoblatt zu Einsatz, Kosten und Grenzen ver-
schiedener Wuchshüllen-Modelle. Freiburg, 4 S.
- **LANDESBETRIEB FORST BADEN-WÜRTTEMBERG [Hrsg.] (2014):**
Richtlinie landesweiter Waldentwicklungstypen. Stuttgart, 116 S.

- **LÖFFLER, C.; FRANK, S.; HEIN, S. (2012):** Verwendung von Wuchshüllen in Baden-Württemberg. AFZ-DerWald Nr. 16/2012, S. 22-25
- **MERKERT, J. (2016):** Grafische Programme mit Python und Qt. c't Programmieren 2016, S. 26-29
- **MERKERT, J. (2016):** Objektorientiert programmieren mit Python. c't Programmieren 2016, S. 20-24
- **MERKERT, J. (2016):** Programmieren lernen mit Python. c't Programmieren 2016, S. 10-17
- **NEMESTOTHY, N. (2010):** Welche Wildzaunarten es gibt, und worauf bei ihrer Errichtung geachtet werden sollte. BauernZeitung 19, 13. Mai 2010: II
- **REETZ, M.; WEBER, M. (2012):** Wuchshüllen: Die Technik machts! AFZ-DerWald Nr. 16/2012, S. 26-27
- **SUMMERFIELD, M. (2011):** Rapid GUI programming with Python and Qt: the definitive guide to PyQt programming. 5. Auflage. Prentice Hall, Upper Saddle River, New Jersey
- **THEISEN, M. (2013):** Wissenschaftliches Arbeiten. Erfolgreich bei Bachelor- und Masterarbeiten. 16. Auflage. Verlag Franz Vahlen, München, 311 S.
- **WOLF, J. (2006):** C++ von A bis Z. Das umfassende Handbuch. 1. Auflage. Galileo Press, Bonn, 1229 S.
- **WOLMERINGER, G. (2007):** Java 6 lernen mit Eclipse. 2. Auflage. Galileo Press, Bonn, 595 S.

7.2 Internetseiten

Auf alle Internetseiten wurde abschließend am 5. Juli 2016 zugegriffen:

- **CX_FREEZE DOCUMENTATION [Version 4.4] (2015):**
<https://cx-freeze.readthedocs.io/en/latest/>
- **DIVE INTO PYTHON 3 (2014):**
<http://www.diveintopython3.net/>

- **GNU (2007):** GNU General Public License Version 3.
<https://www.gnu.org/licenses/gpl-3.0.de.html>
- **GOOGLE PYTHON STYLE GUIDE [Revision 2.59] (2016):**
<https://google.github.io/styleguide/pyguide.html>
- **JEFF KNUPP BLOG (2013):** Open Sourcing a Python Project the Right Way. [https://jeffknupp.com/blog/2013/08/16/ ↵](https://jeffknupp.com/blog/2013/08/16/open-sourcing-a-python-project-the-right-way/)
[open-sourcing-a-python-project-the-right-way/](https://jeffknupp.com/blog/2013/08/16/open-sourcing-a-python-project-the-right-way/)
- **OXYGEN ICONS (2015):**
<https://github.com/pasnox/oxygen-icons-png>
- **PYQT DOCUMENTATION [Version 5.6.1] (2015):**
<http://pyqt.sourceforge.net/Docs/PyQt5/index.html>
- **PYQT5 EXAMPLES [Version 5.5.1] (2016):**
<https://github.com/baoboa/pyqt5/tree/master/examples>
- **PYQTGRAPH DOCUMENTATION [Version 0.9.10] (2014):**
<http://www.pyqtgraph.org/documentation/>
- **PYTHON DEVELOPER'S GUIDE (2016):**
<https://www.python.org/dev/>
- **QT DOCUMENTATION [Version 5.7] (2016):**
<http://doc.qt.io/qt-5/>
- **RUNESTONE INTERACTIVE (2013):** Side Effects.
[http://interactivepython.org/runestone/static/pip2/Functions ↵](http://interactivepython.org/runestone/static/pip2/FunctionsSideEffects.html)
[SideEffects.html](http://interactivepython.org/runestone/static/pip2/FunctionsSideEffects.html)
- **STACK OVERFLOW (o.J.):** Verschiedene Forenbeiträge zu Python 3.
<http://stackoverflow.com/>

7.3 Gespräche

- **FECHNER, R.; SCHERLE, C. (18.05.2016):** Benutzerfreundlichkeit und Rückmeldung zur Funktionalität
- **JOSTEN, T. (06.06.2016):** Benutzerfreundlichkeit und Rückmeldung zur Funktionalität

- **KOHNLE, U.; KÄNDLER, G.; WOHNHAS, M. (25.04.2016):** Erste Rückmeldung zum Wuchshüllenrechner
- **KUCHENBECKER, S. (31.05.2016):** Benutzerfreundlichkeit und Rückmeldung zur Funktionalität
- **SOMMERFELD, P. (31.05.2016):** Benutzerfreundlichkeit und Rückmeldung zur Funktionalität
- **WIEGERT, S. (31.05.2016):** Benutzerfreundlichkeit und Rückmeldung zur Funktionalität

7.4 Gesetzestexte

- Umsatzsteuergesetz (UStG) in der Fassung der Bekanntmachung vom 21.02.2005, zuletzt geändert am 02.11.2015

7.5 Kataloge

- **FORSTBAUMSCHULE STINGEL (2015):** Grün ist Leben.
Katalog Herbst 2015 / Frühjahr 2016, 48 S.
- **GRUBE KG (2015):** Fachkatalog Nr. 56. 2015/2016, 656 S.

7.6 Bildinhalte

- **LAMOUR/HANSEN (2013):** Herbststimmung.
<https://www.foto.wald-rlp.de/pages/view.php?ref=10423>

8 Anhang

8.1 Einrichtung der Entwicklungsumgebung

Im Folgenden ist die gesamte Installation von Python 3 und PyQt 5 in Verbindung mit weiteren Abhängigkeiten für die beiden unterschiedlichen Betriebssysteme von Apple und Microsoft erläutert.

8.1.1 Betriebssysteme von Apple

Installationsanweisungen für Python 3:

- Python 3.4.3 herunterladen und installieren:
[\[https://www.python.org/downloads/release/python-343/\]](https://www.python.org/downloads/release/python-343/)
`$ /Applications/Python 3.4/Update Shell Profile.command`

Installationsanweisungen für sip:

```
$ python3 configure.py -d /Library/Frameworks/Python.framework/Versions/  
3.4/lib/python3.4/site-packages --arch x86_64  
  
$ make  
  
$ sudo make install
```

Installationsanweisungen für PyQt 5:

- Zeilen 2591 bis 2594 in configure.py auskommentieren
`$ python3 configure.py --destdir /Library/Frameworks/Python.framework/
Versions/3.4/lib/python3.4/site-packages --qmake /Developer/Qt5.5.0/5.5/
clang_64/bin/qmake --sip /Library/Frameworks/Python.framework/
Versions/3.4/bin/sip

$ make

$ sudo make install`

Installationsanweisungen für GNU Fortran:

- GNU Fortran herunterladen und installieren:
[\[https://gcc.gnu.org/wiki/GFortranBinaries\]](https://gcc.gnu.org/wiki/GFortranBinaries)
- Hinweise: [\[https://gcc.gnu.org/wiki/GFortranBinariesMacOS\]](https://gcc.gnu.org/wiki/GFortranBinariesMacOS)

Installationsanweisungen für Cython:

- Cython herunterladen: [<http://cython.org/#download>]
- Hinweise: [<http://docs.cython.org/src/quickstart/install.html>]

```
$ python3 setup.py install
```

Installationsanweisungen für nose 1.3.7:

- nose 1.3.7 herunterladen: [<https://pypi.python.org/pypi/nose/1.3.7>]
- Hinweise: [<https://nose.readthedocs.org/en/latest/>]

```
$ pip3 install nose-1.3.7-py3-none-any.whl
```

Installationsanweisungen für NumPy:

```
$ export CC=clang
$ export CXX=clang++
$ export FFLAGS=-ff2c
$ git clone https://github.com/numpy/numpy.git
$ python3 setup.py build
$ python3 setup.py install
```

- NumPy testen
- Hinweise: [<http://www.scipy.org/scipylib/building/macosx.html>]

Installationsanweisungen für PyQtGraph:

- Homepage: [<http://www.pyqtgraph.org/>]
- Hinweise: [<https://github.com/pyqtgraph/pyqtgraph>]

```
$ git clone https://github.com/pyqtgraph/pyqtgraph.git
```

```
$ python3 setup.py install
```

- Beispiele anzeigen

8.1.2 Betriebssysteme von Microsoft

Installationsanweisungen für Python 3 und PyQt 5:

- Python 3.4.3 herunterladen und installieren:
[<https://www.python.org/downloads/release/python-343/>]
- PyQt 5.5 herunterladen und installieren:
[<https://www.riverbankcomputing.com/software/pyqt/download5>]

Installationsanweisungen für cx_Freeze:

- cx_Freeze herunterladen und installieren:
[[https://pypi.python.org/pypi?%3Aaction=display&name=cx_](https://pypi.python.org/pypi?%3Aaction=display&name=cx_Freeze&version=4.3.4) ↵
Freeze&version=4.3.4]

Installationsanweisungen für nose 1.3.7:

- nose 1.3.7 herunterladen: [<https://pypi.python.org/pypi/nose/1.3.7>]
 - Hinweise: [<https://nose.readthedocs.org/en/latest/>]
- ```
$ pip install nose-1.3.7-py3-none-any.whl
```

Installationsanweisungen für NumPy:

- NumPy 1.9.2 (unoptimized) herunterladen:  
[<http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>]
- ```
$ pip install numpy-1.9.2+unoptimized-cp34-none-win_amd64.whl
```
- NumPy testen
 - Hinweise: [[https://gehrcke.de/2015/02/](https://gehrcke.de/2015/02/how-to-set-up-a-64-bit-version-of-numpy-on-windows/) ↵
how-to-set-up-a-64-bit-version-of-numpy-on-windows/]

Installationsanweisungen für PyQtGraph:

- Homepage: [<http://www.pyqtgraph.org/>]
 - PyQtGraph von GitHub als ZIP-Datei herunterladen (Hinweise):
[<https://github.com/pyqtgraph/pyqtgraph>]
- ```
$ python setup.py install
```
- Beispiele anzeigen

## 8.2 Beispielanfrage zur Evaluation

### 8.2.1 Inhalt der E-Mail

Sehr geehrte Forstpraktiker,

im Rahmen meiner Bachelorarbeit bei Herrn Prof. Dr. Sebastian Hein an der Hochschule für Forstwirtschaft Rottenburg möchte ich Sie bitten, dass Sie sich mein bisheriges Ergebnis anschauen. Es handelt sich um die Windows-Anwendung „Wuchshüllenrechner“.

Ziel der Anwendung ist es, einen Vergleich zwischen Einzel- und Flächenschutz darzustellen. Das Programm soll zeigen, wann sich welche Schutzvariante lohnt in Abhängigkeit der Zaunlänge und Pflanzenzahl.

Mir wäre besonders wichtig, was Sie zu folgenden Punkten denken:

- Benutzerfreundlichkeit und Bedienbarkeit
- Verständlichkeit des Diagramms – Welche Beschriftung fehlt Ihnen noch?
- Praxistauglichkeit
- Dateneingabe
- sonstige praktische Hinweise

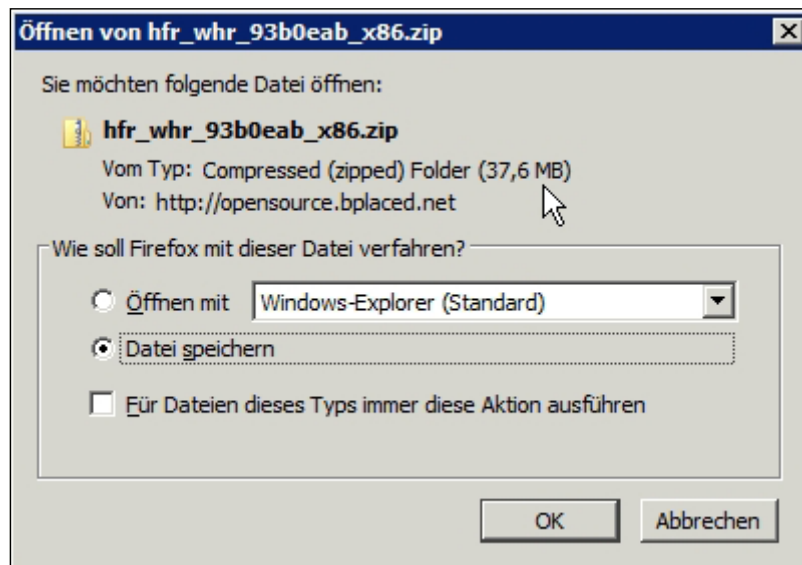
Das Programm können Sie hier herunterladen: [http://opensource.bplaced.net/wuchshuellenrechner/download/x86/hfr\\_whr\\_93b0eab\\_x86.zip](http://opensource.bplaced.net/wuchshuellenrechner/download/x86/hfr_whr_93b0eab_x86.zip)

Außerdem finden Sie hier die wissenschaftliche Grundlage zu meinem Programm: [http://www.waldwissen.net/waldwirtschaft/fuehrung/rechnung/fva\\_hb6\\_kostenrechnung/index\\_DE](http://www.waldwissen.net/waldwirtschaft/fuehrung/rechnung/fva_hb6_kostenrechnung/index_DE)

#### **Wichtig:**

Ich möchte Sie darauf hinweisen, dass Sie Datei wirklich herunterladen müssen. Sobald Sie auf den Link geklickt haben, fragt der Browser, ob Sie die Datei lediglich öffnen oder speichern wollen (Grafik auch im Anhang).

Bitte wählen Sie in jedem Fall die Option *Speichern* aus. Nur so kann das Programm nach dem Entpacken wirklich gestartet werden und es kommt nicht zu einer Fehlermeldung – Systemfehler.



Wie Sie nach dem Download fortfahren müssen, habe ich in der Anleitung im Anhang (*anleitung\_entpacken.pdf*) abgebildet.

Ich bitte darum, diesen technischen Umstand zu entschuldigen und sich davon nicht abschrecken zu lassen. Eine einfache Lösung ist derzeit nicht möglich.

Außerdem habe ich Ihnen zum Testen den Beispieldatensatz im Anhang als Grafik eingefügt. Sie können selbstverständlich mit eigenen Daten testen.

Das Programm enthält keine Viren etc. Ich würde mich wirklich sehr über Ihr Feedback freuen, vielleicht schaffen Sie es bis zum 31. Mai.

Viele Grüße

Tobias Helfenstein

## 8.2.2 Beispieldatensatz

Als Testdatensatz wurden den Testpersonen die Beispieldatensätze aus dem Fachbeitrag von HAMMER zur Verfügung gestellt.

**Tab. 1: Durchschnittliche Kostenstruktur des Beispieldatensatzes**

| alle Werte je Pflanze<br>Ausgenommen A1 | Zaun                         | Wuchshülle            |
|-----------------------------------------|------------------------------|-----------------------|
| <b>1 Meter Zaun inklusive Abbau</b>     | A1 = 14,0 €/m                |                       |
| <b>Tubex Ventex + Stab</b>              |                              | B1 = 1,65 €           |
| <b>Abbau</b>                            |                              | B2 = 1,25 €           |
| <b>Bergahorn</b>                        | A2 = 0,83 €, 50-80 cm        | B3 = 0,32 €, 30-50 cm |
| <b>Douglasie</b>                        | A2 = 0,99 €, 40-70 cm        | B3 = 0,69 €, 20-40 cm |
| <b>Pflanzkosten</b>                     | A3 = 1,17 €                  | B4 = 0,74 €           |
| <b>Installieren der Wuchshülle</b>      |                              | B5 = 1,61 €           |
| <b>Kulturpflege</b>                     | A4 = 0,3€                    | B6 = 0,15 €           |
| <b>Lohnkosten</b>                       | 29,41 € + 19 % = 35 €/Stunde |                       |

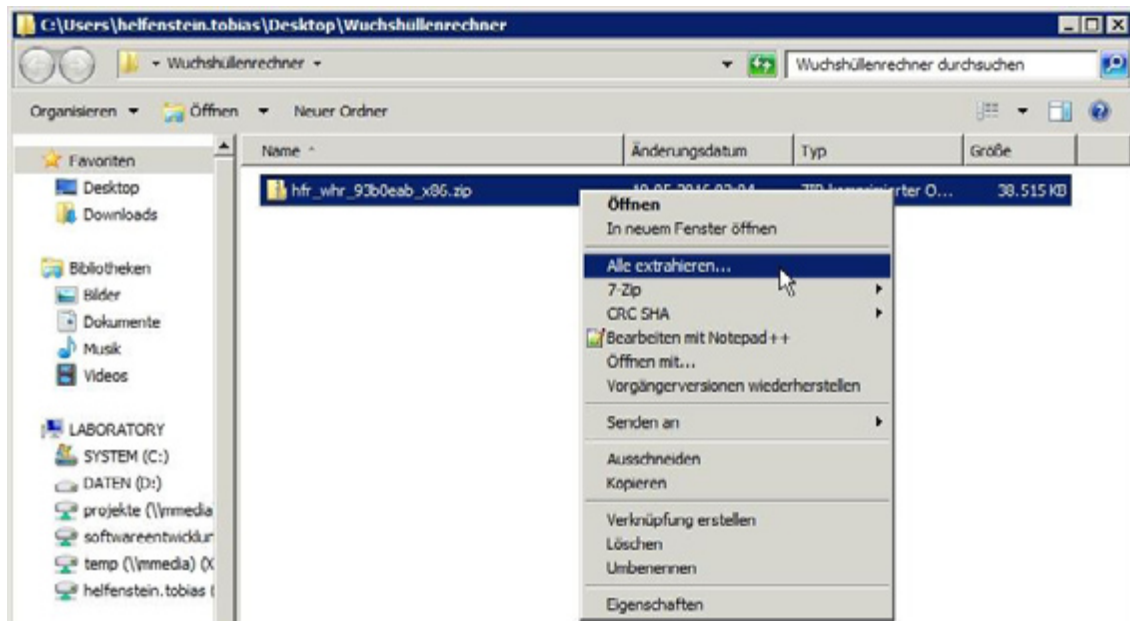
**Zusätzlich benötigte Angaben:**

- Zaunlänge: 400 laufende Meter
- Anzahl der Pflanzen: 900 (für Berg-Ahorn und Küsten-Douglasie)

## 8.2.3 Anleitung zum Entpacken und Starten

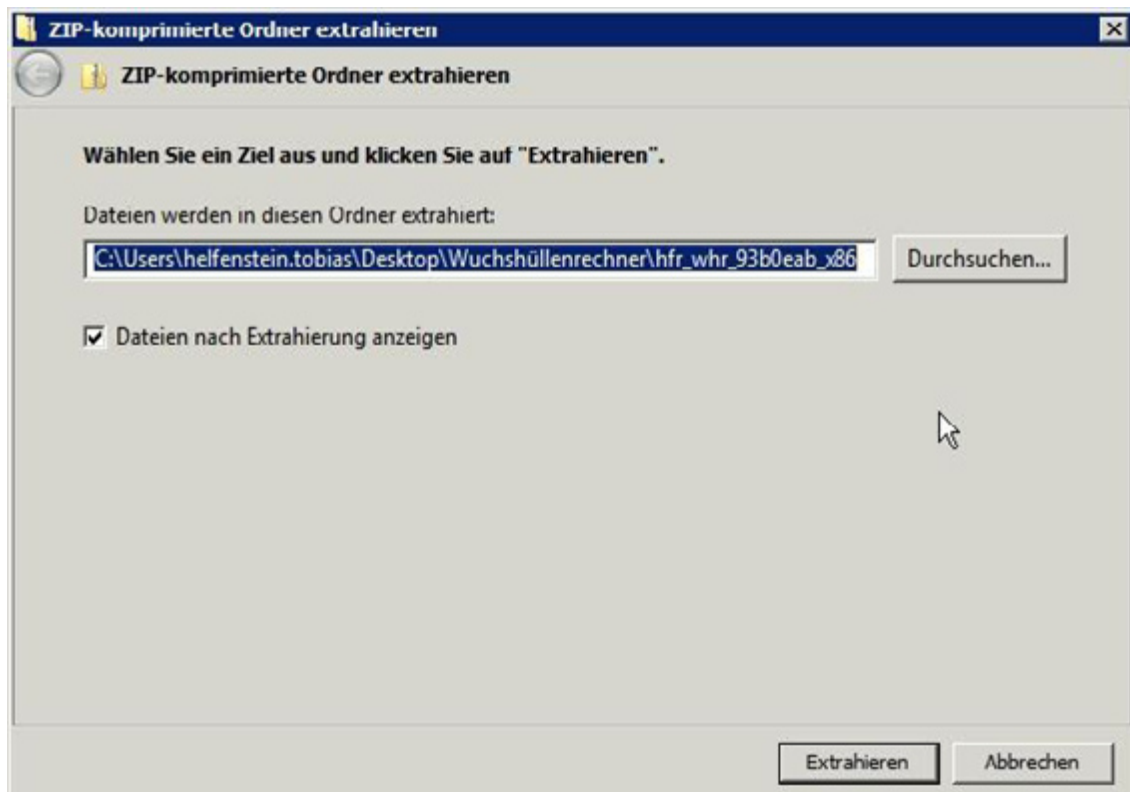
### 1. Schritt

Sobald Sie die Datei heruntergeladen haben, müssen Sie den ZIP-Ordner „hfr\_whr\_93b0eab\_x86.zip“ entpacken. Hierzu machen Sie einen Rechtsklick auf die Datei und wählen „Alle extrahieren...“.



### 2. Schritt

Danach erscheint ein Fenster, welches das Ziel zum Entpacken abfragt. Wählen Sie hierzu bitte ein Verzeichnis, das Sie schnell wieder finden.



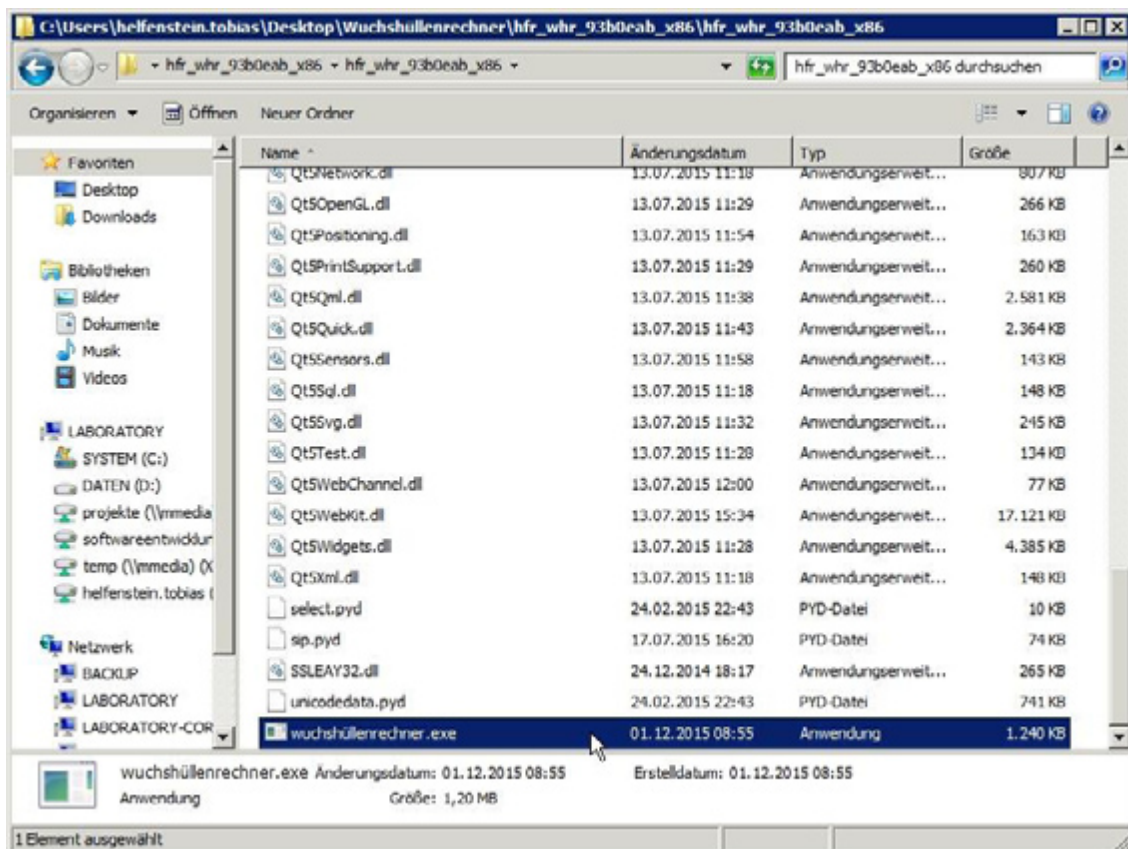
### 3. Schritt

Mit dem Klick auf „*Extrahieren*“ wird die Datei entpackt und das Zielverzeichnis automatisch geöffnet. Das Zielverzeichnis enthält einen Dateiordner „*hfr\_whr\_93b0eab\_x86*“. Diesen Ordner müssen Sie öffnen.



### 4. Schritt

Zum Schluss starten Sie den Wuchshüllenrechner, indem Sie ganz nach unten scrollen und die Datei „*wuchshüllenrechner*“ mit einem Doppelklick ausführen.



## 9 Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Abschlussarbeit im Fachbereich Waldbau mit dem Titel „Entwicklung der Software ‚Wuchshüllenrechner‘ zur Unterstützung der Entscheidung ‚Zaun oder Einzelschutz‘“ selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

**Tobias Helfenstein**

Freiherr-vom-Stein-Straße 37  
55606 Kirn

Kirn, den 8. Juli 2016

.....  
(Unterschrift)

