



Projektdokumentation

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Prüfling: Elvira Musterfrau

Prüfnummer: 4711

Ausbildungsbetrieb: Hottgenroth Software GmbH & Co. KG



Prozessorientierter Projektbericht

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 02.04.2013

Inhaltsverzeichnis

1.	Einleitung.....	4
2.	Projektbeschreibung	4
2.1	Projektaufgabe	4
2.2	Projektumfeld.....	4
2.3	Projektschnittstellen	6
2.3.1	Organisatorische Schnittstellen.....	6
2.3.2	Technische Schnittstellen	6
2.4	Änderungen gegenüber dem Projektantrag	6
3.	Analyse- und Definitionsphase.....	6
3.1	Ist-Analyse	6
3.2	Soll-Konzept.....	7
3.3	Vorgehensmodell	7
3.4	Durchführbarkeitsanalyse	7
3.5	Pflichtenheft	8
4.	Planungsphase.....	8
4.1	Arbeitspakete	8
4.2	Zeit- und Meilensteinplan	8
4.3	Projektstrukturplan	10
4.4	Ressourcen- und Kostenplan.....	10
4.5	Qualitätsplan	10
5.	Entwurfsphase.....	12
5.1	DV-Dokumente (UML)	12
5.2	Testfallkatalog	13
6.	Implementierungsphase.....	13
6.1	Programmoberfläche	13
6.2	Umsetzung der Funktionalitäten in Quellcode	13
7.	Testphase	14
8.	Abschlussphase	15
8.1	Entwicklerdokumentation	15
8.2	Benutzer-Hilfe.....	15
8.3	Soll-Ist-Zeitvergleich und Kosten-Nutzen-Analyse	15
8.4	Qualitätsbericht.....	16
8.5	Übergabe des Projektes an den Auftraggeber	16
8.6	Projektreflexion.....	16
9.	Anlagenverzeichnis.....	17

1. Einleitung

Alle hier entstandenen Dokumente sind von mir, Elvira Musterfrau, ohne Inanspruchnahme fremder Hilfe entstanden. Ebenso wurden alle aufgeführten Tätigkeiten von mir selber durchgeführt.

Im beigefügten [Glossar](#) (siehe Anhang, S. 137 ff.) ist eine Auflistung aller Fachbegriffe und Fremdwörter zu finden. Fachbegriffe und Fremdwörter werden im Text durch einen Hyperlink (blau) kenntlich gemacht. Ebenso sind die Verweise auf im Anhang befindliche Dokumente mit grünen Links gekennzeichnet.

Die Arbeitspakete des Projektes werden jeweils mit einer direkten Gegenüberstellung der geplanten und tatsächlich verwendeten Zeiten versehen.

2. Projektbeschreibung

2.1 Projektaufgabe

Im Rahmen des vorliegenden Projektes wurde für die Softwareanwendung [CAD](#) („Grafische Erfassung“) ein [Prototyp](#) eines [Algorithmus](#) zur automatischen Vollbelegung von Dächern mit [Photovoltaik-\(PV-\)Modulen](#) entwickelt. Er ermöglicht eine automatische Vollbelegung verschiedener Dachformen mit Photovoltaik-Modulen mit unterschiedlichen Startpunkten. CAD wiederum ist in die Anwendung „PV Simulation“ eingebettet, mit der PV-Anlagen geplant werden können. Mit diesem Algorithmus bzw. dieser Funktion kann die optimale Belegung eines Daches mit PV-Modulen ermittelt werden. Diese Funktion kann für unterschiedliche Formen von Dachflächen (rechteckig, trapez-förmig, dreieckig) und für verschiedene Startpunkte (oben links, unten links, oben rechts, unten rechts) angewendet werden.

Der Algorithmus für die Vollbelegung ist keine eigenständige Anwendung, sondern wird als Modul in die Anwendung „PV Simulation“ mit der Komponente „Grafische Erfassung“ (CAD) integriert. Es ist zunächst ein [Prototyp](#) für die zweidimensionale Visualisierung des Vollbelegungs-Algorithmus zu entwickeln, bevor die Funktion in das vorhandene CAD-Programm implementiert werden kann. Der zu erstellende [Prototyp](#) ist ein Entwurf zur Ausgabe der Ergebnisse des Algorithmus an der Benutzeroberfläche und ist nicht mit dem [Vorgehensmodell](#) des „[Prototyping](#)“ zu verwechseln.

2.2 Projektumfeld

Die Firma Hottgenroth GmbH & Co. KG wurde Mitte der 80er Jahre gegründet. Seit 1986 entwickelt und vertreibt das Unternehmen branchenspezifische Software- und webbasierte Anwendungen für das Schornsteinfegerhandwerk, Energieeffizienz, Bauhaupt- und Nebengebäude und die Planung haustechnischer Anlagen. Zusätzlich wird der kaufmännische Bereich abgedeckt.

Die Firma Hottgenroth GmbH & Co. KG stellt ihren Kunden mit derzeit rund 140 Angestellten nicht nur Software-Lösungen zur Verfügung, sondern bietet einen Rundum-Service durch das Angebot von weiteren Dienstleistungen (u.a. Schulungen, Online-Portal, Support) und Hardware-Lösungen.

Die schrittweise Erweiterung der Produktpalette ist im Folgenden stichpunktartig dokumentiert:

- **2002:** Übernahme des Unternehmens ETU Software GmbH mit der dort entwickelten TGA Software-Anwendung (TGA: Technische Gebäude-Ausrüstung) für Haus- und Gebäudetechnik.
- **2006:** Entwicklung einer Datenerfassungs-Software.
- **seit 2007:** Erstellung von kaufmännischen Software-Lösungen.
- **Ende 2008:** Entwicklung einer CAD-Software („Grafische Erfassung“).
- **Mitte 2010:** Umstrukturierung und Neu-Entwicklung der Software „PV Simulation“.

Ende 2008 wurden die ersten Weichen zur Entwicklung einer „3D-Grafischen Erfassung“ (CAD) gestellt. Bis heute wird die Komponente der dreidimensionalen Erfassung und Darstellung von Gebäuden in immer mehr Anwendungen der Firma Hottgenroth Software GmbH & Co. KG integriert.

Mitte 2010 entwickelte die Firma Hottgenroth GmbH & Co. KG eine neue Version der Software „PV Simulation“. Darin wurde die „Grafische Erfassung“ eingebettet, mit deren Hilfe Gebäude grafisch erfasst und deren Dächer mit Modulen belegt werden können. Anschließend werden in „PV Simulation“ die angelegten Module berechnet und verschaltet und der Ertrag berechnet.

Anwendung findet der zu entwickelnde Algorithmus in „PV Simulation“. Die erfassten Module werden in der Projekt-Datenbank abgespeichert. Wird die „Grafische Erfassung“ (CAD) aus einer anderen Anwendung, wie zum Beispiel der hauseigenen [Energieberater-Software](#) oder der Software der [TGA-Heizung](#) heraus aufgerufen, werden die Photovoltaik-Module zwar visualisiert, werden aber noch nicht weiter verarbeitet. Dies ist ausschließlich in „PV Simulation“ möglich. Darüber hinaus erlaubt das von der Firma Hottgenroth GmbH & Co. KG entwickelte „Datenmodell“ fast alle einmal eingegebenen und erfassten Daten (z.B. Adressen, Bauteile, [U-Werte](#), Gebäudestruktur) mit mehreren Anwendungen aufzurufen, zu verändern und weiter zu geben. Dieses Vorgehen, Gebäude nur einmal zu erfassen und die Daten mit anderen Anwendungen abzugleichen und zur Verfügung zu stellen, wird innerhalb des Unternehmens als „Datenaustausch“ bezeichnet. Eine Übersicht der CAD-nutzenden Anwendungen wird in der [Ist-Analyse](#) (siehe Anhang, S. 30) gezeigt.

2.3 Projektschnittstellen

2.3.1 Organisatorische Schnittstellen

Zu den organisatorischen Schnittstellen gehören:

- Herr Max Mustermann (Teamleiter der TGA-Entwicklung) als Auftraggeber
- Herr F. Schulze (Entwicklungs-Abteilung) als Projektverantwortlicher und Projektansprechpartner
- Frau Anna Ausbilder als Ausbilderin
- Herr H. Meier (Abteilung für Qualitätssicherung) als [Blackbox-Tester](#)

2.3.2 Technische Schnittstellen

Der Algorithmus der Dach-Vollbelegung wird keine eigenständige Anwendung, sondern in die „Grafische Erfassung“ integriert. Durch die „Grafische Erfassung“ werden die Maße des Daches und die Größe der Photovoltaik-Module vorgegeben. Herstellerdaten diverser PV-Module können über einen eingebundenen Hersteller-Katalog ausgewählt werden.

2.4 Änderungen gegenüber dem Projektantrag

Im Auftrag der Geschäftsführung übernahm Herr Max Mustermann die Rolle des Auftraggebers. Ansonsten haben sich im Vergleich zum eingereichten [Projektantrag](#) (siehe Anhang, S. 18) keine weiteren Abweichungen ergeben.

3. Analyse- und Definitionsphase

3.1 Ist-Analyse

In einer Besprechung mit dem Projekt-Verantwortlichen, Herrn F. Schulze, wurden die bisherigen Möglichkeiten der Dachbelegung mit Photovoltaik-Modulen in der Anwendung „Grafische Erfassung“ in Verbindung mit „PV Simulation“ erörtert. Die Software bietet dem Anwender bisher keine Möglichkeit Dächern automatisch mit PV-Modulen zu belegen.

Bei dem geplanten Algorithmus für die automatische Vollbelegung von Dächern handelt es sich demnach um eine Neuentwicklung.

Nähere Erläuterungen zur [IST-Situation](#) zur Belegung von Dächern mit PV-Modulen sind im Anhang (siehe Anhang, S. 30 ff.) zu finden.

geplante Zeit: 1 Stunde	benötigte Zeit: 1 Stunde	Differenz: 0 Stunden
-------------------------	--------------------------	----------------------

3.2 Soll-Konzept

Gewünscht ist ein Algorithmus, mit dessen Hilfe Standard-Dachflächen (rechteckig, dreieckig und trapezförmig) automatisch nach Eingabe diverser Größen, wie zum Beispiel Modul-Breite /-Höhe, Abstand zu den Dachkanten und zwischen den Modulen, mit verschiedenen Ausgangspunkten optimal belegt werden können. Optimal heißt, dass eine maximal mögliche Anzahl an Modulen für jede Fläche ermittelt werden soll.

Da der Rechenkern des Algorithmus nach Erstellung des [Prototyps](#) in die bestehende Anwendung „Grafische Erfassung“ (CAD) integriert werden soll, müssen weitreichende Überlegungen bezüglich Kompatibilität und Sichtbarkeit der Funktion innerhalb des Programmes nicht weiter angestellt werden. Ob die Funktion der Dachvollbelegung mit PV-Modulen frei geschaltet ist oder nicht, hängt davon ab, ob dem Anwender das Programm „PV-Simulation“ zur Verfügung steht. Dementsprechend werden die Funktionen in der „Grafischen Erfassung“ frei geschaltet. Das erstellte Dokument zum [SOLL-Konzept](#) ist im Anhang (siehe Anhang, S. 34 ff.) zu finden.

Zum besseren Verständnis der Programm-Abläufe ist ein Use-Case-Diagramm erstellt worden. Dies ist im [Pflichtenheft](#) (siehe Anhang, S. 43 ff.) unter Punkt 3.3 oder auch in den erstellten [DV-Dokumenten](#) (siehe Anhang, S. 65 ff.) zu finden.

geplante Zeit: 2 Stunden	benötigte Zeit: 2 Stunden	Differenz: 0 Stunden
--------------------------	---------------------------	----------------------

3.3 Vorgehensmodell

Für das vorliegende Projekt wird das bekannte [erweiterte Wasserfallmodell](#) als [Vorgehensmodell](#) gewählt (siehe Anhang, S. 39 ff.). Im Gegensatz zum klassischen [\(einfachen\) Wasserfallmodell](#) ist das [erweiterte Wasserfallmodell](#) flexibler in seiner Handhabung. Das heißt, für den Fall, dass Engpässe und Probleme im Lauf des Projektes sichtbar werden, hat man die Option, in die Phase, in der die Verzögerungen planerisch angesiedelt sind, zurück zu springen, um hier das entstandene Problem zu beseitigen. Da die gewünschten Kriterien und Anforderungen schon zu Beginn des Projektes klar definiert sind, kann von flexibleren [Vorgehensmodellen](#), die eine Änderung und Überarbeitung der Funktionen und Anforderungen im Projekt-Verlauf erlauben, wie beispielsweise „[Extreme Programming](#)“, „[Prototyping](#)“ oder das „[Spiralmodell](#)“, was sich für größere und umfangreichere Projekte anbietet, abgesehen werden.

3.4 Durchführbarkeitsanalyse

Im Rahmen der [Durchführbarkeitsanalyse](#) wird das zu erstellende Projekt für den Algorithmus zur automatischen Vollbelegung von Dächern mit Photovoltaik-Modulen auf Machbarkeit, Risiko und Wirtschaftlichkeit untersucht (siehe Anhang, S. 41 ff.). Die Auswertung der Analyse ergibt, dass das Projekt zur Erstellung des Algorithmus ohne weiteres realisierbar ist. Somit ist auch das zu erwartende Risiko für den Fall, dass es nicht umsetzbar wäre, gering. Bei der Betrachtung der

Wirtschaftlichkeit wurde festgestellt, dass der Algorithmus mit Prototyp zur Veranschaulichung keinen monetären Nutzen hat, da es keine separat zu verkaufende Anwendung ist. Das Projekt wird sich über den Verkauf der Anwendung „PV Simulation“ amortisieren, da es eine zusätzliche Funktion für die Anwender dieser Software darstellt.

geplante Zeit: 1 Stunde	benötigte Zeit: 1 Stunde	Differenz: 0 Stunden
-------------------------	--------------------------	----------------------

3.5 Pflichtenheft

Im Rahmen der Erstellung des Pflichtenheftes (siehe Anhang, S. 43 ff.) sind die genauen Musskriterien, wie z.B. verschiedene Startpunkte zur automatischen Vollbelegung (oben links, unten rechts, u.a.), mögliche Formen der Dachflächen (Rechteck, Dreieck und Trapez) sowie die optional eingebbaren Abstände zwischen den Modulen oder zu den Dachrändern ermittelt worden. Die erforderlichen Produktdaten, z.B. die einzugebenden Größen, die Qualitätsanforderungen sowie die Entwicklungs- und Produktumgebung, sind definiert worden.

Mit der Abnahme durch den Auftraggeber und der Unterschrift von Auftragnehmer und Auftraggeber wird das Pflichtenheft zu einem verbindlichen Vertragsgegenstand.

geplante Zeit: 6 Stunden	benötigte Zeit: 6 Stunden	Differenz: 0 Stunden
--------------------------	---------------------------	----------------------

4. Planungsphase

4.1 Arbeitspakete

Mit der Abnahme des Pflichtenheftes kann die Definitions- und Analyse-Phase abgeschlossen werden. In einem nächsten Schritt wird die Planungsphase mit der Identifizierung der einzelnen Arbeitspakete (siehe Anhang, S. 49 ff.) eröffnet. Der Vollständigkeit halber werden auch die Arbeitsschritte, die bis hierhin schon abgearbeitet wurden, ebenfalls mit erfasst.

Das Definieren der Arbeitspakete stellte sich als wenig zeitaufwändig heraus. Hingegen nahm die Kategorisierung, zu welcher Phase ein Arbeitspaket gehört, etwas mehr Zeit in Anspruch, da mehrere Einteilungen denkbar waren. Auch die grafische Darstellung dauerte länger als angenommen.

geplante Zeit: 1 Stunde	benötigte Zeit: 1,5 Stunden	Differenz: + 0,5 Stunden
-------------------------	-----------------------------	--------------------------

4.2 Zeit- und Meilensteinplan

Durch die zeitliche Vorgabe von 70 Stunden für die Durchführung und Bearbeitung der betrieblichen Projektarbeit wird anhand der identifizierten Arbeitspakete die folgende Zeitplanung erstellt. Die angesetzte Zeit zur Projektbearbeitung liegt bei 67 Stunden. Die verbleibende Zeit von 3 Stunden wird als Pufferzeit deklariert.

Phase	Arbeitspaket	Soll-Zeit in h
Definition & Analyse	IST-Analyse	1
	SOLL-Konzept	2
	Durchführbarkeitsanalyse	1
	Pflichtenheft	6
Planung	Arbeitspakete	1
	Planungsdokumente	5
	Qualitätsplan	1,5
Entwurf	Testfallkatalog	1,5
	DV-Entwurfsdokumente	8
	Entwurf Programmoberfläche	3
Implementierung	Erstellung der Programmoberfläche	1
	Programmierung	13
Test	Whitebox-Test	5
	Fehlerkorrektur	1
Abschluss	SOLL-IST-Vergleich	1,5
	Qualitätsbericht	1,5
	Abnahmeprotokoll inkl. Übergabe	0,5
Dokumentation	Entwicklerdokumentation	2,5
	Benutzer-Hilfe	1
	Prozessorientierter Projektbericht	10
Summe		67

Zur grafischen Darstellung des tatsächlichen [Zeitplans](#) ist ein [Gantt-Diagramm](#) (siehe Anhang, S. 53 ff.) verwendet worden. Hier lassen sich die zeitlichen Abfolgen und Abhängigkeiten schnell ablesen und erfassen. Alternativ hätte ein [Netzplan](#) verwendet werden können, der die gleiche Aufgabe hat, wie ein [Gantt-Diagramm](#). Da dieses Projekt allerdings nur von einer Person entwickelt wird, müssen potentiell parallel ablaufende Entwicklungen nicht visualisiert werden.

Des Weiteren fließt die Meilensteinplanung in das [Gantt-Diagramm](#) ein, so dass Termine und Ziele ebenfalls rasch zu erfassen sind. Eine detaillierte Planung der [Meilensteine](#) ist im Anhang (siehe Anhang, S. 53 ff.) zu finden.

Der anvisierte [Blackbox-Test](#) wird von Herrn H. Meier durchgeführt, der für diesen Test vier Stunden ansetzte. Da der [Blackbox-Test](#) parallel zum [Whitebox-Test](#) erfolgt und zudem nicht von der Antragstellerin selber durchgeführt wird, fließt dies auch nicht in die Zeitplanung ein.

Das Erstellen des Zeit- und Meilensteinplans nahm weniger Zeit in Anspruch als angenommen, da die Arbeitspakete zuvor schon überlegt definiert worden sind.

geplante Zeit: 2 Stunden	benötigte Zeit: 1,5 Stunden	Differenz: - 0,5 Stunden
--------------------------	-----------------------------	--------------------------

4.3 Projektstrukturplan

Der für dieses Projekt erstellte [Projektstrukturplan](#) (siehe Anhang, S. 57 ff.) dient der Veranschaulichung der Arbeitspakete in den dazugehörigen Phasen. In diesem Zusammenhang wird nicht nur die Phasenzugehörigkeit, sondern auch der Aufbau des gesamten Projektes visualisiert.

geplante Zeit: 1 Stunde	benötigte Zeit: 1 Stunde	Differenz: 0 Stunden
-------------------------	--------------------------	----------------------

4.4 Ressourcen- und Kostenplan

Die benötigten technischen und personellen Ressourcen sind in einem [Ressourcenplan](#) (siehe Anhang, S. 59 ff.) erfasst worden. Die Firma Hottgenroth GmbH & Co. KG hat eine Stundenpauschale von 89,00 Euro für einen Entwickler festgelegt. In diesem Stundensatz sind sowohl Lohn- und Lohnnebenkosten als auch Raum- und Betriebskosten für den Computer-Arbeitsplatz enthalten. Solche Stundensätze werden auch vollkostenpauschal genannt, da sie alle Nebenkosten prozentual berücksichtigen.

Zwischen den Stundensätzen eines Auszubildenden, eines Entwicklers und eines Abteilungsleiters wird im Unternehmen kein Unterschied gemacht. Tatsächlich braucht ein Auszubildender zwar mehr Unterstützung, allerdings belegt er einen gleichwertigen Arbeitsplatz wie ein regulärer Entwickler oder ein Abteilungsleiter. Somit erwirtschaftet die Firma Hottgenroth Software einen gewissen Gewinn für den Fall, dass ein Auszubildender entwickelt, da dessen tatsächlicher Stundensatz niedriger liegt als der pauschale Stundensatz. Andererseits erleidet das Unternehmen einen gewissen Verlust für den Fall, dass ein Abteilungsleiter an der Entwicklung von Software beteiligt ist, da dessen tatsächlicher Stundensatz oberhalb des pauschal angesetzten Stundensatzes liegt.

Durch die beschriebenen Abweichungen nach oben und unten vom pauschalen Stundensatz gleichen sich etwaige Gewinne und Verluste gegenseitig aus. Eine detaillierte Aufstellung des Stundensatzes und der [Kosten](#), die bei diesem Projekt entstehen, ist im Anhang zu finden (siehe Anhang, S. 59 ff.). Vorab wurden Kosten in Höhe von 6.452,50 € berechnet.

geplante Zeit: 2 Stunden	benötigte Zeit: 2 Stunden	Differenz: 0 Stunden
--------------------------	---------------------------	----------------------

4.5 Qualitätsplan

Um die Qualität eines Produktes oder Projektes zu beschreiben, wird zwischen produktbezogenen und projektbezogenen Maßnahmen der Qualitätssicherung unterschieden. Die angewandten qualitätssichernden Maßnahmen, die mit ihnen zu erreichenden Ziele und die notwendigen Prüfkriterien zur Kontrolle werden in einem [Qualitätsplan](#) (siehe Anhang, S. 62 ff.) zusammengefasst.

Der zuvor erstellte Kostenplan bietet eine Übersicht zur Kontrolle, ob der unter Umständen für das Projekt festgelegte Budgetrahmen nicht überschritten wird. Diese projektbezogene Maßnahme greift in diesem Fall nicht, da durch den Auftraggeber die Firma Hottgenroth GmbH & Co. KG kein Limit beziffert wurde. Der angefertigte Zeit- und Ressourcenplan sowie der Meilensteinplan zählen zu den qualitätssichernden Maßnahmen, die sich auf das Projekt und dessen Durchführung beziehen. Diese Dokumente dienen der Kontrolle über die Einhaltung von Zeiten und Fristen, die durch Meilensteine gekennzeichnet sind. Zur Sicherung der Ergebnisse wird nach dem Erreichen eines Meilensteines ein [Review](#) mit dem Auftraggeber durchgeführt, bevor mit dem Projekt fortgefahren werden kann. Des Weiteren bildet das vom Auftraggeber abgenommene und genehmigte Pflichtenheft eine Komponente der projektbezogenen Maßnahmen, die zur Sicherung der Qualität herangezogen wird. Der strukturierte Ablauf des Projektes wird durch die begründete Wahl des Vorgehensmodells gewährleistet.

Zu den produktbezogenen Maßnahmen, die zur Sicherstellung der Qualität des Produktes durchgeführt werden, zählt unter anderem das Erstellen eines Testkonzeptes. Dieses enthält einen detaillierten Testfallkatalog für den [Whitebox-Tester](#), der ebenso für den [Blackbox-Test](#) Anwendung findet. Hier wird aufgeführt, welche Randbedingungen und Grenzfälle mit welchen zu erwartenden Ergebnissen getestet werden sollen. Es ist wichtig, dass die erwarteten Ergebnisse festgehalten werden, damit Abweichungen und Fehler in der Anwendung gefunden werden können. Für beide Test-Arten wurde der gleiche Testfallkatalog verwendet. Dennoch war die Vorgehensweise unterschiedlich: beim [Whitebox-Test](#) wird in der Entwicklungsumgebung durch das Setzen von Haltepunkten kontrolliert, welche Werte innerhalb der Funktionen ermittelt und übergeben werden. Beim [Blackbox-Test](#) wird aus Sicht des Anwenders getestet, ob das erwartete Ergebnis eintritt (Anzeige von Ergebnissen auf der Benutzer-Oberfläche), ohne die inneren Funktionen des Quellcodes zu kennen.

Verwendete produktbezogene Maßnahmen, um die Wartbarkeit und die Erweiterbarkeit des Algorithmus zu gewährleisten, sind die Verwendung von festgelegten Namenskonventionen (z.B. zur Benennung von Variablen) sowie das Einrücken und Kommentieren des Quellcodes, um seine Lesbarkeit zu erhöhen. Bei der Erstellung des Algorithmus wurden die Paradigmen der objektorientierten Programmierung angewandt, damit der Quellcode bei Erweiterungen ohne weiteres wieder verwendet werden kann. Um eine anwenderfreundliche Bedienbarkeit sicherzustellen, ist ein erster Entwurf der Oberfläche als Prototyp entworfen worden. Da es sich bei dem Algorithmus um keine direkt an der Benutzeroberfläche sichtbare Aktion handelt, kann die Einhaltung der internen Norm zur Softwaregestaltung vernachlässigt werden.

geplante Zeit: 1,5 Stunden	benötigte Zeit: 1,5 Stunde	Differenz: 0 Stunden
----------------------------	----------------------------	----------------------

5. Entwurfsphase

5.1 DV-Dokumente (UML)

Als Einstieg in die Entwurfsphase werden [DV-Entwurfsdokumente](#) in Form von UML-Diagrammen erstellt. Diese erstellten Dokumente sind im Anhang (siehe Anhang, S. 65 ff.) zu finden. Zu den erstellten DV-Entwurfsdokumenten gehören unter anderem ein Use-Case-Diagramm, ein Aktivitätsdiagramm, ein Klassendiagramm sowie ein [Prototyp](#) für die Programmoberfläche.

Im Rahmen des Pflichtenheftes wurde bereits ein Use-Case-Diagramm erstellt, woraus die anderen Diagramme schrittweise entwickelt werden. Das Use-Case- oder auch Anwendungsfall-Diagramm stellt die Interaktion zwischen dem Anwender und den geplanten Funktionen verwendet. Alle Funktionen werden von menschlichen oder nicht-menschlichen Anwendern (so genannte Akteure) ausgeführt. Bezogen auf das Use-Case-Diagramm entsteht eine Anwendungsfall-Erzählung, die einen Ablauf aus der Sicht des Anwenders heraus beschreibt. Diese Erzählung ist in Bezug auf Personennamen und Orte frei erfunden. Sie soll die Wünsche und Bedürfnisse des Anwenders an den zu entwickelnden Algorithmus beschreiben. Weiter werden Use-Case-Beschreibungen erstellt, die Auslöser, Bedingungen und Voraussetzungen für die einzelnen Anwendungsfälle festhalten.

Aus diesen Dokumenten heraus wird ein Aktivitätsdiagramm erstellt. Hier werden neben den Akteuren die Daten dargestellt, die unter den Akteuren ausgetauscht werden. Weiter ist hieraus ersichtlich, welche Funktionen Rückantworten verlangen.

Mit Hilfe der Anwendungsfall-Erzählung, der Use-Case-Beschreibungen und des Aktivitätsdiagramms wird ein Klassendiagramm erstellt. Die Klassen lassen sich mittels [Nominalphrasierung](#) herausfiltern. Die Methoden werden durch die Verben verkörpert. Das Klassendiagramm dient zur Darstellung der statischen Aspekte des Algorithmus. Zudem werden hier die benötigten Methoden und Attribute dargestellt.

Darüber hinaus ist ein [Entwurf für das Aussehen der Benutzeroberfläche](#) erstellt worden (siehe Anhang, S. 65 ff.).

Die Erstellung der Anwendungsfall-Erzählung und der Use-Case-Beschreibungen war ziemlich zeitintensiv, sodass hier eine Stunde mehr Zeit benötigt wurde, als ursprünglich geplant.

geplante Zeit: 8 Stunden	benötigte Zeit: 9 Stunden	Differenz: + 1 Stunde
--------------------------	---------------------------	-----------------------

5.2 Testfallkatalog

Mit Hilfe des erstellten Pflichtenheftes, Absprachen mit dem Auftraggeber und der Erstellung der DV-Dokumentation können detaillierte [Test-Szenarien](#) verfasst werden (siehe Anhang, S. 78 ff.). Anhand der Funktionen und Aktionen, die in den UML-Diagrammen dargestellt werden, lassen sich Rückschlüsse auf die zu erwartenden Ergebnisse der Testfälle ziehen.

Da sowohl für einen [Blackbox](#)- als auch für einen [Whitebox-Test](#) konkrete Testfälle mit zu erwartenden Ergebnissen festgehalten werden müssen, wird dieselbe Testliste für beide Testarten herangezogen.

geplante Zeit: 1,5 Stunden	benötigte Zeit: 1,5 Stunden	Differenz: 0 Stunden
----------------------------	-----------------------------	----------------------

6. Implementierungsphase

6.1 Programmoberfläche

In der vorangegangenen Entwurfsphase ist ein erster [Entwurf für die Benutzer-Oberfläche](#) entworfen worden (siehe Anhang, S. 65 ff.). Nach kurzer Rücksprache mit dem Auftraggeber, Herrn Max Mustermann, kann die entworfene [GUI](#) für den [Prototyp](#) übernommen werden.

Richtlinien für Style und Aussehen der Programmoberfläche können außer Acht gelassen werden, da für die Umsetzung des Algorithmus zur Dachvollbelegung mit Photovoltaik-Modulen eine rasche Visualisierung erforderlich ist und die Anpassung an den [HS-GUI-Style-Guide](#) erst beim Einbau des Algorithmus in die Haupt-Anwendung „Grafische Erfassung“ (CAD) erfolgt.

geplante Zeit: 1 Stunde	benötigte Zeit: 1 Stunde	Differenz: 0 Stunden
-------------------------	--------------------------	----------------------

6.2 Umsetzung der Funktionalitäten im Quellcode

Nach Erstellung der Benutzer-Oberfläche können die Funktionen, die im Verlauf der Entwurfsphase bestimmt und beschrieben worden sind, implementiert werden. In den DV-Entwurfsdokumenten sind verschiedene Klassen und Methoden beschrieben worden, die im Quellcode umgesetzt und kommentiert wurden.

Bei der Implementierung wird unter anderem auf die Einhaltung der internen [Programmierrichtlinien](#) der Firma Hottgenroth GmbH & Co. KG geachtet. Der [kommentierte Quellcode](#) ist im Anhang (siehe Anhang, S. 88 ff.) zu finden.

Die Umsetzung der mathematischen Berechnungen, die dem Algorithmus zugrunde liegen, war zeitlich aufwändiger als im Entwurf angenommen. Daher wurde die geplante Implementierungszeit überschritten.

geplante Zeit: 13 Stunden	benötigte Zeit: 13,5 Stunden	Differenz: + 0,5 Stunden
---------------------------	------------------------------	--------------------------

7. Testphase

Nach Abschluss der Implementierungsphase wird der entstandene Algorithmus einem umfassenden Test unterzogen.

Den ersten Teil der Testphase bildet ein [Whitebox-Test](#), der von der Antragstellerin selber durchgeführt wurde. Der für die Testphase zugrundeliegende [Testfallkatalog](#) ist im Anhang zu finden (siehe Anhang, S. 78 ff.). Im Rahmen des Testvorganges fiel auf, dass beim Löschen der ganzen Zeichnung (d.h. der Dachfläche mit Modulen) nicht alle Werte, die vom Anwender eingegeben werden können, auch tatsächlich wieder auf die voreingestellten Werte zurückgesetzt werden. Dieser Fehler ist durch einen Rücksprung in die Implementierung behoben worden. Wiederholte anschließende Tests ergaben, dass nun das gewünschte Verhalten (das Einsetzen von [Default-Werten](#)) korrekt erfolgte.

In einem weiteren Testabschnitt wurde der Prototyp des Algorithmus an Herrn H. Meier (QS-Beauftragter) übergeben, der ebenfalls den Testfallkatalog zur Verfügung gestellt bekommen hatte, um einen [Blackbox-Test](#) durchzuführen. Dieser hat sich anhand des vorhandenen Testfallkataloges eigene Testszenarien konzipiert. Nach Aussage von Herrn Meier verlief der [Blackbox-Test](#) wie im Testfallkatalog beschrieben. Daher ist kein separates Testprotokoll erstellt worden. Die Zusicherung, dass alle Testfälle gegengeprüft worden sind und der Algorithmus wie erwartet funktioniert, wurde mündlich gegeben. Der von Herrn H. Meier selbst konzipierte Testfallkatalog wurde nicht ausgehändigt.

Das Durchgehen des Testfallkataloges beim [Whitebox-Test](#) nahm weniger Zeit in Anspruch als ursprünglich angenommen. Trotz wiederholten Test-Durchlaufs nach Behebung des gefundenen Fehlers ergab sich ein zeitlicher Gewinn von einer Stunde.

geplante Zeit: 5 Stunden	benötigte Zeit: 4 Stunden	Differenz: - 1 Stunde
--------------------------	---------------------------	-----------------------

Fehlerkorrektur:

Der beim [Whitebox-Test](#) aufgefallene Fehler konnte mittels Rücksprung in die Phase der Implementierung zügig behoben werden, da nur geringfügige Änderungen am Quellcode vorgenommen werden mussten. Daher wurde nicht die ganze für etwaige Fehlerbehebung einkalkulierte Zeit in Anspruch genommen.

geplante Zeit: 1 Stunde	benötigte Zeit: 0,5 Stunden	Differenz: - 0,5 Stunden
-------------------------	-----------------------------	--------------------------

8. Abschlussphase

8.1 Entwicklerdokumentation

Auf Grund der gewünschten Option den Algorithmus erweitern zu können, wurde eine [Entwicklerdokumentation](#) erstellt (siehe Anhang, S. 121 ff.). Dies soll projektfremden Entwicklern den Einstieg in den Quellcode erleichtern und eine Erweiterung der Funktionen ermöglichen. Aus diesem Grund werden hier die Methoden und der Algorithmus an sich detailliert beschrieben.

geplante Zeit: 2,5 Stunden	benötigte Zeit: 2,5 Stunden	Differenz: 0 Stunden
----------------------------	-----------------------------	----------------------

8.2 Benutzer-Hilfe

Um dem Benutzer des Prototypen eine Hilfestellung zu bieten, ist eine [Benutzer-Hilfe](#) verfasst worden (siehe Anhang, Seite 124 ff.). Das Projekt ist keine Endfassung in Bezug auf Bedienung und Aussehen der Benutzer-Oberfläche. Dennoch wurde eine Benutzerhilfe erstellt, die dem [Blackbox-Tester](#), Herrn H. Meier, als Handreichung zur Verfügung gestellt wurde.

geplante Zeit: 1 Stunde	benötigte Zeit: 1 Stunde	Differenz: 0 Stunden
-------------------------	--------------------------	----------------------

8.3 Soll-Ist-Zeitvergleich und Kosten-Nutzen-Analyse

Nach erfolgter Umsetzung des Projektes ist ein rückblickender [Soll-Ist-Zeitvergleich](#) durchgeführt worden (siehe Anhang, S. 130 ff.). Hierbei wurden die tatsächlich benötigten Zeiten den ursprünglich geplanten Zeiten gegenübergestellt. Es fällt auf, dass bei einigen Arbeitspaketen weniger Zeit benötigt wurde, als veranschlagt. Deshalb musste die einkalkulierte Pufferzeit nicht in Anspruch genommen werden. Somit wurde auch die vorgeschriebene Maximalzeit von 70 Stunden nicht überschritten.

Da sich eingesparte und zusätzlich benötigte Zeiten ausgeglichen haben, sind lediglich bei den Fragen zum Soll-Konzept und der Projektabnahme minimale Zeitveränderungen aufgetreten. So wurden 30 Minuten mehr von Auftraggeber Herrn Max Mustermann benötigt. Somit entstanden Gesamtkosten von 6.497,00 €. Eine detaillierte Aufstellung der tatsächlich entstandenen [Kosten](#) ist im Anhang zu finden (siehe Anhang, S. 130 ff.).

Der Nutzen, der durch die Entwicklung des Vollbelegungs-Algorithmus entsteht, ist monetär nicht messbar. Es entsteht zwar eine Zeitersparnis für den Anwender, da die Vollbelegung automatisch erfolgt, allerdings ist dieser Nutzen für den Verkauf der Anwendung nicht ausschlaggebend. Wichtiger für die Firma Hottgenroth GmbH & Co. KG ist die Kundenbindung, was durch die Weiterentwicklung von Funktionen und die Optimierung der Anwenderfreundlichkeit erreicht wird.

geplante Zeit: 1,5 Stunden	benötigte Zeit: 1,5 Stunden	Differenz: 0 Stunden
----------------------------	-----------------------------	----------------------

8.4 Qualitätsbericht

Als Abschluss des Projektes ist ein separater [Qualitätsbericht](#) erstellt worden (siehe Anhang, S. 133 ff.). Hierfür wurden die vorher festgelegten Ziele des Qualitätsplans anhand der festgelegten Meilensteine kontrolliert und geprüft. Über die verwendeten Maßnahmen hinaus sind in dem Qualitätsbericht auch Ergebnisse dokumentiert worden. Abschließend lässt sich sagen, dass alle Punkte des Qualitätsplans vollständig erfüllt worden sind.

geplante Zeit: 1,5 Stunden	benötigte Zeit: 1,5 Stunden	Differenz: 0 Stunden
----------------------------	-----------------------------	----------------------

8.5 Übergabe des Projektes an den Auftraggeber

Auf Grund der Terminvorgabe seitens der „Industrie- und Handelskammer Köln“ wurde das Projekt bereits vor dem vom Auftraggeber festgelegten Termin des 12.04.2013 fertiggestellt und dem Auftraggeber ausgehändigt. Am 28.03.2013 wurde das Projekt vollständig fertig gestellt und konnte mit allen erstellten Dokumenten an Herrn Max Mustermann übergeben werden. Das erstellte [Abnahmeprotokoll](#) ist im Anhang (siehe Anhang, S. 135 ff.) zu finden.

geplante Zeit: 0,5 Stunden	benötigte Zeit: 0,5 Stunden	Differenz: 0 Stunden
----------------------------	-----------------------------	----------------------

8.6 Projektreflexion

Der Algorithmus zur automatischen Vollbelegung von Dachflächen mit Photovoltaik-Modulen konnte gemäß den vom Auftraggeber bestimmten Anforderungen umgesetzt werden. Zeitliche Mehranforderungen konnten durch anderweitige zeitliche Einsparungen ausgeglichen werden.

Bezüglich des Quellcodes kann angemerkt werden, dass der Aufbau der Klassen unter Umständen verbessert werden könnte. Es wäre denkbar, eine Klasse „Dachfläche“ zu erstellen, von der die drei Formen (Rechteck, Trapez und Dreieck) jeweils eine abgeleitete Klasse sind. Da die Formen der Dachflächen im Endeffekt jedoch durch der Anwendung „Grafische Erfassung“ (CAD) vorgegeben wird, kann dieser Punkt vernachlässigt und auch verworfen werden.

Weiter fiel nach Abschluss der Implementierung auf, dass alternativ zu den verwendeten Checkboxes auch Radio-Button verwendet werden könnten. Vorteil der Radio-Button wäre gewesen, dass keine Absicherung notwendig gewesen wäre, die sicherstellt, dass auch wirklich nur eine Dachform ausgewählt worden ist.

Mitte April 2013 soll der Algorithmus in die Anwendung der „Grafischen Erfassung“ (CAD) eingebaut und in der nächsten Auslieferung der Software „PV Simulation“ als Funktion verfügbar sein.

9. Anlagenverzeichnis

1. Projektantrag	18
2. Ist-Analyse	30
3. Soll-Konzept	34
4. Vorgehensmodell	39
5. Durchführbarkeitsanalyse	41
6. Pflichtenheft	43
7. Arbeitspakete	49
8. Zeit- und Meilensteinplan	53
9. Projektstrukturplan	57
10. Ressourcen- und Kostenplan	59
11. Qualitätsplan	62
12. DV-Entwurfsdokumente	65
13. Testfallkatalog und Testfallprotokoll	78
14. Kommentierter Quellcode	88
15. Entwicklerdokumentation	121
16. Benutzer-Hilfe	124
17. Soll-Ist-Vergleich und Kosten-Nutzen-Analyse	130
18. Qualitätsbericht	133
19. Abnahmeprotokoll	135
20. Glossar	137
21. Persönliche Erklärung	142



Projektantrag

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau

Datum: 04.02.2013

Elvira Musterfrau
Prüflingsnummer: 4711
Azubi-Ident-Nummer: 2101117

Antrag für die betriebliche Projektarbeit

Antrag: Elvira Musterfrau
Antrag vom: 21.12.2012

Prüflingsnummer: 4711

Antrag für die betriebliche Projektarbeit

Elvira Musterfrau

Ausbildungsberuf: Fachinformatiker/-in Fachrichtung Anwendungsentwicklung

Antragsteller(in):

Ausbildungsbetrieb:

Straße: Von-Hünefeld-Str. 3
PLZ, Ort: 50829 Köln
E-Mail: Audsbilder.Mail@web.de

Prüflingsnummer:	Abschlussprüfung:	Antragsdatum:
4711	S2013	

Inhalt

1	Projektbezeichnung.....	22
1.1	Kurze Projektbeschreibung	22
2	Projektumfeld.....	22
2.1	Organisatorisches Projektumfeld	22
2.2	Technisches Projektumfeld	24
2.3	Projektbeteiligte	24
3	Projektplanung einschließlich Zeitplan	24
3.1	Netzplan	26
4	Geplante Dokumentationen zur Projektarbeit.....	27

1 Projektbezeichnung

Auftrag / Teilauftrag:

Entwicklung eines Algorithmus zur automatischen Vollbelegung von Dächern mit Photovoltaik-Modulen (PV-Modulen) für die Anwendung „CAD – PV Simulation“.

1.1 Kurze Projektbeschreibung

Im Rahmen der betrieblichen Projektarbeit wird bei der Firma Hottgenroth GmbH & Co. KG für die Softwareanwendung [CAD](#) („Grafische Erfassung“) ein Algorithmus entwickelt, der eine flexibel wählbare automatische Vollbelegung verschiedener Dachformen mit [Photovoltaik-Modulen \(PV-Modulen\)](#) ermöglicht. CAD ist wiederum in die Anwendung „PV Simulation“ eingebettet, mit der PV-Anlagen geplant werden können.

Dieser Algorithmus bietet eine Funktion, mit der die optimale Belegung eines Daches mit PV-Modulen ermittelt werden kann. Diese Vollbelegung ist für diverse Dachformen (rechteckig, trapezförmig, dreieckig) für verschieden wählbare Ausgangspunkte (oben links, unten links, oben rechts, unten rechts) vorgesehen. So hat der Kunde zur bisherigen manuellen Erstellung von Modulfeldern auf Dächern eine automatisierte Routine, die zudem die Gesamtanzahl der Module ausgibt.

Der Algorithmus für die Vollbelegung wird keine eigenständige Anwendung sein, sondern als Modul in die Anwendung „PV Simulation“ mit der Komponente CAD („Grafische Erfassung“) integriert.

Zunächst wird ein Prototyp für die zweidimensionale Visualisierung des Vollbelegungs-Algorithmus entwickelt, bevor die Funktion in das vorhandene CAD-Programm implementiert wird.

2 Projektumfeld

2.1 Organisatorisches Projektumfeld

Die Firma Hottgenroth GmbH & Co. KG wurde Mitte der 80er Jahre gegründet. Seit 1986 vertreibt das Unternehmen branchenspezifische Software-Anwendungen für das Gewerbe der Schornsteinfeger und Energieberater.

Die Hottgenroth GmbH & Co. KG stellt ihren Kunden mit derzeit rund 140 Angestellten nicht nur Software-Lösungen zur Verfügung, sondern bietet einen Rundum-Service durch das Angebot von weiteren Dienstleistungen (u.a. Schulungen, Online-Portal, Support) und Hardware-Lösungen.

Die schrittweise Erweiterung der Produktpalette ist im Folgenden stichpunktartig dokumentiert:

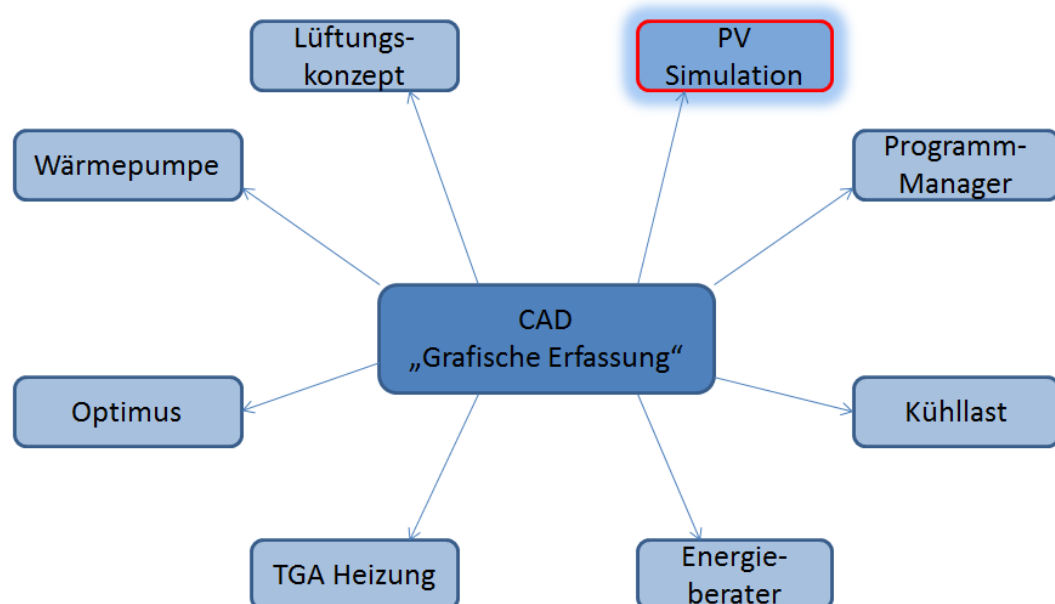
- **2002:** Übernahme des Unternehmens ETU Software GmbH mit der dort entwickelten TGA Software-Anwendung für Haus- und Gebäudetechnik.
- **2006:** Entwicklung einer Datenerfassungs-Software.

- **seit 2007:** Erstellung von kaufmännischen Software-Lösungen.
- **Ende 2008:** Entwicklung einer CAD-Software („Grafische Erfassung“).
- **Mitte 2010:** Umstrukturierung und Neu-Entwicklung der Software „PV Simulation“.

Ende 2008 wurden die ersten Weichen zur Entwicklung einer „3D-Grafischen Erfassung“ (CAD) gestellt, die seitdem immer komplexer und in immer mehr Anwendungen der Firma Hottgenroth Software GmbH & Co. KG eingebettet wird.

Mitte 2010 entwickelte die Firma Hottgenroth GmbH & Co. KG eine neue Version der Software „PV Simulation“. Hier wurde die „Grafische Erfassung“ eingebettet, mit deren Hilfe Gebäude grafisch erfasst und mit Modulen belegt werden können. Anschließend werden in „PV Simulation“ die angelegten Module berechnet und verschaltet.

Anwendung kann der Algorithmus nicht nur in „PV Simulation“ finden, sondern je nach Ausstattung mit Lizenzen auch beispielsweise in der hauseigenen Energieberater-Software oder der Software der TGA-Heizung. Das von der Firma Hottgenroth GmbH & Co. KG entwickelte „Datenmodell“ erlaubt nahezu der gesamten Produktpalette des Unternehmens einmal eingeebene und erfasste Daten (z.B. Bauteile, U-Werte, Eigenschaften) mit mehreren Anwendungen aufzurufen, zu verändern und weiter zu geben. Eine Übersicht der CAD-nutzenden Anwendungen ist hier beigefügt.



- **CAD:** „computer-aided design“; Software zur Erstellung von Gebäuden mit 3D-Darstellung zur weiteren Verarbeitung.
- **PV Simulation:** Photovoltaik Simulation zur Berechnung und Konzeptionierung von Photovoltaik-Anlagen.

- **Programm-Manager:** Programm zur gemeinsamen Verwaltung und Koordination von Projekten zwischen verschiedenen Anwendungen der Hottgenroth-Produktpalette umsetzt.
- **Kühllast:** Software zur Berechnung der Kühllasten von Gebäuden.
- **Energieberater:** Anwendung zur energetischen Planung und Bewertung von Wohngebäuden nach aktuellen Normen (z.B. EnEV; DIN 4108-6; 18599).
- **TGA Heizung:** Software zum Auslegen und Berechnen von Heizungsanlagen (TGA: Technische Gebäude-Ausrüstung).
- **Optimus:** Anwendung zur Berechnung und zum Nachweis des hydraulischen Abgleichs von Heizungsanlagen im Gebäudebestand.
- **Wärmepumpe:** Software zur Optimierung und Berechnung von Wärmepumpen-Anlagen.
- **Lüftungskonzept:** Anwendung zur Ausgabe von Luftvolumenströmen sowie zur Auslegung von Lüftungsanlagen.

Die obigen Definitionen sind in Anlehnung an den „Gesamtkatalog 2013“ der Firma Hottgenroth Software GmbH & Co. KG entstanden.

2.2 Technisches Projektumfeld

Die Entwicklung des Algorithmus zur automatischen Vollbelegung von Dächern mit PV-Modulen erfolgt auf einem Intel Core 2 Rechner. Dieser läuft mit dem Betriebs-System Windows Vista (inkl. Service Pack 2) und der Entwicklungs-Umgebung Microsoft Visual Studio 2010, womit ein Großteil der Anwendung CAD („Grafische Erfassung“) entwickelt worden ist. Die Entwicklung des Algorithmus erfolgt in der Programmiersprache C#.

Die textliche Dokumentation wird mit Microsoft Office Word 2010 erstellt. Die erforderlichen Prozesse werden mit Hilfe von Microsoft Office Visio 2007 grafisch dargestellt.

2.3 Projektbeteiligte

Folgende Personen sind an diesem Projekt beteiligt:

Auftraggeber des Projektes:	Geschäftsführung
Projektverantwortlicher / Ansprechpartner:	F. Schulze
Ausbilderin:	Anna Ausbilder
Testing / Qualitätssicherung:	H. Meier
Entwicklung des Projektes:	Elvira Musterfrau

3 Projektplanung einschließlich Zeitplan

Für die Durchführung dieses Projektes werden eine Gesamtstundenzahl von 67 Stunden sowie eine zusätzliche Pufferzeit von 3 Stunden angesetzt. Die benötigten Stunden je Projektphase sind der unten stehenden detaillierten Aufstellung zu entnehmen:

1) Definitions- und Analyse-Phase (10 Stunden)

Durchführung der IST-Analyse:	1 Stunde
Entwicklung des SOLL-Konzeptes:	2 Stunden
Durchführbarkeits-Analyse:	1 Stunde
Erstellung des Pflichtenheftes:	6 Stunden

2) Planungs-Phase (9 Stunden)

Identifizierung und Definition der Arbeitspakete:	1 Stunde
Erstellung von Projektstruktur-, Zeit-, Meilenstein-, Kosten- und Ressourcenplan:	5 Stunden
Erstellung des Qualitätsplans und Testfallkataloges:	3 Stunden

3) Durchführungs-Phase (31 Stunden)

Planen und Erstellen der DV-Entwurfsskizzen (UML):	8 Stunden
Entwurf der Programm-Oberfläche:	3 Stunden
Programmierung und Implementierung des Algorithmus:	14 Stunden
Whitebox-Test:	5 Stunden
Blackbox-Test (nicht von der Antragstellerin durchgeführt)	
Fehlerkorrektur:	1 Stunde

4) Abschluss-Phase (3,5 Stunden)

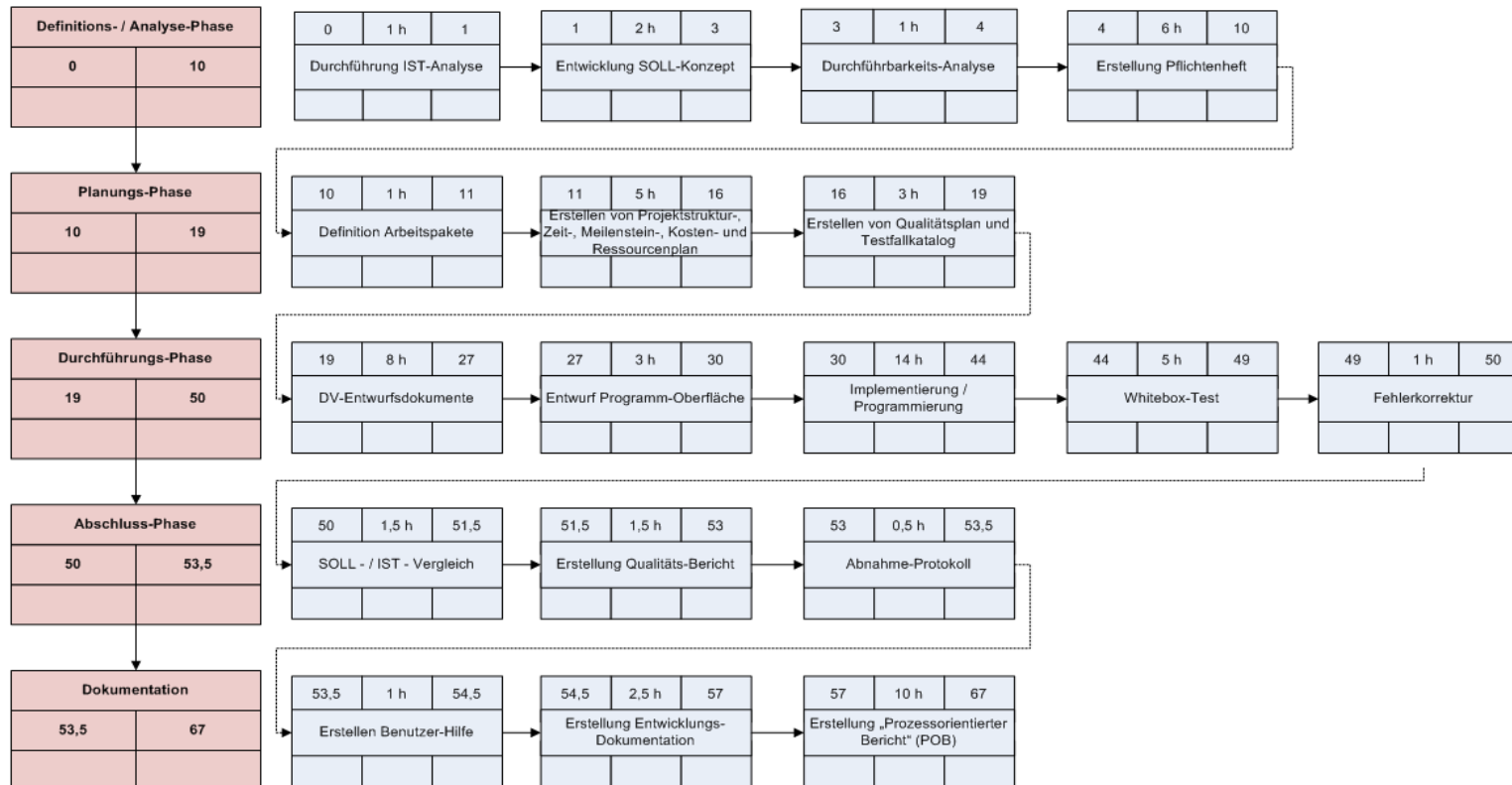
SOLL- / IST-Vergleich:	1,5 Stunden
Erstellung des abschließenden Qualitäts-Berichtes:	1,5 Stunden
Abnahme-Protokoll:	0,5 Stunden

5) Dokumentation (13,5 Stunden)

Erstellung des prozessorientierten Projektberichtes (POB):	10 Stunden
Erstellung der Entwicklungs-Dokumentation:	2,5 Stunden
Erstellen einer Benutzer-Hilfe:	1 Stunde

Geplante Arbeitszeit:	67 Stunden
Puffer-Zeit:	3 Stunden
Maximale Gesamtzeit:	70 Stunden

3.1 Netzplan



Legende:

Aufgabenname		Früher Beginn	Dauer	Frühe Fertigstellung
Geplanter Anfang	Geplanter Ablauf	Aufgabenname		
Spätester Anfang	Spätester Abschluss	Später Beginn	Pufferzeit	Spätes Ende

Da es sich um den geplanten Zeitablauf handelt, sind hier noch keine Pufferzeiten bzw. Verzögerungen eingetragen.

4 Geplante Dokumentationen zur Projektarbeit

1) Definitions-Phase

IST-Analyse
SOLL-Konzept
Durchführbarkeits-Analyse
Pflichtenheft

2) Planungs-Phase

Arbeitspakete
Zeitplan
Meilensteinplan
Projektstrukturplan
Ressourcenplan
Kostenplan
Qualitätsplan
Testfallkatalog

3) Durchführungs-Phase

DV-Entwurfsdokumente (UML-Diagramme)
Quellcode-Dokumentation (kommentiert)
Testprotokolle
Fehlerprotokoll

4) Abschluss-Phase

Kosten-Nutzen-Analyse
SOLL- / IST-Vergleich
Qualitätssicherungs-Bericht
Abnahmeprotokoll

5) Dokumentation

Entwicklungs-Dokumentation
Prozessorientierter Projektbericht
Benutzer-Hilfe

Sollten Dokumente eingereicht werden, die nicht vom Antragsteller selber erstellt worden sind, so werden diese entsprechend gekennzeichnet.

Antragsbestätigung der IHK

Von: pao@koeln.ihk.de
Antwort an: pao@koeln.ihk.de
An: Elvira.Musterfrau@web.de
Datum: 26. Januar 2013 17:39
Betreff: IHK Köln. Projektantrag genehmigt. Fachinformatiker/-in(Berufsnr.: 5120)
Musterfrau, Elvira (Azubi-Ident: 0000815).
Gesendet von: koeln.ihk.de

Sehr geehrte Frau Musterfrau,

Ihr Antrag für die/den betriebliche/n Projektarbeit/Fachaufgabe/Arbeitsauftrag wurde von der IHK Köln dem zuständigen Prüfungsausschuss zur weiteren Bearbeitung zugeteilt.

Der Prüfungsausschuss hat den Antrag genehmigt.

Bitte beachten Sie folgende Hinweise:

Hinweis zu Ihrem Antrag:

Die Pufferzeit ist mit 3 Stunden etwas knapp angesetzt. Bitte beachten Sie, dass die Gesamtprojektzeit von 70 Stunden in keinem Fall überschritten werden darf!

Allgemeine Hinweise zur Erstellung der Projektdokumentation:

Das Kernstück der Dokumentation ist der 'prozessorientierte Projektbericht'. In ihm ist Ihr Vorgehen im Projekt mit Vorgehensmodell, geplante und benötigte Prinzipien, Methoden, Techniken und Werkzeuge und eine kurze Zusammenstellung aller in den einzelnen Phasen erzielten, wesentlichen Ergebnisse zu beschreiben. Alle relevanten Ergebnisse aus den einzelnen Projektphasen und ggf. Ihnen vor Beginn der Projektarbeit zur Verfügung gestellten Dokumente (z.B. Lastenheft, Pflichtenheft,...) sind der Projektdokumentation beizufügen, so dass es für einen sachkundigen Dritten möglich ist, die Produktentwicklung nachzuvollziehen.

Bitte beachten Sie bei der Erstellung Ihrer Projektarbeit die Ausführungen, Hinweise und Bearbeitungsvorschläge im Kapitel 4 der Broschüre 'Fachinformatiker/-in Anwendungsentwicklung - Einführung zur Abschlussprüfung' auf der IHK-Internetseite: www.ihk-koeln.de, Dokumentennr. 318. Sofern an Ihrem Projekt weitere Personen beteiligt waren, sind deren bzw. gemeinsam erreichte Arbeitsergebnisse in Ihrem Projektbericht und dem Anhang deutlich zu kennzeichnen. Dies gilt auch für Arbeitsergebnisse, die Sie außerhalb der hier beschriebenen Projektzeit erzielt haben und die Sie in diesem Projekt wieder verwenden möchten. Es können nur Teile gewertet werden, die von Ihnen alleine während der beschriebenen Projektzeit erstellt wurden.

Vertrauliche oder datenschutzrelevante Daten sind als solche zu kennzeichnen. Sofern es aus datenschutzrechtlichen Gründen erforderlich ist, können Teile der Information durch Schwärzen etc. unkenntlich gemacht werden. Es ist nicht gestattet, verfälschte (Pseudo-) Namen, unrichtige Datumsangaben etc. zu verwenden. Die Nachvollziehbarkeit einzelner Prozessschritte muss unbedingt gewährleistet sein. Dokumente mit rechtlich verbindlichem Charakter sind mit entsprechender Unterschrift

versehen eingescannt beizufügen.

Bitte beachten Sie unbedingt den Termin:

08.04.2013 12:00 Uhr

Bis zu diesem Termin müssen Sie Ihre/n Dokumentation/Report im pdf-Format von max. 5 MB in das IHK-Portal eingestellt haben.

Der vorstehende Termin ist eine Ausschlussfrist. Die Nichteinhaltung dieser Frist sowie die Nichtbeachtung der formalen Vorgaben und Hinweise führt zu einer Wertung der/des Dokumentation/Reports mit "0 Punkte/Nicht bestanden".

Wir wünschen Ihnen viel Erfolg.

Mit freundlichen Grüßen

Industrie- und Handelskammer zu Köln
Im Auftrag

Nicole Wüstefeld
Geschäftsbereich Aus- und Weiterbildung
Tel. [+49 221 1640-624](tel:+492211640624)
Fax [+49 221 1640-649](tel:+492211640649)
E-Mail: nicole.wuestefeld@koeln.ihk.de
Internet: <http://www.ihk-koeln.de>



IST-Analyse

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau

Datum: 04.02.2013

1. Verwendung der „Grafischen Erfassung“ (CAD)

Die „Grafische Erfassung“ ist eine der zentralen Komponenten, die von zahlreichen Anwendungen der Hottgenroth-Produktpalette genutzt wird (siehe Abb. 1):

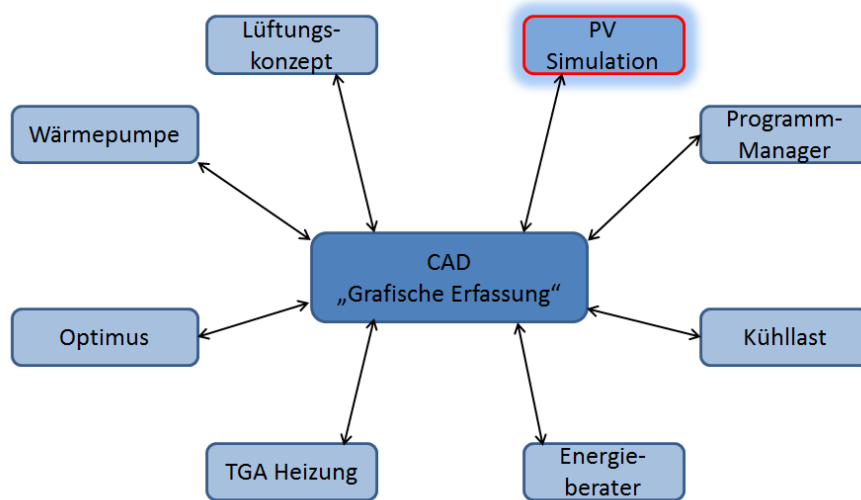


Abb. 1: Übersicht über das Zusammenspiel der Programme

- **CAD:** „computer-aided design“; Software zur Zeichnung von Gebäuden in 2 und 3 Dimensionen und zur Bauteilzerlegung als Grundlage für die Verwendung durch weitere Anwendungen.
- **PV Simulation:** Photovoltaik-Simulation zur Berechnung und Konzeptionierung von Photovoltaik-Anlagen.
- **Programm-Manager:** Programm zur gemeinsamen Verwaltung und Koordination von Projekten zwischen verschiedenen Anwendungen der Hottgenroth-Produktpalette.
- **Kühllast:** Software zur Berechnung der Kühllasten von Gebäuden.
- **Energieberater:** Anwendung zur energetischen Planung und Bewertung von Wohngebäuden nach aktuellen Verordnungen und Normen (z.B. Energieeinspar-Verordnung (EnEV) ; DIN V 4108-6; DIN V 4701-10; DIN V 18599).
- **TGA Heizung:** Software zum Auslegen und Berechnen von Heizungsanlagen (TGA: Technische Gebäude-Ausrüstung).
- **Optimus:** Anwendung zur Berechnung und zum Nachweis des hydraulischen Abgleichs von Heizungsanlagen im Gebäudebestand und Neubau.
- **Wärmepumpe:** Software zur Optimierung und Berechnung von Wärmepumpen-Anlagen.
- **Lüftungskonzept:** Anwendung zur Ausgabe von Luftvolumenströmen sowie zur Auslegung von Lüftungsanlagen.

Die obigen Definitionen sind in Anlehnung an den „Gesamtkatalog 2013“ der Firma Hottgenroth Software GmbH & Co. KG entstanden.

2. Bisherige Handhabung der Modul-Belegung in „PV Simulation“

In die Software „PV Simulation“ ist die Anwendung CAD zur grafischen Erfassung von Gebäuden eingebettet. Es können Gebäude gezeichnet werden, die zur Erfassung der Gebäudedaten an „PV Simulation“ übergeben werden. Nach der Wahl eines Photovoltaik-Moduls mit Hersteller, Bezeichnung und damit Geometrie, wird in die „Grafische Erfassung“ gewechselt.

Hier werden die Dächer des erfassten Gebäudes manuell mit Einzelmodulen oder mit Modulfeldern belegt. Diese können auf alle Dachformen gelegt werden. Modulfelder lassen sich derzeit nur in einem rechteckigen Feld auf Dachflächen legen. Dreieckige Dachflächen können mit der derzeitigen Lösung des Aufziehens von Modul-Feldern nicht optimal belegt werden, da eine optimale Ausnutzung der Fläche nicht gewährleistet ist.

2.1 Dachbelegung mit Einzelmodulen

Wird nach der Erfassung des Grundrisses, der Auswahl eines Modul-Herstellers und eines Modul-Typen in „PV Simulation“ die „Grafische Erfassung“ (CAD) aufgerufen, lassen sich Einzelmodule oder manuelle Modulfelder anlegen.

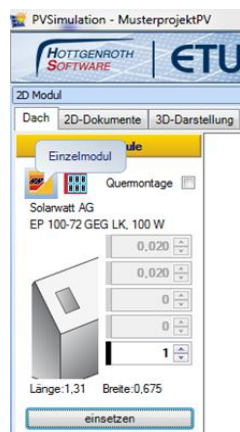


Abb. 2: Wahl Einzelmodul

Wird die Option des „Einzelmoduls“ gewählt (siehe Abb. 2), kann der Anwender die vorhandene Dachfläche sukzessive mit Modulen belegen.

Die Zeichenfunktion wird durch Betätigung des Buttons „einsetzen“ aktiviert, wobei die Daten von Hersteller, Modul-Bezeichnung und Abmessungen des Moduls an der Oberfläche angezeigt werden. Durch Klicken auf die Dachfläche können Module abgelegt werden (siehe Abb. 3).

Da die Abstände zwischen den Modulen im Modus des „Einzelmoduls“ nicht festgelegt werden können, muss der Anwender selber darauf achten, dass die Abstände möglichst gleichmäßig sind.

Eine optimale Ausnutzung und gleichmäßige Belegung der vorhandenen Dachfläche ist bei der Belegung mit Einzelmodulen nicht möglich.

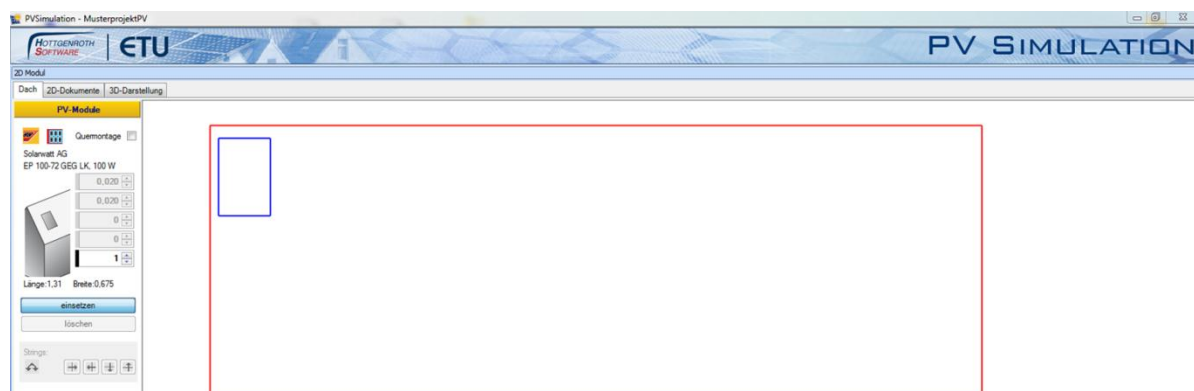


Abb. 3: Zeichnen eines Einzelmoduls

2.2 Dachbelegung mit Modulfeldern

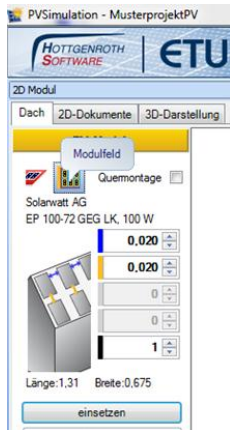


Abb. 4: Modulfeld-
Modus wählen

Wählt der Anwender die Option „Modulfeld“ (siehe Abb. 4), so sind auch hier die Daten des zuvor gewählten Modul-Herstellers, die Modul-Bezeichnung und die Abmessungen des Moduls einsehbar.

Weiter kann der Benutzer die Abstände zwischen den Modulen in horizontaler und in vertikaler Richtung bestimmen. Über den Button „einsetzen“ wird die Zeichenfunktion aktiviert.

Nach Markierung des Feldes auf der zu belegenden Dachfläche wird mit einem farbigem Rechteck die Lage des Modulfeldes festgelegt (siehe Abb. 5).



Abb. 5: Modulfeld manuell aufziehen

Ist die gewünschte Modulfeldgröße erreicht, wird die Konstruktion mit einem weiteren Mausklick auf die Zeichenunterlage abgeschlossen. Anhand der Modul-Daten (Abmessungen, Abstände) wird die maximal mögliche Anzahl für dieses Modulfeld berechnet. Das Ergebnis wird grafisch entsprechend angezeigt (siehe Abb. 6).

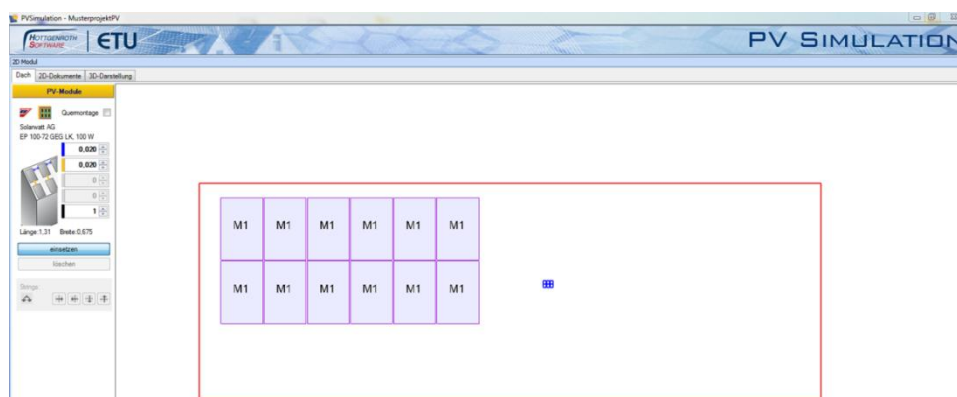


Abb. 6: Ansicht Dach mit Modulfeld

Bei diesen beiden Arten der Dachbelegung kann die vorhandene Fläche nicht optimal ausgenutzt werden. Aus diesem Grund soll ein Algorithmus erstellt werden, der diese Vollbelegung von Dachflächen mit optimaler Auslegung automatisch umsetzt.



SOLL-Konzept

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau

Datum: 05.02.2013

1. Verwendungszweck und Einsatzbereich

Der Algorithmus zur automatischen Vollbelegung von Standard-Dachformen mit Photovoltaik-Modulen soll die optimale Ausnutzung der Dachflächen ermöglichen.

Festgelegte Standard-Dachflächen der unterschiedlichen Dachformen sind:

- Rechteck-Dachfläche
- Dreieckige Dachfläche
- Trapezförmige Dachfläche

Der Prototyp soll unter Berücksichtigung von Modul-Maßen, sowie Randabständen zu den Dachkanten und Verlege-Abständen zwischen den Modulen eine optimale Flächenausnutzung erlauben. [Sperrflächen](#) wie Dachfenster, Dachgauben, Schornsteine oder Satelliten-Anlagen werden noch nicht berücksichtigt.

Der Startpunkt der Vollbelegung ist aus einer vorgegebenen Liste auszuwählen. So soll der Anwender die gewählte Dachfläche von

- oben links,
- oben rechts,
- unten links,
- unten rechts

mit Photovoltaik-Modulen belegen können.

Abstände zu den Dachkanten und zwischen den Modulen sind vom Anwender einstellbar. Zur Kontrolle der eingegebenen Abstände ist im Rahmen der Prototyp-Anwendung eine Visualisierung des Abstandes zu den Dachkanten mit Hilfe einer Umrandung zulässig.

Hier muss angemerkt werden, dass der erwähnte [Prototyp](#) in diesem Fall nicht mit dem [Vorgehensmodell](#) „[Prototyping](#)“ gleichzusetzen ist. Der zu erstellende Prototyp beinhaltet einen vollständigen Algorithmus, der mittels einer provisorisch entworfenen Oberfläche visualisiert wird.

1.1 Einsatzvoraussetzungen

Der zu entwickelnde Algorithmus, der in die Anwendung „Grafische Erfassung“ (CAD) integriert wird, soll später unter den für „PV Simulation“ festgelegten System-Voraussetzungen laufen.

Festgelegte Minimal-Anforderungen an die Rechner der Endanwender sind:

- Betriebssystem: Microsoft Windows XP oder höher
- Arbeitsspeicher: 1 GB RAM oder mehr
- Festplattenkapazität: 500 MB
- Grafikkarte: 3D-Grafikkarte mit DirectX9.0c; 128 MB interner Speicher
- Bildschirmauflösung: 1024 x 768 Pixel oder höher mit mindestens 16-Bit Farbtiefe

1.2 Kompatibilität

Der Algorithmus zur automatischen Dachvollbelegung ist eine zusätzliche Funktion in der Anwendung „Grafische Erfassung“, die selber wiederum in „PV Simulation“ eingebettet ist.

Bei dem Projekt handelt es sich um keine eigenständige Anwendung. Somit gelten für die Kompatibilität die gleichen Regelungen, die für die „Grafische Erfassung“ und „PV Simulation“ bestehen. Die oben genannten Programme müssen mit jeder neuen Version, die erstellt wird, kompatibel sein. Es muss zudem sichergestellt werden, dass Projekte, die mit älteren Versionen erstellt wurden, mit neueren Versionen kompatibel sind.

2. Systemspezifikation

2.1 Daten des Algorithmus

Damit der zu entwickelnde Algorithmus die Anzahl der PV-Module und ihre Lage berechnen und zeichnerisch darstellen kann, sind die Koordinaten bzw. Maße der Dachflächen, der Abstand der Module zu den Dachkanten, der Abstand zwischen den einzelnen Modulen in vertikaler und horizontaler Richtung, sowie die Maße der Module selber notwendig.

Für den Prototyp des Algorithmus sollen die Dachflächen individuell gezeichnet und die fehlenden Eckpunkte berechnet und zwischengespeichert werden. Die Abstände zu den Dachkanten und zwischen den Modulen sollen als feste Zahlenwerte in einer Eingabemaske eingetragen werden.

Zum besseren Verständnis der verwendeten Größen und Bezeichnungen ist eine Skizze einer trapezförmigen Dachfläche beigefügt (siehe Abb. 7).

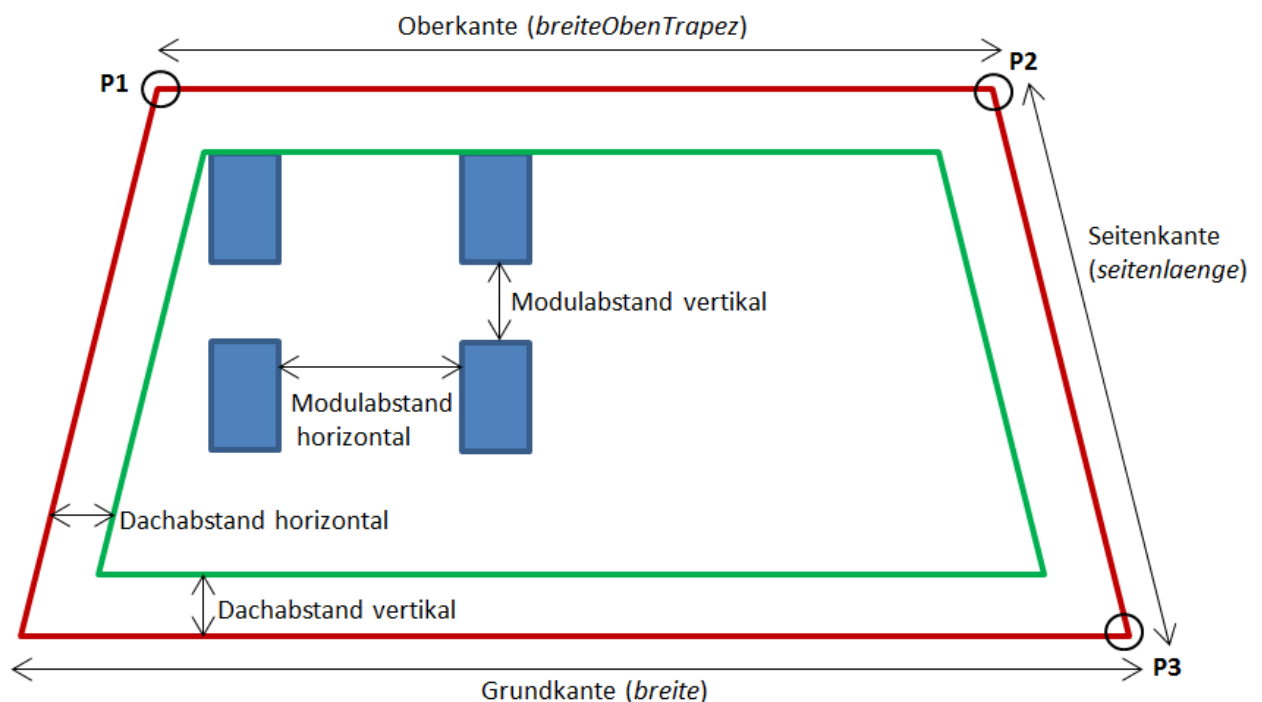


Abb. 7: Benennung von Größen

Name der Variablen	Beschreibung
punkt1X	x-Wert des ersten Klickpunkts merken (für Dreieck, Rechteck und Trapez)
punkt1Y	y-Wert des ersten Klickpunkts merken (für Dreieck, Rechteck und Trapez)
punkt2X	x-Wert des zweiten Klickpunkts merken (für Dreieck, Rechteck und Trapez)
punkt2Y	y-Wert des zweiten Klickpunkts merken (für Dreieck, Rechteck und Trapez)
punkt3X	x-Wert des dritten Klickpunkts merken (für Trapez)
punkt3Y	y-Wert des dritten Klickpunkts merken (für Trapez)
klickzaehler	Zählt die Anzahl der angeklickten Punkte für die Dachflächen hoch. Bei Rechteck- und Dreieck-Dächern werden 2 Punkte benötigt, bei Trapez-Dächern 3 Punkte.
breite	Länge der Grundkante des gezeichneten Daches.
breiteObenTrapez	Länge der Oberkante des gezeichneten Daches (nur bei Trapez-Dächern erforderlich)
seitenlaenge	Länge der (schrägen) Seitenkanten des gezeichneten Daches (bei Dreieck- und Trapez-Dächern notwendig)

2.2 Programmablauf

Um den zu entwickelnden Prototypen zur automatischen Vollbelegung von Dachflächen mit Photovoltaik-Modulen verwenden zu können, muss der Anwender zuerst die gewünschte Dachform auswählen und anschließend mittels Mausklicks die Eckpunkte der Dachflächen skizzieren. Die Dach-Konturen werden nach Betätigen der Schaltfläche **„Dachfläche zeichnen“** gezeichnet. Gleichzeitig werden die Abmessungen des Daches berechnet und angezeigt.

Anschließend kann der Anwender die Größe der Photovoltaik-Module sowie den vertikalen und horizontalen Abstand zwischen den Modulen angeben. Sollten hier keine Angaben getätigt werden, sollen [Default-Werte](#) (voreingestellte Werte) verfügbar sein.

Des Weiteren sind Werte für den Abstand der Module zu den Dachkanten voreingestellt, die vom Anwender jedoch individuell geändert werden können. Auch der Ausgangspunkt, von wo aus die Module auf das Dach gelegt werden sollen (z.B. oben links, etc.), soll eingestellt werden können.

Wird die Schaltfläche **„Module zeichnen“** gewählt, sollen die Module beginnend beim vorgesehenen Startpunkt auf die Dachfläche gelegt werden.

In der Anwendung „PV Simulation“ mit „Grafischer Erfassung“ (CAD) müssen nach Einbau des Algorithmus in die Hauptanwendung die Eingabeoptionen für das Dach und die Module wegfallen. Diese Informationen über Dach-Geometrie und Modul-Daten werden durch die Anwendungen „PV

Simulation“ (Modul-Daten) und die „Grafische Erfassung“ (Dach-Geometrie) zur Verfügung gestellt. Möglichkeiten zur Eingabe der Abstände zwischen den Modulen und zu den Dachkanten müssen über die „Grafische Erfassung“ (CAD) eingebbar sein.



Vorgehensmodell

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau

Datum: 25.03.2013

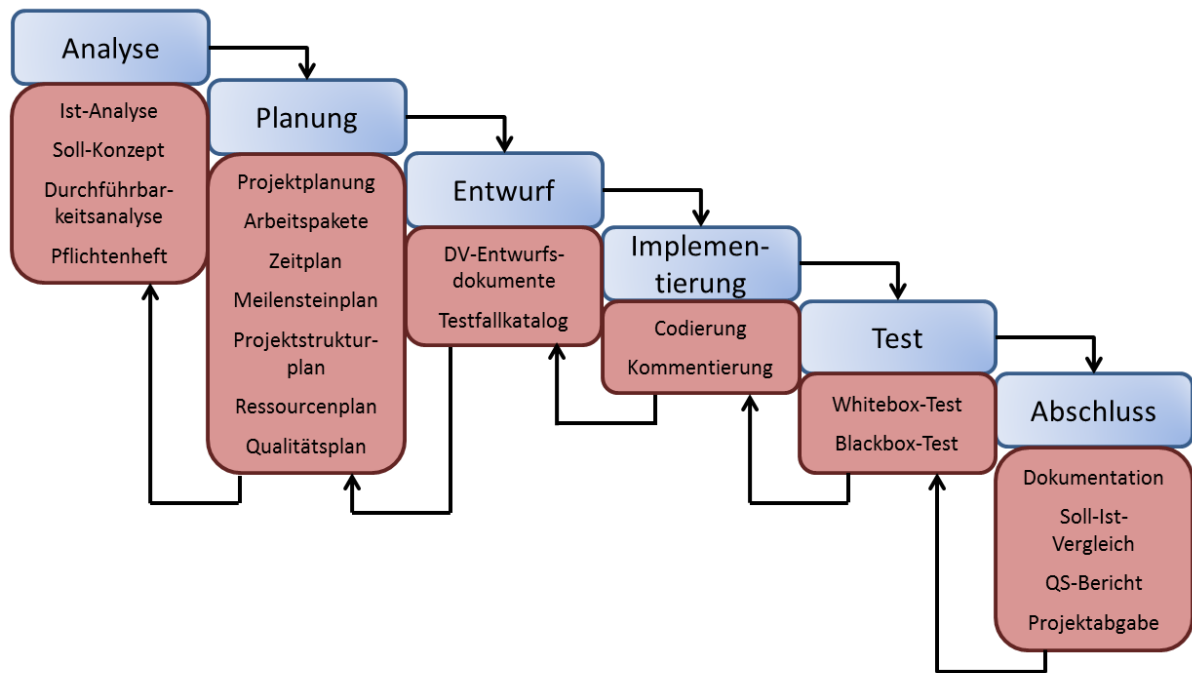


Abb. 8: erweitertes Wasserfallmodell

Für das vorliegende Projekt wurde das angepasste [erweiterte Wasserfallmodell](#) verwendet (siehe Abb. 8). Im Gegensatz zum [einfachen Wasserfallmodell](#) ist die erweiterte Version sehr dynamisch und flexibel. Sollten im Lauf des Projektes Engpässe oder Verzögerungen auftauchen, kann beim erweiterten Wasserfallmodell immer in die vorherige Phase gewechselt werden, um an dieser Stelle nachzuarbeiten.

Zudem sind die Phasen im verwendeten [Vorgehensmodell](#) untereinander klar abgegrenzt und werden jeweils mit begleitenden qualitätssichernden Maßnahmen (z.B. [Reviews](#), Tests, u.a.) abgeschlossen, bevor mit der darauffolgenden Phase des Projektes begonnen werden kann.

Im Folgenden wird ein Beispiel angeführt, wie die Möglichkeit des Rücksprungs genutzt werden kann:

Das Projekt ist bereits fertig implementiert, in kommentierten Quellcode umgesetzt und an die Test-Abteilung zur Qualitätssicherung weiter geleitet worden. Hier fällt nun ein Fehler auf, der behoben werden muss.

Jetzt besteht die Möglichkeit, von der Phase „Test“ in die Phase „Implementierung“ zurück zu wechseln, um den Fehler zu beheben und entsprechende Kommentare im Quellcode zur Kenntlichmachung einzufügen.

Anschließend kann die Implementierungs-Phase abgeschlossen werden. Es ist jedoch darauf zu achten, dass die Test-Phase jetzt von Vorne beginnt und das Projekt nochmals vollständig getestet werden muss, auch wenn alle anderen Test-Szenarien ohne Beanstandung erfüllt worden sind.



Durchführbarkeitsanalyse

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau

Datum: 06.02.2013

1. Machbarkeit

Der Algorithmus zur automatischen Vollbelegung von Dächern mit Photovoltaik-Modulen ist die Neuentwicklung einer Funktion, die der besseren Handhabung der Anwendung „PV Simulation“ dienen soll.

Die in Gesprächen ermittelten Funktionalitäten erscheinen als umsetzbar. Zudem ist nur eine zweidimensionale Darstellung der vom Algorithmus errechneten Ergebnisse verlangt.

Somit kann festgehalten werden, dass die technische Machbarkeit für dieses Projekt ohne Einschränkungen gegeben ist.

2. Projektrisiko

Auf Grund der reellen Machbarkeit hält sich das Risiko in Bezug auf Kosten und das Projekt an sich eher gering.

Zusätzliche technische (z.B. Rechner, Monitor, u.a.) oder personelle Ressourcen (z.B. weitere Team-Mitglieder) müssen nicht weiter eingeplant werden, da bereits alle Arbeitsplätze mit Rechnern, Monitoren und Software-Lizenzen ausgestattet sind. Zudem kann bei der Erstellung des Algorithmus auf bereits in Visual Studio vorhandene mathematische Klassen und Funktionen zugegriffen werden. Die bestehenden mathematischen Regeln und Funktionen reichen für dieses Projekt voll und ganz aus.

3. Wirtschaftlichkeit

Da es sich bei dem Algorithmus zur automatischen Vollbelegung von Dächern mit Photovoltaik-Modulen nicht um eine eigenständige Anwendung handelt, kann ein potentieller Gewinn und der wirtschaftliche Nutzen nicht in Zahlen ermittelt werden. Das Projekt ist eine zusätzliche Funktion zur Verbesserung der Handhabung der Anwendung „PV Simulation“. Somit erschließt sich aus dem Projekt selber kein monetärer Nutzen. Allerdings kann über die Optimierung der Handhabung der Dachvollbelegung Kundenbindung erreicht werden.

Indirekt ließe sich die Amortisation der entstehenden Kosten für den zu entwickelnden Algorithmus über die erzielten Verkäufe der Anwendung „PV Simulation“ ermitteln.



Pflichtenheft

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau

Datum: 13.02.2013

1. Zielbestimmungen

1.1 Musskriterien

Der zu entwickelnde Algorithmus zur Vollbelegung von Dächern mit Photovoltaik-Modulen muss die optimale Belegung der angegebenen Dachflächen ermöglichen. Das Ergebnis über die berechneten Module soll an der Programm-Oberfläche ausgegeben werden, damit schnell nachvollzogen werden kann, welcher Startpunkt (oben links, unten rechts, u.a.) die beste Belegungs-Option darstellt.

Beim Ermitteln der optimalen Belegungsart soll das Dach nicht immer neu gezeichnet werden müssen, sondern nur die Module auf dem Dach. Anschließend muss ein neuer Start-Punkt zur Vollbelegung wählbar sein, um eine neue Vollbelegung durchzuführen.

Des Weiteren soll sichergestellt werden, dass keine Module zeichnenbar sind, wenn noch keine Dachfläche definiert ist.

Bei der Wahl der Dachfläche soll immer nur ein Haken zu setzen sein. Alle anderen Haken müssen entfernt werden. Somit ist sicherzustellen, dass nur eine Dachflächen-Form ausgewählt wurde.

Wird die ganze Zeichnung gelöscht, so muss dies auch mit der Zeichenfläche geschehen.

Für die Eingabe verschiedener Abstände müssen bei dem zu entwickelnden Prototypen Eingabefelder vorhanden sein, die bei Nicht-Verwendung mit festgeschriebenen Werten ([Default-Werten](#)) belegt sein müssen.

1.2 Wunschkriterien

Zum derzeitigen Zeitpunkt sind keine Wunschkriterien formuliert worden.

1.3 Abgrenzungskriterien

Der zu entwickelnde Algorithmus ist kein eigenständiges Programm. Er wird nachträglich in die Anwendung „Grafische Erfassung“ (CAD) integriert und steht danach der Anwendung „PV Simulation“ zur Verfügung.

1.4 Abgabefrist

Seitens des Auftraggebers wurde der 12.04.2013 als Abgabetermin festgesetzt. Der Algorithmus wird für die dann ausstehende Auslieferung der Anwendung „PV Simulation“ benötigt.

2. Produkteinsatz

2.1 Anwendungsbereiche

Der Algorithmus findet seine Anwendung in der „Grafischen Erfassung“ (CAD), die wiederum in der Software-Anwendung „PV Simulation“ Verwendung findet.

2.2 Zielgruppen

Zielgruppe sind die Anwender des Programms „PV Simulation“. Dies können beispielsweise Handwerker, Planer oder Ingenieure sein.

3. Produktfunktionen

3.1 Funktionen

/F010/: Dachflächen-Form auswählen

Drei verschiedene Dachflächen-Formen sind vorab wählbar. Die Auswahl ist durch einen Haken zu kennzeichnen.

/F020/: Eckpunkte für Dach zeichnen

Die gewählte Dachflächen-Form wird mit 2 Klicks (Rechteck- und Dreieck-Dachfläche) bzw. mit 3 Klicks (Trapez-Dachfläche) festgelegt. (Die angeklickten Punkte werden nicht farblich markiert.)

/F030/: Dachfläche darstellen

Die Dachkontur der Dachfläche wird rot umrandet dargestellt.

/F040/: Dach-Daten ausgeben

Abmessungen der Dachflächen werden auf der Programm-Oberfläche angezeigt.

/F050/: Eingabe von Modul-Daten und Abständen mit Startpunkt-Auswahl

Höhe und Breite der Photovoltaik-Module, die Abstände zwischen den Modulen in vertikaler und horizontaler Richtung, sowie der Startpunkt für die Vollbelegung sind einzugeben. Alternativ werden [Default-Werte](#) verwendet.

/F060/: Module auf Dachfläche darstellen

Mögliche Module werden nach Berücksichtigung der Dachabstände visualisiert.

/F070/: Module löschen

Module und Skizze des Randabstandes werden entfernt.

/F080/: Alle Eingaben zurücksetzen

Module, Randabstände, Dachkontur und alle Angaben zu den Bemaßungen werden gelöscht und auf die festgelegten [Default-Werte](#) zurückgesetzt. Auch die Auswahl der Dachflächen-Form wird aufgehoben.

3.2 Algorithmen

Algorithmus zur Eckpunkte- und Dachdaten-Berechnung:

Mit dieser Funktion sollen die fehlenden Eckpunkte der Dächer berechnet werden, die zur Visualisierung der Dachflächen benötigt werden. Zur Ausgabe an der Programm-Oberfläche werden hier die Daten der Dachflächen (z.B. Seitenlänge, Höhe, Breite) rechnerisch ermittelt.

Algorithmus zur Vollbelegung mit Modulen:

Diese Funktion benutzt zum Aufziehen eines rechteckigen Rasters über die ganze Dachfläche die Daten, die bei der Ermittlung der Dachdaten berechnet worden sind. Anschließend sollen nur die Module gezeichnet werden, die auch wirklich innerhalb der gezeichneten Dachfläche nach Abzug der Rand-Abstände liegen.

3.3 Use-Case-Diagramm

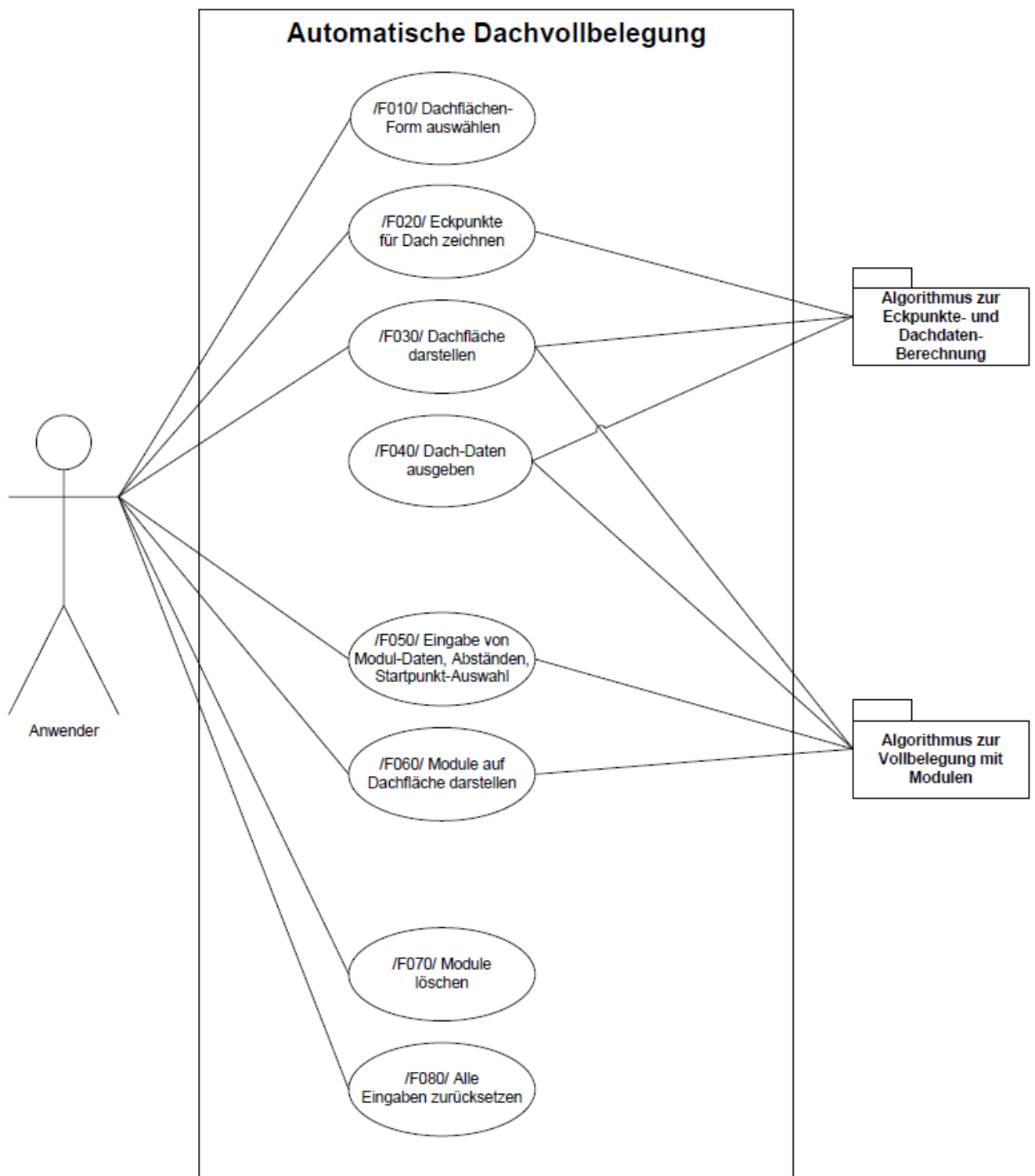


Abb. 9: Use-Case-Diagramm des Algorithmus

4. Produktdaten

4.1 Allgemein

In der Hauptanwendung „Grafische Erfassung“ werden die Daten über die Form des Daches und die daraus resultierenden Dachflächen mittels dreidimensionaler Koordinaten in einer Access-Datenbank (Microsoft Office; im mdb-Format) abgespeichert. Auch die Abmessungen der Photovoltaik-Module, die mit der Anwendung „PV Simulation“ festgelegt wurden, werden in der Datenbank abgespeichert und werden aus dieser wieder abgefragt.

4.2 Daten

Die angeklickten Eckpunkte der Dachkonturen werden in diesem ersten Schritt in lokalen Variablen zwischengespeichert, die nach Löschen aller Eingaben bzw. nach Beenden des Programmes nicht mehr zur Verfügung stehen. Ebenso verhält es sich mit allen einzugebenden Abmessungen, Abständen und den errechneten Maßen. Diese werden in keiner extra erstellten Datenbank gespeichert. Erst beim Einbau in die Hauptanwendung „Grafische Erfassung“ (CAD) muss gewährleistet sein, dass die benötigten Werte aus der Datenbank geholt werden können und die optimale Belegungs-Art mit den verfügbaren Modulen abgespeichert werden können.

Da hier als erster Schritt ein zweidimensionaler Prototyp zur Darstellung des Algorithmus zur Vollbelegung von Dachflächen mit Photovoltaik-Modulen erstellt werden soll, kann an dieser Stelle auf eine Datenbank verzichtet werden.

5. Qualitätsanforderungen

Der Algorithmus soll dem Anwender die Vollbelegung von Dächern erleichtern und eine optimale Flächen-Ausnutzung gewährleisten. Die Korrektheit der Berechnungen hat bei den Qualitätsanforderungen oberste Priorität. Die Wartbarkeit erlangt an Wichtigkeit, da in absehbarer Zeit die möglichen Dachformen erweitert werden sollen. So sollen beispielsweise auch eine diagonale Verlegung von Photovoltaik-Modulen in Kombination mit Aufständigung in der „Grafischen Erfassung“ (CAD) möglich sein. Hier müsste der Algorithmus zur Vollbelegung mathematisch entsprechend angepasst werden.

6. Benutzeroberfläche der Anwendung

Für die Gestaltung des Prototyps für den Algorithmus zur automatischen Vollbelegung von Dächern mit Photovoltaik-Modulen gibt es keine Vorschriften die Oberfläche der Anwendung betreffend.

Beim nächsten Schritt, dem Einbau in die Anwendung „Grafische Erfassung“ (CAD), der nach Abschluss des Projektes erfolgt, richtet sich die GUI nach dem HS-GUI-Style-Guide.

7. Entwicklungsumgebung

7.1 Software

7.1.1 Modul- und Testumgebung

- Betriebssystem: Microsoft Windows Vista inkl. Service Pack 2
- Programmiersprache: Visual C# unter Microsoft Visual Studio 2010

7.1.2 Dokumentation

- Microsoft Office Professional Plus 2010 (Word, Excel, PowerPoint) zur Erstellung aller Dokumente einschließlich des Prozessorientierten Projektberichts. Microsoft Office wurde auch zum Erstellen der PDF-Datei verwendet.
- Microsoft Office Visio 2007 zur Erstellung der UML-Diagramme und Abbildungen.

7.2 Hardware

7.2.1 Modul- und Testumgebung

- Prozessor: Intel® Core™ 2 ; 1,87 GHz
- Arbeitsspeicher: 2 GB
- Grafikkarte: GeCube Radeon X1600 Pro; DirectX9.0; 512 MB Speicher

8. Produktumgebung

8.1 Soft- und Hardware

Der Algorithmus soll in die Anwendung „Grafische Erfassung“ (CAD) integriert werden und für die Anwendung „PV Simulation“ zur Verfügung stehen. Demnach entspricht die Produktumgebung des Algorithmus der Umgebung, die von „PV Simulation“ vorausgesetzt wird.

Für „PV Simulation“ gelten folgende Angaben für Hard- und Software:

- Betriebssystem: Microsoft Windows XP oder höher
- Arbeitsspeicher: 1 GB RAM oder mehr
- Festplattenkapazität: 500 MB
- Grafikkarte: 3D-Grafikkarte mit DirectX9.0c oder höher; 128 MB interner Speicher
- Bildschirmauflösung: 1024 x 768 Pixel oder höher mit mindestens 16-Bit Farbtiefe

Köln, den 14.02.2013

Max Mustermann
(Auftraggeber)

Elvira Musterfrau
(Auftragnehmer)



Arbeitspakete

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau

Datum: 18.02.2013

Dieses Dokument dient dazu, die Inhalte der einzelnen Arbeitspakete aufzulisten, zu beschreiben und ihre Phasenzugehörigkeit deutlich zu machen.

1. Definitions- und Analysephase

1.1 Durchführung einer IST-Analyse

Um einen guten Einstieg in das Projekt zu gewährleisten, wird mit der IST-Analyse begonnen. Ziel dieses Arbeitspaketes ist es, einen konkreten Überblick über den IST-Zustand zu erhalten.

1.2 Entwicklung eines SOLL-Konzeptes

Die Festlegung der Funktionen, über die der Algorithmus schlussendlich verfügen soll, steht in diesem Arbeitspaket im Mittelpunkt. Ziel ist es, ein SOLL-Konzept zu erstellen, in dem die einzelnen Punkte – in diesem Fall Funktionen – schriftlich spezifiziert werden.

1.3 Erstellung einer Durchführbarkeitsanalyse

Machbarkeit, Projektrisiko und Wirtschaftlichkeit werden in diesem Arbeitspaket für das Projekt untersucht und detailliert diskutiert.

1.4 Erstellung eines Pflichtenheftes

Die Erstellung eines Pflichtenheftes zur Festlegung und Definition der Wünsche und Anforderungen des Auftraggebers ist die zentrale Aufgabe dieses Arbeitspaketes. Das Pflichtenheft muss vom Auftraggeber abgenommen werden. Die Abnahme stellt zugleich den Abschluss der Definitions- und Analysephase dar.

2. Planungsphase

2.1 Identifizierung und Definition der Arbeitspakete

Zu Beginn der Planungsphase müssen die einzelnen Arbeitspakete definiert und identifiziert werden. Hier werden die einzelnen Schritte des Projektes aufgelistet und kurz beschrieben.

2.2 Erstellung der Projektplanungsdokumente

Unter anderem gehören der Projektstrukturplan, der Meilensteinplan, der Zeitplan (Gantt-Diagramm), der Ressourcenplan und der Kostenplan zu den Projektplanungsdokumenten. Die Erstellung der eben erwähnten Dokumente ist das Ziel dieses Arbeitspaketes.

2.3 Erstellung eines Qualitätsplans

Die Zusammenfassung von Anforderungen, Vorschriften und Sicherheitsmaßnahmen, die an die Qualität des Projektes gestellt werden, ist das primäre Ziel dieses Arbeitspaketes. Mit der Erstellung des Qualitätsplans wird die Planungsphase abgeschlossen. Damit ist der erste Meilenstein erreicht.

3. Entwurfsphase

3.1 Erstellung eines Testfallkataloges

In diesem Arbeitspaket wird ein umfassender Testfallkatalog erstellt, in dem Vorgehensweise und Ergebnisse der späteren Tests festgelegt werden.

3.2 Erstellung der DV-Entwurfsdokumente

Die Erstellung verschiedener sinnvoller Entwurfsdokumente unter Anwendung verschiedener UML-Diagramme steht im Mittelpunkt dieses Arbeitspaketes. Zu diesen Entwurfsdokumenten zählen unter anderem das Use-Case-Diagramm, das Klassendiagramm, das Sequenzdiagramm oder das Aktivitätsdiagramm, um nur einige der UML-Diagramme zu nennen.

3.3 Entwurf der Programmoberfläche

Im letzten Arbeitspaket der Entwurfsphase wird ein Prototyp zum Visualisieren der Programmoberfläche entworfen, um einen ersten Eindruck zu vermitteln.

4. Implementierungsphase

4.1 Erstellung der Programmoberfläche

Ziel dieses Arbeitspaketes ist die endgültige Umsetzung des zuvor entwickelten Prototyps der Programmoberfläche.

4.2 Programmierung des Algorithmus zur Dachvollbelegung

Die Umsetzung der erstellten Entwurfsdokumente in Quellcode ist das zentrale Ziel dieses Arbeitspaketes. Mit diesem Arbeitspaket schließt auch die Implementierungsphase ab. Es folgt die Testphase. Bevor jedoch mit umfassenden Tests begonnen wird, erfolgt ein [Review](#) mit dem Auftraggeber, da mit der Implementierung bereits der zweite Meilenstein erreicht ist.

5. Testphase

5.1 Whitebox-Test

Anhand des zuvor erstellten Testfallkataloges wird dieses Arbeitspaket abgearbeitet. Ergebnis ist ein Testfallprotokoll, in dem Auffälligkeiten oder auch reibungslose Abläufe vermerkt sind. Nach Abschluss des Tests ist der dritte Meilenstein erreicht.

6. Abschlussphase

6.1 Erstellung der Entwicklerdokumentation

Den ersten Abschnitt der Abschlussphase bildet die Erstellung der Entwicklerdokumentation. Ziel dieses Arbeitspaketes ist ein Funktionsdokument für Entwickler.

6.2 Erstellung des Soll-Ist-Vergleichs

Die Gegenüberstellung von geplanten (Soll-Werte) und tatsächlich (Ist-Werte) benötigten Zeiten ist Kern dieses Arbeitspaketes. Es wird eine Kosten-Nutzen-Analyse durchgeführt, die das Projekt abschließend nochmals reflektiert.

6.3 Erstellung des Abschluss-Qualitätsberichtes

Ob alle Vorschriften, Maßnahmen und Anforderungen, die an die Qualität des erstellten Projektes gestellt worden sind, eingehalten wurden, wird mit der Erstellung des Abschluss-Qualitätsberichtes überprüft.

6.4 Erstellung des Abnahmeprotokolls und Übergabe des Projektes

Finales Ziel dieses Arbeitspaketes ist die Erstellung eines Abnahmeprotokolls. Zudem wird das Projekt inklusive aller erstellten Dokumente an den Auftraggeber übergeben. Mit Erreichung des vierten Meilensteins ist auch die Abschlussphase beendet.

7. Erstellung des Prozessorientierten Projektberichtes

Die Erstellung des „Prozessorientierten Projektberichtes“ befindet sich streng betrachtet außerhalb der eigentlichen Phasen, da er nicht mehr zu dem Projekt gehört. Vielmehr bietet er eine gute Übersicht über den Verlauf des Projektes, beinhaltet eine detaillierte Aufstellung der Tätigkeiten und stellt den sukzessiven Ablauf des Projektes dar.



Zeit- & Meilensteinplan

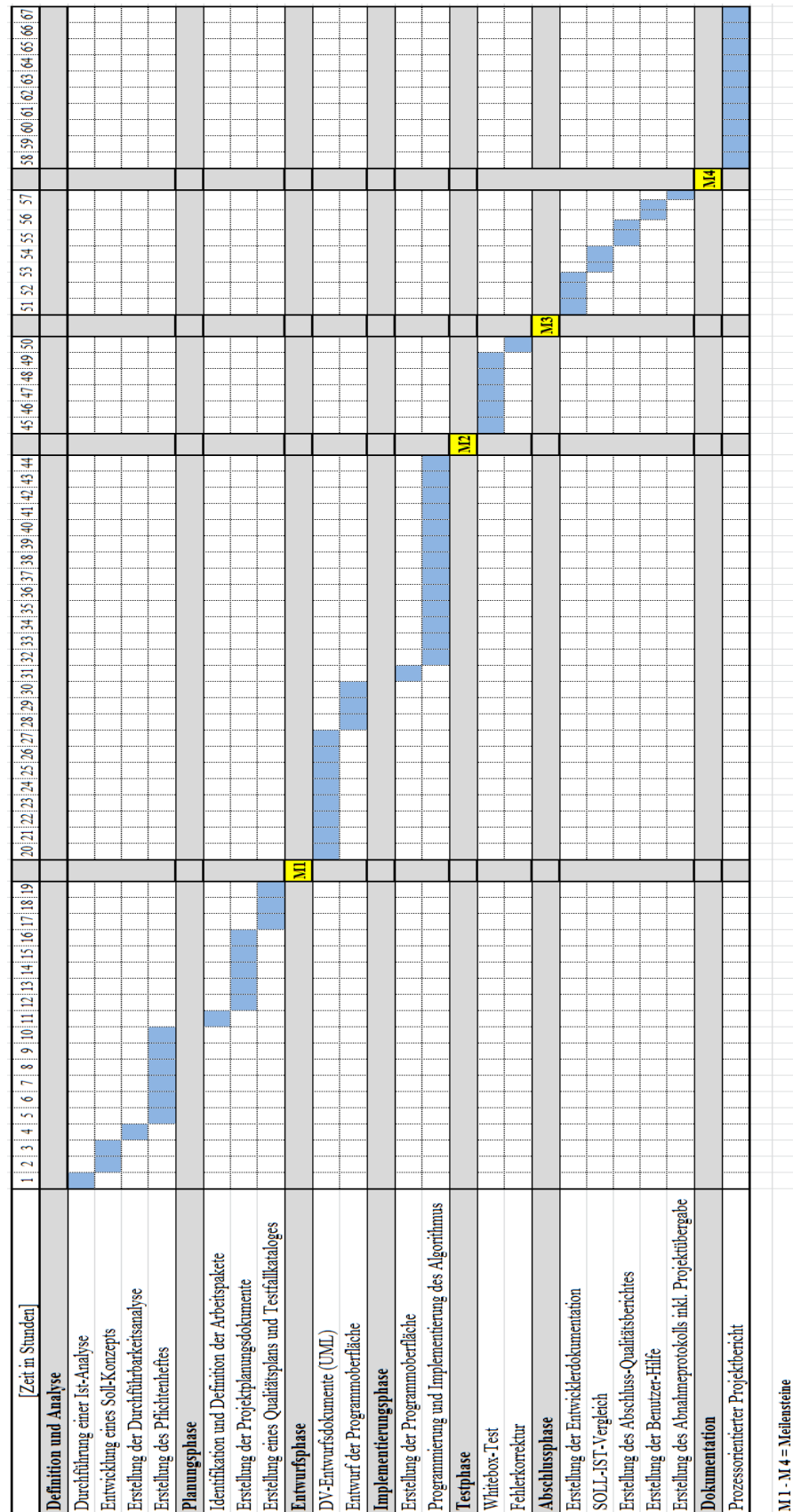
Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
 Photovoltaik-Modulen“**

Autor: Elvira Musterfrau

Datum: 18.02.2013

1.1 Grafische Übersicht



1.2 Allgemeine Zeitplanung

Das Gantt-Diagramm verdeutlicht den zeitlichen Ablauf des Projektes.

Für die Durchführung des gesamten Projektes wurden 67 Stunden geplant. Insgesamt 70 Stunden standen als Zeitrahmen zur Verfügung. Die Differenz von 3 Stunden ist als Puffer vorgesehen worden. Dieser kann genutzt werden, falls einzelne Phasen und Arbeitspakete doch etwas mehr Zeit in Anspruch nehmen, als zuvor geplant wurde.

Die blau dargestellten Zeiten entsprechen den zu Anfang veranschlagten Zeiten der einzelnen Arbeitspakete. Die Meilensteine sind mit **M1** bis **M4** gekennzeichnet.

2. Meilensteinplan

2.1 Meilenstein 1 – Ende der Planungsphase

Der erste Meilenstein wird mit dem Ende der Definitions-, Analyse- und Planungsphase nach voraussichtlich 19 Stunden erreicht. Für diesen Meilenstein müssen folgende Dokumente auf jeden Fall erstellt worden sein:

- [IST-Analyse](#)
- [SOLL-Konzept](#)
- [Durchführbarkeitsanalyse](#)
- [Pflichtenheft](#)
- [Arbeitspakete](#)
- [Zeitplan](#)
- [Meilensteinplan](#)
- [Projektstrukturplan](#)
- [Ressourcenplan](#)
- [Qualitätsplan](#)

2.2 Meilenstein 2 – Ende der Implementierungsphase

Die Entwurfs- und Implementierungsphase wird durch den zweiten Meilenstein beendet. Dieser Meilenstein wird nach 25 Stunden erreicht sein und folgende Dokumente umfassen:

- [Use-Case-Diagramm](#)
- [Aktivitätsdiagramm](#)
- [Klassendiagramm](#)
- [Prototyp der Benutzeroberfläche](#)
- [kommentierter Quellcode](#)
- [Testfallkatalog](#)

2.3 Meilenstein 3 – Ende der Testphase

Das Ende der Testphase wird mit dem dritten Meilenstein erreicht. Dieser Meilenstein sollte inklusive Fehlerkorrektur nach 6 weiteren Stunden erreicht worden sein. Das hieraus resultierende Dokument ist das:

- [Testprotokoll](#)

2.4 Meilenstein 4 – Ende der Abschlussphase

Nach weiteren 7 Stunden wird die Abschlussphase mit dem vierten Meilenstein abgeschlossen. Bis zu diesem Punkt müssen folgende Dokumente erstellt worden sein:

- [Entwicklerdokumentation](#)
- [Qualitätsbericht](#)
- [Benutzer-Hilfe](#)
- [Abnahmeprotokoll](#)
- [SOLL-IST-Vergleich](#)

Insgesamt wird der vierte Meilenstein nach 57 Stunden erreicht sein. Ist dies der Fall, findet das zu entwickelnde Projekt seinen vorläufigen Abschluss. Daran anschließend kann mit dem Erstellen des „Prozessorientierten Projektberichtes“ begonnen werden, der wiederum mit 10 weiteren Stunden veranschlagt wurde. Insgesamt wurden 67 Stunden für die Durchführung der betrieblichen Projektarbeit inklusive Erstellung aller Dokumente angesetzt. Somit bleiben von den vorgeschriebenen 70 Zeitstunden noch 3 Stunden als Puffer zum Auffangen etwaiger zeitlicher Verschiebungen übrig.

Der „Prozessorientierte Projektbericht“ ist das letzte zu erstellende Dokument im Rahmen dieses Projektes. Demnach kennzeichnet der Abschluss des Projektberichtes gleichzeitig das Ende des gesamten Projektes.

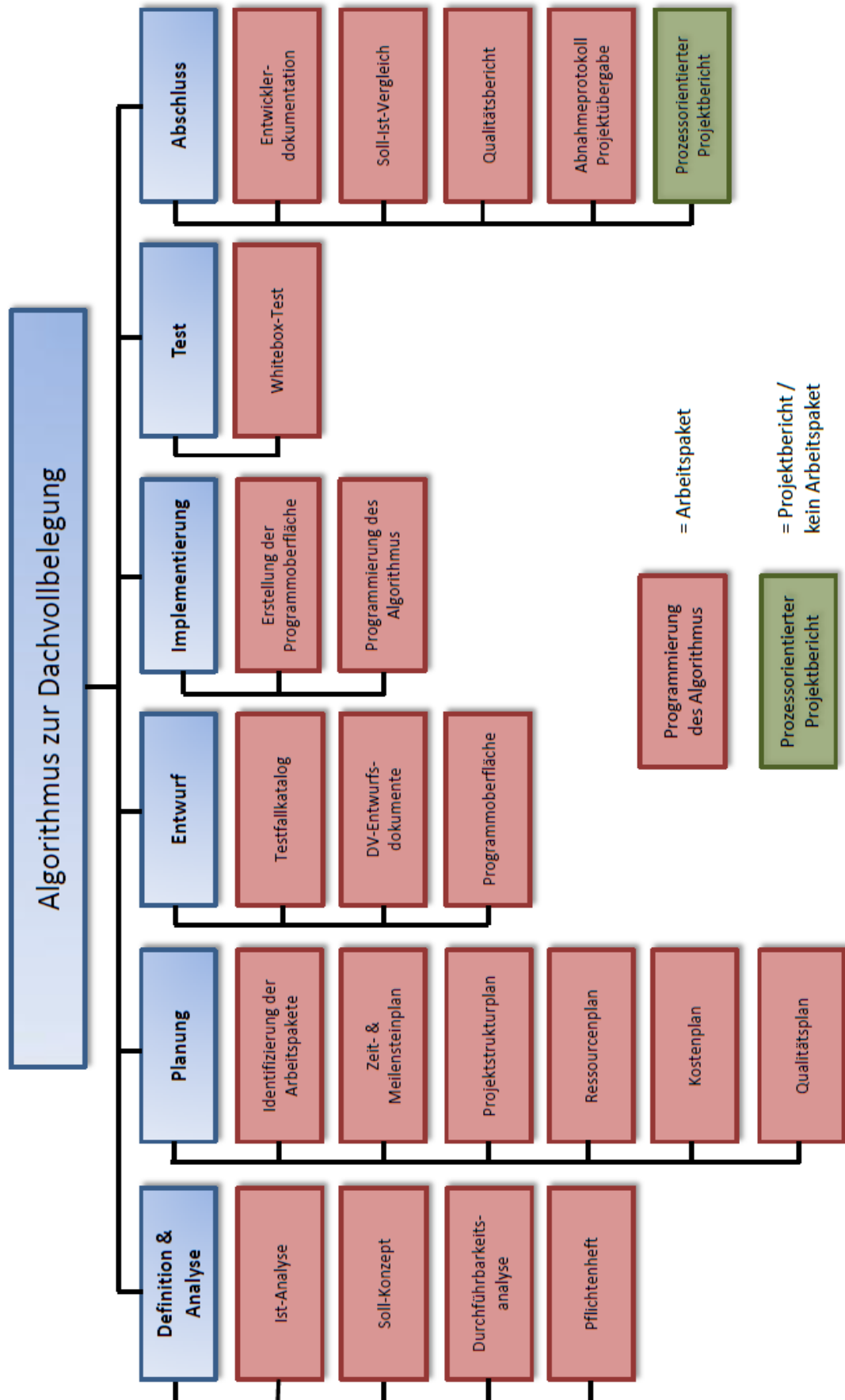


Projektstrukturplan

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 19.02.2013





Ressourcen- & Kostenplan

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 19.02.2013

1. Ressourcenplan

1.1 Personelle Ressourcen

An der Planung und Durchführung des Projektes sind mehrere Personen beteiligt:

- Frau Elvira Musterfrau plant und realisiert das Projekt.
- Herr F. Schulze ist an der Ermittlung des IST-Zustandes beteiligt.
- Herr Max Mustermann ist für die Abnahme des Pflichtenheftes, sowie die abschließende Projektabnahme verantwortlich.
- Herr H. Meier führt den geplanten [Blackbox-Test](#) durch.

1.2 Technische Ressourcen

Für die Entwicklung und den geplanten [Blackbox-Test](#) wird ein Windows-Computer benötigt. Weitere Angaben über die technischen Voraussetzungen und Rahmenbedingungen des Entwicklungs- und Test-Rechners sind im Pflichtenheft unter Punkt 8 „[Produktumgebung](#)“ (siehe Anhang, S. 43 ff.) zu finden. Da ein Arbeitsplatzrechner der Firma Hottgenroth GmbH & Co. KG über die geforderten Hard- und Software-Kapazitäten verfügt, ist hier keine weitere Planung notwendig.

1.3 Räumliche Ressourcen

Zur Umsetzung des Projektes werden die räumlichen Gegebenheiten der Firma Hottgenroth GmbH & Co. KG genutzt. Diese stehen der Antragstellerin zur uneingeschränkten Nutzung zur Verfügung. Somit ist auch hier keine weitere Planung zur Organisation von Räumlichkeiten notwendig.

2. Kostenplan

2.1 Einmalige Projektkosten

Zu den einmaligen Projektkosten gehören primär die Personalkosten, die bei der Durchführung und Umsetzung eines Projektes entstehen. Die entstehenden Kosten für Räume und technische Ausrüstung sind prozentual bereits in den veranschlagten Stundensätzen enthalten. Darüber hinaus müssen keine weiteren Gerätschaften beschafft oder Räume angemietet werden, sodass hier keine weiteren Kosten entstehen.

Die geplante Bearbeitung des Projektes soll sich über 67 Stunden zuzüglich eines dreistündigen Puffers erstrecken. In die entstehende Kostenberechnung fließen jedoch nur die tatsächlich benötigten Stunden ein, was in diesem Fall 67 Stunden sind. Hierzu kommen weitere 4 Stunden, in denen Herr H. Meier den [Blackbox-Test](#) durchgeführt hat. Weiter muss die Zeit von einer halben Stunde für die Besprechung der Programmoberfläche und die Projektabnahme durch Herrn Max Mustermann berücksichtigt werden. Auch die Stunde, die Herr F. Schulze zur Erfassung der Ist-Situation zur Verfügung stellte, muss in die Kostenberechnung einfließen.

Name	Zeitaufwand in Stunden	Stundensatz	Kosten
Elvira Musterfrau	67	89,00 €	5.963,00 €
F. Schulze	1	89,00 €	89,00 €
Max Mustermann	0,5	89,00 €	44,50 €
H. Meier	4	89,00 €	356,00 €
Gesamt			6.452,50 €

Die Firma Hottgenroth GmbH & Co. KG hat eine Stundenpauschale von 89,00 Euro für einen Entwickler festgelegt. In diesem Stundensatz sind sowohl Lohn- und Lohnnebenkosten als auch Raum- und Betriebskosten für den Computer-Arbeitsplatz enthalten. Solche Stundensätze werden auch vollkostenpauschal genannt, da sie alle Nebenkosten prozentual berücksichtigen. Ebenso werden Sonderposten wie Stromverbrauch, Wasser oder Kaffee für die Mitarbeiter auf diesen Stundensatz umgelegt.

2.2 Laufende Projektkosten

Laufende Projektkosten entstehen meist in Form von Versand- oder Support-Kosten, die jedoch erst nach der Erstellung des Produktes entstehen. Für das hier vorliegende Projekt können diese Kosten nicht separat erfasst werden, da es sich lediglich um eine zusätzliche Funktion innerhalb der Anwendung „PV Simulation“ handelt. Somit entstehen laufende Kosten nur für die Endanwendung „PV Simulation“. Da diese Anwendung nicht Gegenstand dieses Projektes ist, wird von einer weiteren Untersuchung bezüglich der laufenden Kosten abgesehen.

3. Wirtschaftlichkeitsanalyse

Bei dem vorliegenden Projekt handelt es sich um eine funktionelle Erweiterung einer bereits bestehenden Anwendung. Daher kann an dieser Stelle auf eine Analyse der Wirtschaftlichkeit bzw. eine Ermittlung der Zeit, ab wie vielen Verkäufen sich das Projekt amortisieren würde, verzichtet werden.

Das Projekt hat einen nicht berechenbaren monetären Nutzen für die Firma Hottgenroth GmbH & Co. KG. Durch die Optimierung der Dachvollbelegung kann der Anwender seine Gebäude rascher mit Modulen belegen. Dies führt zur Erhöhung der Benutzerfreundlichkeit und damit auch Kundenzufriedenheit und verbessert somit die Kundenbindung.



Qualitätsplan

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 20.02.2013

Damit die Qualität umfassend gesichert werden kann, muss zwischen projekt- und produktbezogenen Maßnahmen unterschieden werden.

1. Produktqualität

Ziele	Maßnahmen	Prüfkriterien
Korrektheit	Pflichtenheft	Vergleich von Soll- und Ist-Werten
	Testfallkatalog, Testprotokoll	Übereinstimmung der Ergebnisse, die für den Testfall festgelegt sind
Bedienbarkeit / Verwendbarkeit	Oberflächenprototyping	Kontrolle / Prüfung durch den Auftraggeber und den Qualitätsmanagement-Beauftragten
	Softwaregestaltung	Prüfung und Einhaltung des HS GUI Style-Guide ; Prüfung durch den Qualitätsmanagement-Beauftragten
Erweiterbarkeit / Wartbarkeit	Namenskonventionen	Prüfung der HS Programmierrichtlinien C# ; Prüfung durch eine zweite Person
	Einrückungen zur besseren Lesbarkeit des Quellcodes	Prüfung der HS Programmierrichtlinien C# ; Prüfung durch eine zweite Person
	Kommentieren des Quellcodes	Prüfung der HS Code-Dokumentationsrichtlinien ; Prüfung durch eine zweite Person
	Anlegen einer technischen Dokumentation (DV-Konzept)	Kontrolle, ob die Diagramme umsetzbar sind
Wiederverwendbarkeit	Einhaltung der Paradigmen der objektorientierten Programmierung	Prüfung durch eine zweite Person

2. Projektqualität

Ziele	Maßnahmen	Prüfkriterien
Einhaltung des Budgets	Erstellung eines Kostenplans	Stimmen die Angaben mit dem Soll-Ist-Vergleich überein?
Strukturierter Ablauf	Entscheidung zu einem Vorgehensmodell	Es gibt kein Prüfkriterium
Einhaltung des vorgegebenen Zeitrahmens	Erstellung einer Zeitplanung	Soll-Ist-Vergleich der Zeiten
	Planung der Meilensteine	Stimmen die Angaben mit dem Soll-Ist-Vergleich überein?
	Aufstellung der Ressourcen	Stimmen die Angaben mit dem Soll-Ist-Vergleich überein?



DV-Entwurfsdokumente

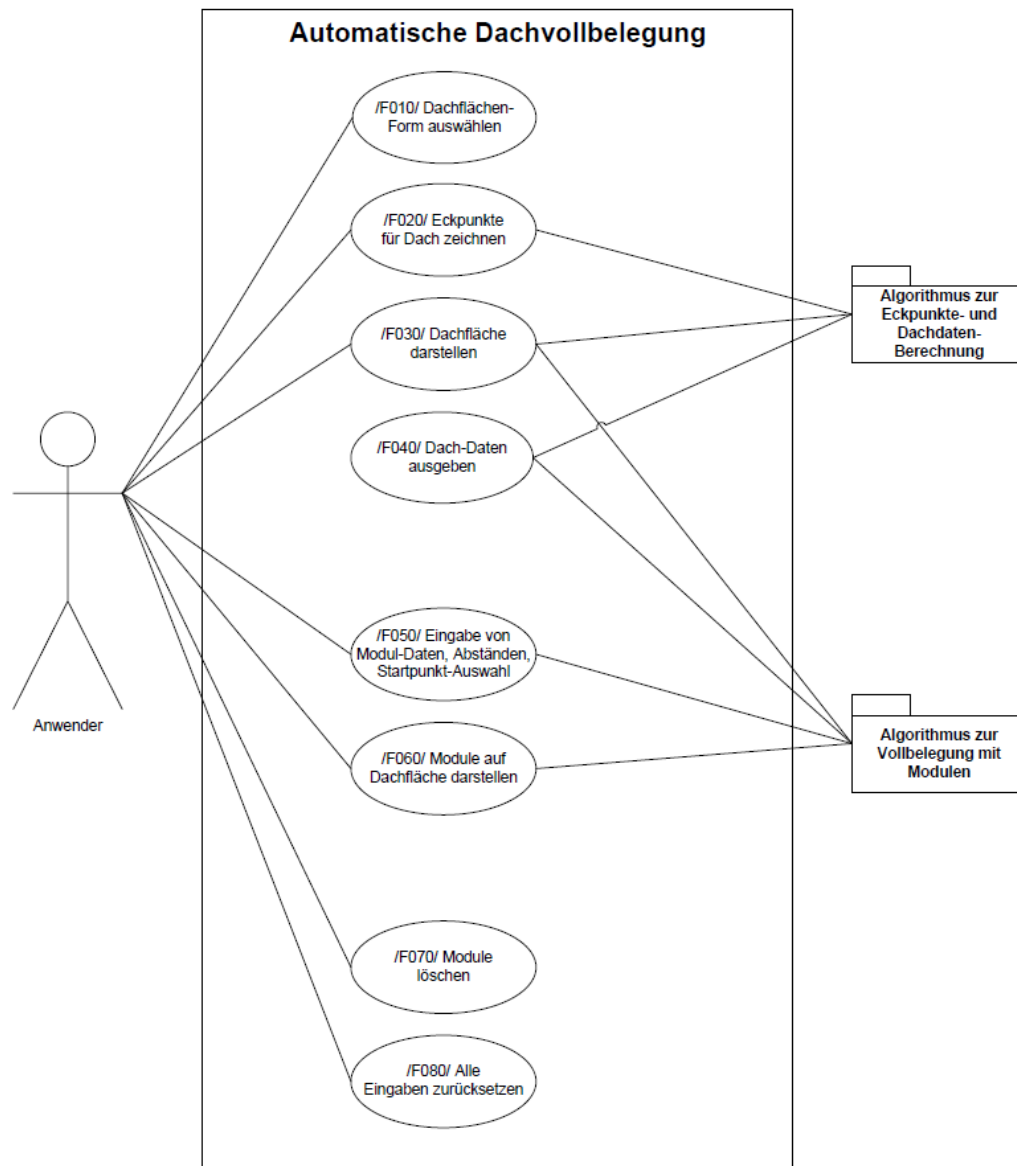
Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 25.02.2013

1. Use-Case

1.1 Use-Case-Diagramm



In dem Use-Case-Diagramm wird die Bedienung des Algorithmus durch den Anwender dargestellt. Die angezeigten Funktionen sind vom Endanwender frei nutzbar.

Bei der Funktion „**Dachflächen-Form wählen**“ stehen dem Anwender 3 Dachflächen-Formen (Rechteck, Trapez, Dreieck) zur Verfügung. Hieraus ist eine Form zu wählen. Anschließend klickt der User für die Funktion „**Eckpunkte für Dach zeichnen**“ zwei (rechteckiges / dreieckiges Dach) bzw. 3 Punkte (trapezförmiges Dach) an. Die Koordinaten der Eckpunkte werden vom „**Algorithmus zur Eckpunkte- und Dachdaten-Berechnung**“ gespeichert. Mit Aktivierung der Schaltfläche „**Dachfläche zeichnen**“ wird die Funktion „**Dachfläche darstellen**“ aktiviert. Hierfür werden die errechneten Werte der fehlenden Eckpunkte verwendet, die zuvor vom „**Algorithmus**

zur **Eckpunkte- und Dachdaten-Berechnung**“ berechnet und bereitgestellt worden sind. Die Koordinaten der dargestellten Eckpunkte sowie die errechneten Dach-Daten werden dem „**Algorithmus zur Vollbelegung mit Modulen**“ zur Verfügung gestellt.

Der Anwender hat verschiedene Eingabemöglichkeiten, um die Modulgröße sowie Abstände zwischen den Modulen und zu den Dachkanten zu erfassen. Vorab sind sinnvolle Werte als [Default-Werte](#) festgelegt, die durch den Anwender verändert werden können. Weiter kann der Anwender in der Funktion „**Eingabe von Modul-Daten, Abständen, Startpunkt-Auswahl**“ einen Eckpunkt der Dachfläche auswählen, von dem aus die Modul-Belegung beginnen soll. Diese Informationen verwendet der „**Algorithmus zur Vollbelegung mit Modulen**“, um die Position der Module zu errechnen. Mit Aktivierung der Schaltfläche „**Module zeichnen**“ wird die Berechnung im Hintergrund gestartet. Das Ergebnis in Form der Modulpositionen wird auf der Benutzeroberfläche dargestellt. Auch die Anzahl der positionierten Module wird angezeigt.

Sollen verschiedene Startpunkte und die daraus resultierende Vollbelegung (Position und Anzahl) miteinander verglichen werden, können über die Schaltfläche „**Module löschen**“ die Module und die Hilfslinie der Dachrand-Abstände gelöscht werden. Mit dem Button „**Module zeichnen**“ und der vorherigen Auswahl eines anderen Startpunktes kann der Anwender eine alternative Vollbelegung anzeigen lassen. Die Funktion „**Alle Eingaben zurücksetzen**“ löscht die ganze Zeichnung und stellt die eingegebenen Werte für Module, Abstände und den Startpunkt auf die voreingestellten Default-Werte zurück.

1.2 Anwendungsfall-Erzählung / User-Story aus Sicht des Anwenders

Die folgende Erzählung ist in Hinblick auf Namen und Orte frei erfunden:

Der Chef des Handwerksbetriebes Werner Schmitz ist auf die Planung und Konstruktion von Photovoltaik-Anlagen spezialisiert und hat sich eine Version der Anwendung „PV Simulation“ mit „Grafischer Erfassung“ (CAD) der Firma Hottgenroth GmbH & Co. KG auf einer Messe gekauft. Damit möchte er auf Anfrage von Familie Schulze ermitteln, wie eine Photovoltaik-Anlage auf dem Dach ihres Einfamilienhauses realisiert werden könnte.

Hierfür legt Werner Schmitz ein neues Projekt an und wird aufgefordert, die Adress- und Standort-Daten des Hauses einzugeben. Herr Schulze, Auftraggeber für dieses Projekt, hat einen Grundriss seines Einfamilienhauses zur Verfügung gestellt. Jetzt kann Werner Schmitz aus „PV Simulation“ heraus die „Grafische Erfassung“ (CAD) starten, um das Haus nachzuzeichnen. Hier kommt er rasch voran, da für seine Zwecke der Grundriss des Hauses reicht, auf den er das Dach des Hauses

mit seiner Form und Neigungswinkeln modellieren kann. Anschließend verlässt Werner Schmitz die „Grafische Erfassung“ (CAD).

Zurück in „PV Simulation“ ruft er den Katalog auf, um das seiner Meinung nach passende Modul seines bevorzugten Photovoltaik-Modul-Herstellers auszusuchen. Die Abmessungen des Moduls (Höhe und Breite) werden den Katalog-Einträgen entnommen. Nach Auswahl der technischen Ausstattung der Module und der für die Auslegung von PV-Modulen benötigten [Inverter](#) geht Werner Schmitz wieder in die „Grafische Erfassung“. Hier wechselt er in den Arbeitsbereich, wo das Dach gezeichnet wurde. In einem dem Dach untergeordneten Bereich findet Werner Schmitz den Bereich, wo er PV-Module auf das Dach legen kann. Zur Wahl standen ihm ursprünglich die Funktionen, Einzelmodule auf das Dach zu legen, oder Felder mit Modulen aufzuziehen. Aus vorherigen Erfahrungen weiß er schon, dass er mit beiden Möglichkeiten nicht das optimale Ergebnis der Vollbelegung erzielen kann. Immer wieder blieben Dachflächen übrig, die nicht genutzt werden konnten.

Doch auf der Messe hat Werner Schmitz eine neue Funktion gezeigt bekommen: eine automatische Vollbelegung! Neugierig, wie das Ganze funktioniert, klickt er auf den Knopf mit der Beschriftung „**Automatische Vollbelegung**“. Nach der Aufforderung die Dachfläche zu wählen, die er mit PV-Modulen belegen möchte, öffnet sich ein zusätzliches Fenster. Hier kann Werner Schmitz verschiedene Angaben machen. Unter anderem kann er die Abstände zwischen den einzelnen Modulen in horizontaler und vertikaler Richtung eintragen. Auch den Abstand, der zu den Dachkanten eingehalten werden soll, kann er festlegen. Diese Auswahlmöglichkeiten empfindet Werner Schmitz als angenehm, da er konkrete Werte eingeben kann. Bisher musste er bei den Abständen zu den Dachkanten schätzen und hoffen, dass die Abstände in etwa stimmen. Weiter sieht Werner Schmitz in dem angezeigten Fenster, dass er einen Startpunkt für die Vollbelegung aussuchen kann. Gespannt schaut er sich die Möglichkeiten an und stellt fest, dass er in jeder beliebigen Ecke der Dachfläche mit der Modul-Belegung beginnen kann. Das ist natürlich eine sehr schöne Angelegenheit, findet Werner Schmitz. Probesthalber wählt er als Startpunkt „oben rechts“ aus und klickt auf „**OK**“.

Wenige Augenblicke später wird auf der Zeichenfläche der „Grafischen Erfassung“ (CAD) die Modul-Belegung angezeigt. Neugierig misst Werner Schmitz unter „2D-Dokumente“ in der „Grafischen Erfassung“ die Abstände zu den Dachrändern und zwischen den Modulen nach. Es stimmt auf den Millimeter genau! Bewegt Werner Schmitz die Maus über das entstandene Modulfeld, erscheint ein kleines Kästchen am Mauszeiger. Hier steht die Anzahl der gelegten Module. Das ist sehr bequem. Doch was ist, wenn er einen anderen Startpunkt aussucht? Kann

Werner Schmitz dann mehr Module zeichnen? Er probiert es einfach aus: mit der Maus zieht er ein farbiges Feld über das gezeichnete Modulfeld. Jetzt erscheinen alle markierten Module mit rosafarbiger Umrandung. Auf der Messe wurde ihm gesagt, dass er das Feld über die Taste „Entf“ (Entfernen) ganz schnell löschen kann, wenn es vorher selektiert ist. Funktioniert einwandfrei, stellt er fest.

Jetzt will Werner Schmitz auch die anderen Startpunkte für die Vollbelegung ausprobieren und schauen, ob sich bei der maximalen Modulanzahl, die positioniert werden kann, etwas ändert. Deshalb klickt er wieder auf den Kopf „**Automatische Vollbelegung**“ und wählt die Dachfläche, auf die er die Module legen möchte. Bei den Abständen, die noch seinen ursprünglich eingegebenen Werten entsprechen, ändert er nichts. Nur dieses Mal wählt Werner Schmitz „unten links“ als Startpunkt für die Vollbelegung und bestätigt mit „**OK**“. Das Ergebnis bekommt er auch jetzt wieder in der „Grafischen Erfassung“ (CAD) angezeigt. Die Anzeige der Modulfelder am Mauszeiger verrät Werner Schmitz, dass bei diesem Startpunkt ein Modul mehr auf das Dach gelegt werden kann. Vorher musste er selber ausprobieren, wie die Module platzsparend angelegt werden können.

Jetzt spart ihm diese neue Funktion nicht nur Zeit, sondern auch Nerven. Und Werner Schmitz sieht schnell, welcher Startpunkt das optimale Ergebnis für die Auslegung mit Photovoltaik-Modulen ist. Er ist zufrieden, seine Kunden sind es auch und er ist mit der neuen Funktion mehr als zufrieden!

1.3 Use-Case-Beschreibungen

Name	/F010/: Dachform auswählen
Kurzbeschreibung	Der Anwender sucht die gewünschte Dachform aus 3 Möglichkeiten aus und markiert sie.
Akteure	Anwender
Auslöser	Setzen eines Hakens bei der gewünschten Dachform.
Vorbedingung	Programmstart
Ablauf	1.) Programm starten 2.) Gewünschte Dachform markieren
Ergebnis	Gewünschte Dachform ist markiert.
Nachbedingung	Benötigte Eckpunkte werden angeklickt

Name	/F020/: Eckpunkte für Dach zeichnen
Kurzbeschreibung	Der Anwender klickt für Höhe und Breite der gewünschten Dachform 2-3 Punkte auf der Zeichenfläche an.
Akteure	1.) Anwender 2.) Algorithmus zur Eckpunkte- und Dachdaten-Berechnung
Auslöser	Anwender hat die gewünschte Dachform markiert
Vorbedingung	Dachform ist ausgewählt
Ablauf	Anwender klickt zur Markierung von Breite und Höhe der gewünschten Dachform 2 – 3 Punkte an
Ergebnis	x- und y-Koordinaten der Eckpunkte werden für weitere Berechnungen gespeichert
Nachbedingung	Es wurde wirklich die benötigte Anzahl an Punkten angeklickt

Name	/F030/: Dachfläche darstellen
Kurzbeschreibung	Per Knopfdruck werden die fehlenden Eckpunkte zum Zeichnen der gewünschten Dachfläche berechnet und an der Benutzeroberfläche visualisiert.
Akteure	1.) Anwender 2.) Algorithmus zur Eckpunkte- und Dachdaten-Berechnung 3.) Algorithmus zur Vollbelegung mit Modulen
Auslöser	Betätigen der Schaltfläche „ Dachfläche zeichnen “
Vorbedingung	Für die gewünschte Dachfläche benötigte Anzahl an Eckpunkten wurde angeklickt und die Koordinaten gespeichert
Ablauf	1.) Betätigen der Schaltfläche „ Dachfläche zeichnen “ 2.) Aus den gespeicherten Koordinaten werden die fehlenden Eckpunkte berechnet 3.) Eckpunkte werden zur Visualisierung der Dachfläche an der Benutzeroberfläche miteinander verbunden 4.) Koordinaten der Eckpunkte werden dem „Algorithmus zur Vollbelegung mit Modulen“ zur Verfügung gestellt
Ergebnis	Kontur der Dachfläche wird dem Anwender angezeigt
Nachbedingung	---

Name	/F040/: Dach-Daten ausgeben
Kurzbeschreibung	Per Knopfdruck werden die Außenmaße der Dachflächen berechnet und an der Benutzeroberfläche angezeigt.
Akteure	1.) Anwender 2.) Algorithmus zur Eckpunkte- und Dachdaten-Berechnung 3.) Algorithmus zur Vollbelegung mit Modulen
Auslöser	Betätigen der Schaltfläche „ Dachfläche zeichnen “
Vorbedingung	Die für die gewünschte Dachfläche benötigte Anzahl an Eckpunkten wurde angeklickt und die Koordinaten gespeichert
Ablauf	1.) Aus den gespeicherten Koordinaten werden Maße (Höhe, Breite, Seitenlängen) der Dachformen berechnet 2.) Maße der Dachform werden dem „Algorithmus zur Vollbelegung mit Modulen“ zur Verfügung gestellt
Ergebnis	Außenmaße der gewählten Dachform werden für den Anwender angezeigt
Nachbedingung	---

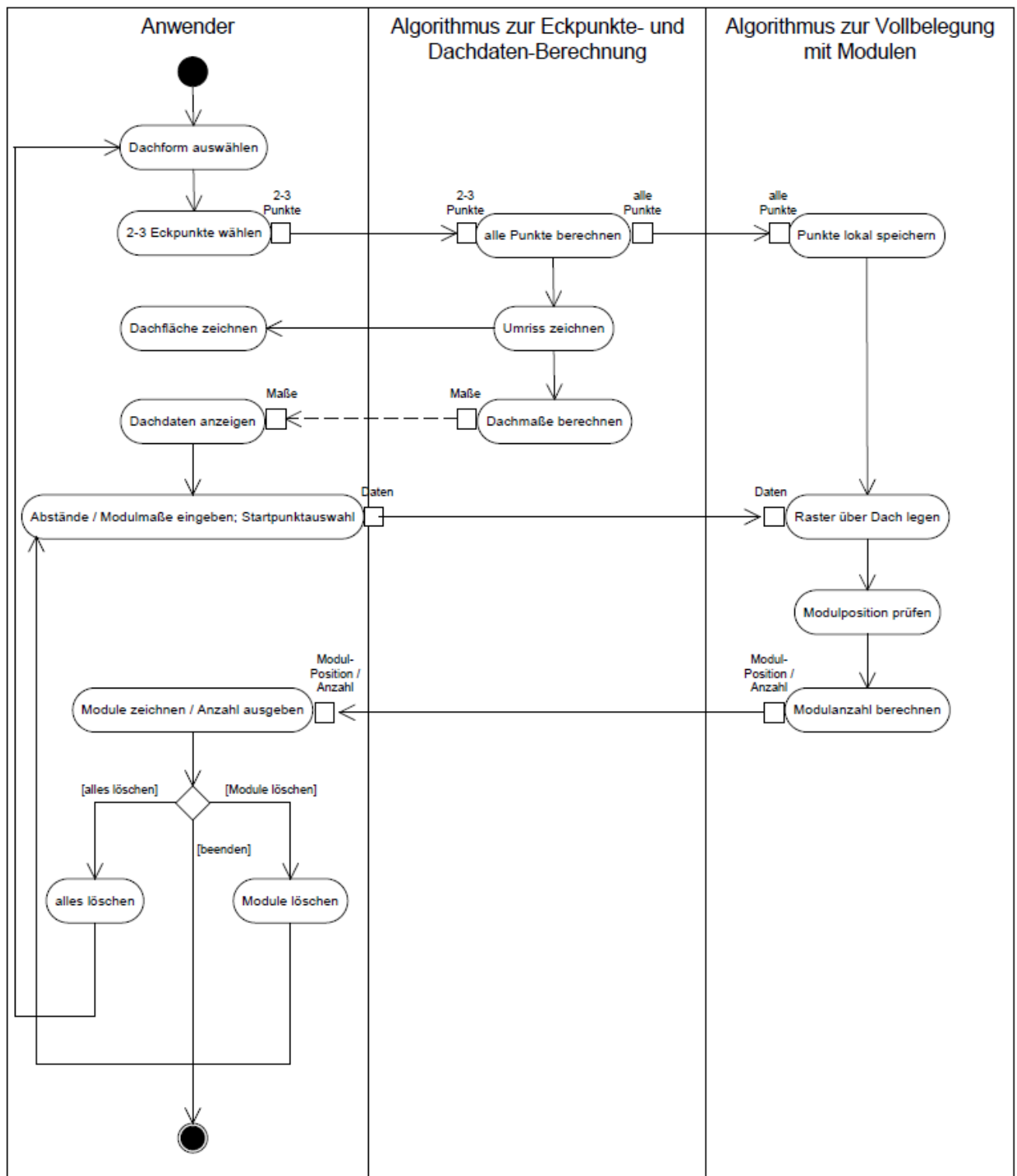
Name	/F050/: Eingabe von Modul-Daten, Abständen, Startpunkt-Auswahl
Kurzbeschreibung	Daten der Module (Breite, Höhe), Abstände zwischen den Module (vertikal, horizontal), Randabstände zu den Dachkanten (vertikal, horizontal) und der Startpunkt, wo die Vollbelegung beginnen soll, können angepasst werden.
Akteure	1.) Anwender 2.) Algorithmus zur Vollbelegung mit Modulen
Auslöser	Dachfläche vorhanden
Vorbedingung	Dachfläche ist gezeichnet worden
Ablauf	1.) Anwender ändert die änderbaren Werte für Modul-Daten, Abstände (zwischen Modulen, zum Dachrand) und den Startpunkt 2.) Werte werden nach Betätigung der Schaltfläche „ Module zeichnen “ dem „Algorithmus zur Vollbelegung mit Modulen“ zur Berechnung übergeben.
Ergebnis	Werte sind der Anwender-Auswahl angepasst; Berechnung kann durchgeführt werden.
Nachbedingung	Auslösen des „Algorithmus zur Vollbelegung mit Modulen“

Name	/F060/: Module auf der Dachfläche darstellen
Kurzbeschreibung	Die Lage und Anzahl der berechneten Module wird grafisch auf der Benutzeroberfläche dargestellt.
Akteure	1.) Anwender 2.) Algorithmus zur Vollbelegung mit Modulen
Auslöser	Betätigen der Schaltfläche „ <i>Module zeichnen</i> “
Vorbedingung	Daten der Dachfläche und der Module, Abstände und Startpunkt vorhanden
Ablauf	1.) Schaltfläche „ <i>Module zeichnen</i> “ betätigen 2.) Lage der Module wird ausgehend vom Startpunkt in Abhängigkeit von Modul-Daten, Abständen und gewählter Dachfläche berechnet. 3.) Grafische Ausgabe des Ergebnisses für den Anwender
Ergebnis	Module werden auf Dachfläche gezeichnet
Nachbedingung	---

Name	/F070/: Module löschen
Kurzbeschreibung	Hilfslinie des Randabstandes zum Dach und Module werden gelöscht. Konturen der Dachfläche bleiben sichtbar. Eingebbare Werte werden nicht verändert.
Akteure	Anwender
Auslöser	Betätigen der Schaltfläche „ <i>Module löschen</i> “
Vorbedingung	Dach, Module und Randabstände sind auf der Benutzeroberfläche dargestellt
Ablauf	1.) Schaltfläche „ <i>Module löschen</i> “ betätigen 2.) Gezeichnete Module und die Hilfslinie für den Randabstand zu den Dachkanten werden gelöscht
Ergebnis	Kontur der Dachfläche bleibt sichtbar; Dach-Daten werden angezeigt; änderbare Daten werden nicht zurückgesetzt
Nachbedingung	---

Name	/F080/: Alle Eingaben zurücksetzen
Kurzbeschreibung	Modul- und Dachzeichnung wird gelöscht. Angezeigte Werte (Moduldaten, Abstände, Startpunkt) werden auf Default-Werte zurückgesetzt. Dach-Daten werden gelöscht.
Akteure	Anwender
Auslöser	Betätigen der Schaltfläche „ <i>Alle Eingaben zurücksetzen</i> “
Vorbedingung	Dach, Module und Randabstände sind auf der Benutzeroberfläche dargestellt
Ablauf	<ol style="list-style-type: none">1.) Schaltfläche „<i>Alle Eingaben zurücksetzen</i>“ betätigen2.) Ganze Zeichnung wird gelöscht3.) Eingebbare Werte auf Default-Werte zurücksetzen4.) Ausgabefelder für Dachdaten leeren
Ergebnis	Alle Eingaben, Werte und Zeichnungen sind gelöscht.
Nachbedingung	Neues Dach kann angelegt werden.

2. Aktivitätsdiagramm



Das Aktivitätsdiagramm stellt das Zusammenspiel zwischen Oberflächen-Eingaben und den internen stattfindenden Berechnungen dar.

Um ein Dach mit Photovoltaik-Modulen belegen zu können, muss der Anwender zuerst eine Dachform definieren (/F010/ im Pflichtenheft) und für diese Dachform Eckpunkte bestimmen

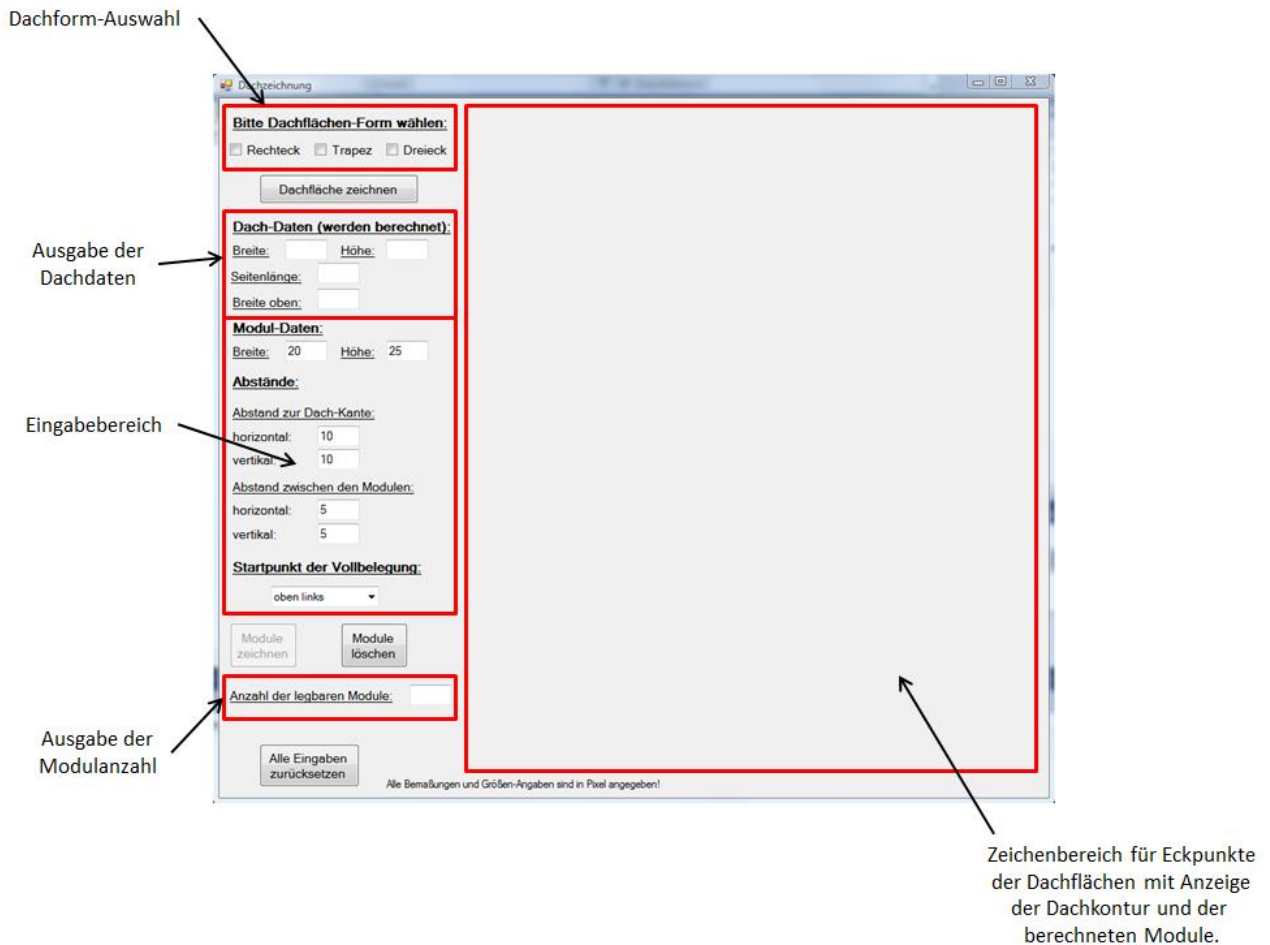
(/F020/ im Pflichtenheft). Die ausgewählten Punkte werden vom „Algorithmus zur Eckpunkte- und Dachdaten-Berechnung“ verwendet, um die fehlenden Eckpunkte und die Maße der Dachfläche zu berechnen. Diese werden miteinander verbunden und an der Benutzeroberfläche als Dachkontur dargestellt (/F030/ im Pflichtenheft). Die Maße der Dachfläche werden für den Anwender an der Oberfläche ausgegeben (/F040/ im Pflichtenheft).

Nach Eingabe der Modul-Daten, Abstände zu den Dachkanten und der Auswahl des Startpunktes (/F050/ im Pflichtenheft) werden diese Daten vom „Algorithmus zur Vollbelegung mit Modulen“ zur Berechnung der Modul-Position und der Gesamtzahl der Module verwendet. Die errechneten Ergebnisse werden anschließend auf der Benutzeroberfläche angezeigt (/F060/ im Pflichtenheft).

Um Vergleiche zwischen den verschiedenen Startpunkten zu ermöglichen, kann der Anwender die gezeichneten Module löschen, ohne die Dachform zu löschen (/F070/ im Pflichtenheft). So kann er feststellen, welcher Startpunkt die optimale Auslegung der Module bietet.

Um eine neue Dachform anzulegen, müssen alle Eingaben gelöscht und auf [Default-Werte](#) zurückgesetzt werden (/F080/ im Pflichtenheft).

3. Prototyp der Programmoberfläche

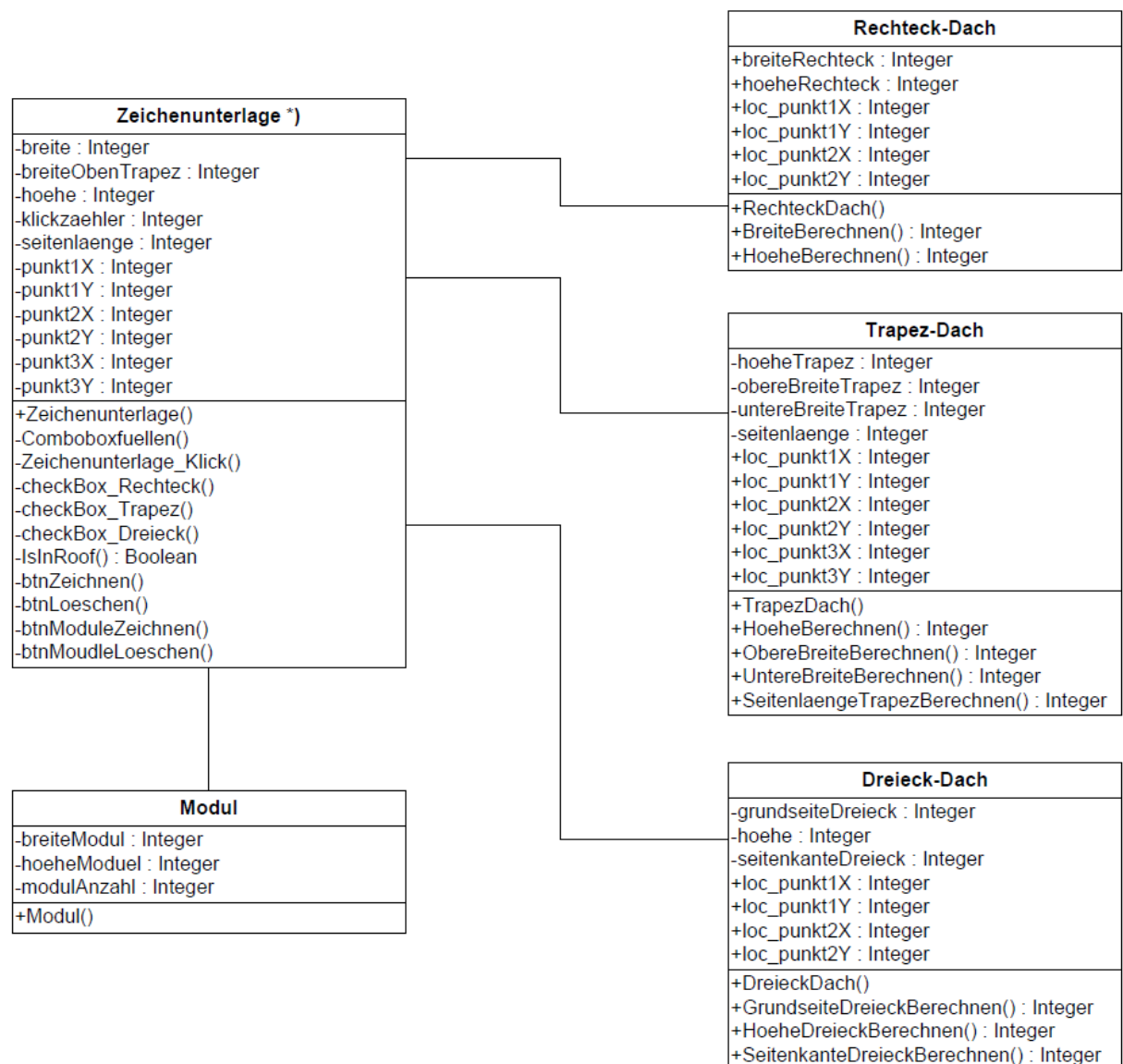


Um den Algorithmus zur automatischen Dachvollbelegung zu visualisieren, sind verschiedene Bereiche für die Benutzeroberfläche erforderlich. Daher wurde der Oberflächen-[Prototyp](#) in einen Zeichenbereich und einen Ein- und Ausgabebereich unterteilt.

Lediglich der Algorithmus mit den Berechnungen wird für die „Grafische Erfassung“ (CAD) übernommen. Daher wird die Benutzeroberfläche in der „Grafischen Erfassung“ (CAD) für die Eingabemöglichkeiten etwas anders aussehen. So werden die Abstände zwischen den Modulen vom Anwender in Zentimeter eingegeben und müssen zur Darstellung in Pixel umgerechnet werden. Zudem muss noch die dritte Dimension für die 3D-Darstellung berücksichtigt und eingerechnet werden. Die Eingabeaufforderung für die Abstände zu den Dachkanten und zwischen den Modulen wird in einem separaten Eingabedialog erscheinen.

Die erstellte Benutzeroberfläche ist lediglich eine [Prototyp](#)-Visualisierung, um die im Hintergrund stattfindenden Berechnungen zu kontrollieren und darzustellen.

4. Klassendiagramm



*) Diese Klasse beschreibt unter anderem die Funktionen, die auf der Benutzeroberfläche zur Verfügung stehen.

Dieses Klassendiagramm zeigt eine Übersicht der Klassen, die für die Umsetzung des Algorithmus zur automatischen Dachvollbelegung benötigt werden. Hier sind die verwendeten Variablen mit den gegebenen Zugriffsrechten dargestellt. Auch die verwendeten Funktionen sind auf einen Blick zu erfassen.

Auf die Funktionen der Dach-Klassen (Rechteck-Dach, Trapez-Dach, Dreieck-Dach) kann ausschließlich öffentlich zugegriffen werden, da Ergebnisse aus eben diesen Funktionen in der Basis-Klasse „Zeichenunterlage“ verwendet werden müssen.



Testfallkatalog & Testfallprotokoll

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 11.03.2013

1. Testfallkatalog für Whitebox- und Blackbox-Test

Der hier aufgestellte Testfallkatalog wurde für einen [Whitebox-Test](#) in der Entwicklungsumgebung von Visual Studio 2010 herangezogen, sowie an Herrn H. Meier gegeben, der diesen Katalog als Grundlage für seinen [Blackbox-Test](#) nahm. Darüber hinaus wurden vom Testfallkatalog unabhängige Szenarien durchgeführt, die hier nicht weiter aufgeführt werden.

ED = Elvira Musterfrau; Name des Testers

Erfasst ist hier das erste Testprotokoll des [Whitebox-Tests](#), bei dem auch der Fehler festgehalten ist.

/F010/: Dachform wählen

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
1	Auswahl Dachform (/F010/ im Pflichtenheft)	Rechteck-Dachfläche auswählen	Nur Rechteck-Dachfläche ist markiert.	Wie erwartet	ED
2	Auswahl Dachform (/F010/ im Pflichtenheft)	Trapez-Dachfläche auswählen	Nur Trapez-Dachfläche ist markiert.	Wie erwartet	ED
3	Auswahl Dachform (/F010/ im Pflichtenheft)	Dreieck-Dachfläche auswählen	Nur Dreieck-Dachfläche ist markiert.	Wie erwartet	ED

/F020/: Eckpunkte für Dach zeichnen

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
4	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Rechteck</u> : "Rechteck" markieren; keinen Punkt anklicken; "Dachfläche zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED
5	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Rechteck</u> : "Rechteck" markieren; einen Punkt anklicken; "Dachfläche zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
6	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Rechteck</u> : "Rechteck" markieren; drei Punkt anklicken; "Dachfläche zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED
7	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Trapez</u> : "Trapez" markieren, keinen Punkt anklicken; "Dachfläche zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED
8	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Trapez</u> : "Trapez" markieren, einen Punkt anklicken; "Dachfläche zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED
9	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Trapez</u> : "Trapez" markieren, zwei Punkte anklicken; "Dachfläche zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED
10	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Trapez</u> : "Trapez" markieren, vier Punkte anklicken; "Dachfläche zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED
11	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Dreieck</u> : "Dreieck" markieren, keinen Punkt anklicken, "Dachfläche zeichnen" wählen	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
12	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Dreieck</u> : "Dreieck" markieren, einen Punkt anklicken, "Dachfläche zeichnen" wählen	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED
13	Eckpunkte für Dachform zeichnen (/F020/ im Pflichtenheft)	<u>Dreieck</u> : "Dreieck" markieren, drei Punkte anklicken, "Dachfläche zeichnen" wählen	Meldung wird ausgegeben mit Hinweis, wie viele Punkte für die Dachform benötigt werden.	Wie erwartet	ED

/F030/: Dachfläche zeichnen

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
14	Dachfläche darstellen (/F030/ im Pflichtenheft)	<u>Rechteck</u> : "Rechteck" markieren, 2 Punkte anklicken, "Dachfläche zeichnen" wählen.	Gewählte Dachfläche wird rot umrandet dargestellt.	Wie erwartet	ED
15	Dachfläche darstellen (/F030/ im Pflichtenheft)	<u>Trapez</u> : "Trapez" markieren, 3 Punkte anklicken, "Dachfläche zeichnen" wählen.	Gewählte Dachfläche wird rot umrandet dargestellt.	Wie erwartet	ED
16	Dachfläche darstellen (/F030/ im Pflichtenheft)	<u>Dreieck</u> : "Dreieck" markieren, 2 Punkte anklicken, "Dachfläche zeichnen" wählen.	Gewählte Dachfläche wird rot umrandet dargestellt.	Wie erwartet	ED

/F040/: Dachdaten ausgeben

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
17	Dach-Daten ausgeben (/F040/ im Pflichtenheft)	Rechteck-Dachfläche anlegen wie unter Test-Punkt 14	Abmessungen werden in den vorgesehenen Feldern angezeigt.	Wie erwartet	ED

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
18	Dach-Daten ausgeben (/F040/ im Pflichtenheft)	Trapez-Dachfläche anlegen wie unter Test-Punkt 15	Abmessungen werden in den vorgesehenen Feldern angezeigt.	Wie erwartet	ED
19	Dach-Daten ausgeben (/F040/ im Pflichtenheft)	Dreieck-Dachfläche anlegen wie unter Test-Punkt 16	Abmessungen werden in den vorgesehenen Feldern angezeigt.	Wie erwartet	ED

/F050/: Modul-Daten ändern

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
20	Modul-Daten ändern (/F050/ im Pflichtenheft)	Beliebige Dachfläche zeichnen; Breite und Höhe der Module im zulässigen positiven Bereich verändern; "Module zeichnen" wählen	Gezeichnete Modul-Größe ändert sich	Wie erwartet	ED
21	Modul-Daten ändern (/F050/ im Pflichtenheft)	Beliebige Dachfläche zeichnen; Breite und Höhe der Module im negativen Bereich verändern; "Module zeichnen" wählen	Meldung wird ausgegeben mit Hinweis, dass nur Werte größer "0" erlaubt sind.	Wie erwartet	ED

/F050/: Abstände zwischen Modulen ändern

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
22	Abstände zwischen Modulen verändern (/F050/ im Pflichtenheft)	Beliebige Dachfläche zeichnen; voreingestellte Abstände zwischen den Modulen (horizontal / vertikal) im erlaubten positiven Bereich verändern; "Module zeichnen" wählen.	Abstände zwischen den Modulen werden verändert	Wie erwartet	ED

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
23	Abstände zwischen Modulen verändern (/F050/ im Pflichtenheft)	Beliebige Dachfläche zeichnen; voreingestellte Abstände zwischen den Modulen (horizontal / vertikal) im negativen Bereich verändern; "Module zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, dass nur Werte größer "0" erlaubt sind.	Wie erwartet	ED

/F050/: Abstände zum Dachrand ändern

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
24	Abstände zum Dachrand verändern (/F050/ im Pflichtenheft)	Beliebige Dachfläche zeichnen; voreingestellte Abstände zu den Dachkanten (horizontal / vertikal) innerhalb des erlaubten positiven Bereichs verändern; "Module zeichnen" wählen.	Abstand zu den Dachkanten verändert sich; Grüne Abstandslinie verschiebt sich.	Wie erwartet	ED
25	Abstände zum Dachrand verändern (/F050/ im Pflichtenheft)	Beliebige Dachfläche zeichnen; voreingestellte Abstände zu den Dachkanten (horizontal / vertikal) innerhalb des negativen Bereichs verändern; "Module zeichnen" wählen.	Meldung wird ausgegeben mit Hinweis, dass nur Werte größer "0" erlaubt sind. Es wird keine Linie gezeichnet.	Wie erwartet	ED

/F050/: Startpunkt-Auswahl ändern

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
26	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Rechteck</u> : Rechteckige Dachfläche zeichnen; Startpunkt "oben links" einstellen; "Module zeichnen wählen".	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
27	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Rechteck:</u> Rechteckige Dachfläche zeichnen; Startpunkt "oben rechts" einstellen; "Module zeichnen wählen".	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED
28	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Rechteck:</u> Rechteckige Dachfläche zeichnen; Startpunkt "unten links" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED
29	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Rechteck:</u> Rechteckige Dachfläche zeichnen; Startpunkt "unten rechts" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED
30	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Trapez:</u> Trapezförmige Dachfläche zeichnen; Startpunkt "oben links" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED
31	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Trapez:</u> Trapezförmige Dachfläche zeichnen; Startpunkt "oben rechts" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED
32	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Trapez:</u> Trapezförmige Dachfläche zeichnen; Startpunkt "unten links" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
33	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Trapez:</u> Trapezförmige Dachfläche zeichnen; Startpunkt "unten rechts" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED
34	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Dreieck:</u> Dreieckige Dachfläche zeichnen; Startpunkt "oben links" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED
35	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Dreieck:</u> Dreieckige Dachfläche zeichnen; Startpunkt "oben rechts" einstellen; "Module zeichnen" wählen.	Kein Unterschied zum Ergebnis von Test-Fall 34 auf Grund der Spitze des Dreieckes.	Wie erwartet	ED
36	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Dreieck:</u> Dreieckige Dachfläche zeichnen; Startpunkt "unten links" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED
37	Startpunkt-Auswahl (/F050/ im Pflichtenheft)	<u>Dreieck:</u> Dreieckige Dachfläche zeichnen; Startpunkt "unten rechts" einstellen; "Module zeichnen" wählen.	Modul-Raster fängt beim gewünschten Startpunkt an; Kontrolle durch Abstandszeichnung zum Dachrand.	Wie erwartet	ED

/F060/: Module auf Dachfläche darstellen

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
38	Module auf Dachfläche darstellen (/F060/ im Pflichtenheft)	<u>Rechteck</u> : Rechteckige Dachfläche zeichnen; "Module zeichnen" wählen.	Module werden mit gewünschtem Dachabstand auf der Dachfläche angezeigt; Modulanzahl wird im entsprechenden Feld angezeigt	Wie erwartet	ED
39	Module auf Dachfläche darstellen (/F060/ im Pflichtenheft)	<u>Trapez</u> : Trapezförmige Dachfläche zeichnen; "Module zeichnen" wählen.	Module werden mit gewünschtem Dachabstand auf der Dachfläche angezeigt; Modulanzahl wird im entsprechenden Feld angezeigt	Wie erwartet	ED
40	Module auf Dachfläche darstellen (/F060/ im Pflichtenheft)	<u>Dreieck</u> : Dreieckige Dachfläche zeichnen; "Module zeichnen" wählen.	Module werden mit gewünschtem Dachabstand auf der Dachfläche angezeigt; Modulanzahl wird im entsprechenden Feld angezeigt	Wie erwartet	ED
41	Module auf Dachfläche darstellen (/F060/ im Pflichtenheft)	Nach Start der Anwendung "Module zeichnen" wählen	Button ist ausgegraut	Wie erwartet	ED

/F070/: Module löschen

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
42	Module löschen (F070/ im Pflichtenheft)	<u>Rechteck</u> : Rechteckige Dachfläche zeichnen; "Module zeichnen" wählen; "Module löschen" aktivieren.	Zeichnung der Dachfläche weiterhin sichtbar. Module und Randabstands-Zeichnung zu den Dachkanten sind gelöscht.	Wie erwartet	ED

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
43	Module löschen (F070/ im Pflichtenheft)	<u>Trapez:</u> Trapezförmige Dachfläche zeichnen; "Module zeichnen" wählen; "Module löschen" aktivieren.	Zeichnung der Dachfläche weiterhin sichtbar. Module und Randabstands-Zeichnung zu den Dachkanten sind gelöscht.	Wie erwartet	ED
44	Module löschen (F070/ im Pflichtenheft)	<u>Dreieck:</u> Dreieckige Dachfläche zeichnen; "Module zeichnen" wählen; "Module löschen" aktivieren.	Zeichnung der Dachfläche weiterhin sichtbar. Module und Randabstands-Zeichnung zu den Dachkanten sind gelöscht.	Wie erwartet	ED

/F080/: Alle Eingaben zurücksetzen

Nr.	Testpunkt	Vorgehensweise	erwartetes Ergebnis	tatsächliches Ergebnis	getestet von:
45	Alle Eingaben zurücksetzen (/F080/ im Pflichtenheft)	<u>Rechteck:</u> Rechteckige Dachfläche zeichnen; "Module zeichnen" wählen; "Alle Eingaben zurücksetzen" aktivieren.	Dachfläche und Module werden gelöscht; eingbbare Werte werden mit Default-Werten gefüllt.	Modul-Breite / -Höhe nicht zurück-gesetzt	ED
46	Alle Eingaben zurücksetzen (/F080/ im Pflichtenheft)	<u>Trapez:</u> Trapezförmige Dachfläche zeichnen; "Module zeichnen" wählen; "Alle Eingaben zurücksetzen" aktivieren.	Dachfläche und Module werden gelöscht; eingbbare Werte werden mit Default-Werten gefüllt.	Modul-Breite / -Höhe nicht zurück-gesetzt	ED
47	Alle Eingaben zurücksetzen (/F080/ im Pflichtenheft)	<u>Dreieck:</u> Dreieckige Dachfläche zeichnen; "Module zeichnen" wählen; "Alle Eingaben zurücksetzen" aktivieren.	Dachfläche und Module werden gelöscht; eingbbare Werte werden mit Default-Werten gefüllt.	Modul-Breite / -Höhe nicht zurück-gesetzt	ED



Kommentierter Quellcode

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 06.03.2013


```
// * Dachvollbelegung: Version 1.0
// *
// * Fertig gestellt am: 08.03.2013
// *
// * Programmiert von: Elvira Musterfrau
// *
// * Vorgesehen für: PV-Simulation mit der CAD-Komponente
// *
// * geändert am: ---
// *
// * Beschreibung:
// *
// * Prototyp zur automatischen Dachvollbelegung von Rechteck-, Trapez- und Dreiecks-Dächern
// *
// * Hierfür wird ein Raster aus Modulen über die gesamte Dachfläche gezogen.
// * Anschließend fragt eine Funktion ab, ob das Modul auf der erlaubten Dachfläche liegt, oder
// * nicht.
// * Liegt das Modul rechnerisch erlaubt, wird es erst gezeichnet.
// * Ein Modulzähler zählt die gelegten Module und gibt die Gesamtzahl zurück.
// *
// * einzugeben sind: Modulbreite und -höhe,
// *                  die Abstände zwischen den Modulen,
// *                  die Abstände zu den Dachrändern,
// *                  Startpunkt für die Dachbelegung (ComboBox-Auswahl)
// * berechnet werden: Seitenwinkel (bei Trapez und Dreieck; wird nur intern verwendet und nicht
// * angezeigt),
// *                  Anzahl der maximal legbaren Module (horizontal und vertikal),
// *                  Innenpunkte der Dachflächen (Randabstand wird hier abgerechnet)
// *                  Rasterstartpunkte, von denen aus ein Raster aus Modulen über die Dachfläche
// *                  aufgezogen wird
// *
// * Belegung der Dachflächen erfolgt mittels einer geschachtelten Zählschleife
// *
// * Zum Zeichnen der Dachflächen werden die Koordinaten der angeklickten Punkte gespeichert
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;

namespace Prototyp_Dachvollbelegung_PV_Simulation
{
    public partial class Zeichenunterlage : Form
    {
```

```
///<summary>
///globale Variablen, die in der ganzen Klasse bekannt sein sollen
///Werte der angeklickten Punkte werden vorab auf "0" gesetzt
///</summary>
#region Membervariablen
//angeklickten Punkt 1 merken
int punkt1X = 0;
int punkt1Y = 0;

//angeklickten Punkt 2 merken
int punkt2X = 0;
int punkt2Y = 0;

//angeklickten Punkt 3 merken
int punkt3X = 0;
int punkt3Y = 0;

//Anzahl der Klicks hochzählen
int klickzaehler = 0;

//erzeugen der geometrischen Objekte (Rechteck-, Dreiecks-, Trapez-Dach und PV-Modul)
private CRechteckDach ObjectRechteck;
private CTrapezDach ObjectTrapez;
private CDreieckDach ObjectDreieck;
private CModul ObjectModul;
public Graphics Grafik;

//Bemaßungen der Zeichnungen vorab mit "0" festlegen; werden später überschrieben
int breite = 0;
int breiteObenTrapez = 0; //nur bei Trapez
int hoehe = 0;
//Seitenfläche (wird bei Dreieck und Trapez benötigt)
int seitenlaenge = 0;
#endregion

public Zeichnenunterlage()
{
    InitializeComponent();
    //Zuweisung der geometrischen Objekte auf ihre Klassen
    ObjectRechteck = new CRechteckDach();
    ObjectTrapez = new CTrapezDach();
    ObjectDreieck = new CDreieckDach();
    ObjectModul = new CModul();
    Grafik = this.CreateGraphics();

    //Button zum Zeichnen der Module ausgrauen; soll erst aktiv sein, wenn Dach gezeichnet ist.
    btnModulZeichnen.Enabled = false;
}
```

```
//Default-Werte für Modul- und Randabstände; in "String" umgewandelt zur Anzeige in TextBox
textBox_Modulabstand_X.Text = Convert.ToString(5);
textBox_Modulabstand_Y.Text = Convert.ToString(5);
textBox_Rand_X.Text = Convert.ToString(10);
textBox_Rand_Y.Text = Convert.ToString(10);
textBox_ModulBreite.Text = Convert.ToString(20);
textBox_ModulHoehe.Text = Convert.ToString(25);

Combobox_fuellen();
}

//Combobox zur Auswahl des Startpunktes füllen
private void Combobox_fuellen()
{
    //Combobox anlegen und mit möglichen Startpunkten füllen
    comboBox_Startpunkt.Items.AddRange(new string[] { "oben links", "oben rechts", "unten
links", "unten rechts" });

    //Standard-Einstellung der ComboBox festlegen
    comboBox_Startpunkt.SelectedItem = "oben links";
}

///<summary>
///merken der angeklickten Punkte, um hinterher damit zu zeichnen und zu rechnen
///Punkte werden auf der Zeichenunterlage angeklickt.
///x- und y-Koordinaten werden gespeichert und an lokale Variablen der Dachtypen-Klassen
übergeben
///</summary>
#region angeklickte Punkte merken
private void Zeichenunterlage_MouseClick(object sender, MouseEventArgs e)
{
    //Liste zum Speichern der angeklickten Punkte für das Dach
    ArrayList angeklicktePunkte = new ArrayList();

    //ersten Klickpunkt merken
    if (klickzaehler == 0)
    {
        punkt1X = e.X;

        ObjectRechteck.loc_punkt1X = angeklicktePunkte.Add(punkt1X);
        ObjectTrapez.loc_punkt1X = angeklicktePunkte.Add(punkt1X);
        ObjectDreieck.loc_punkt1X = angeklicktePunkte.Add(punkt1X);
        ObjectModul.loc_punkt1X = angeklicktePunkte.Add(punkt1X);

        punkt1Y = e.Y;

        ObjectRechteck.loc_punkt1Y = angeklicktePunkte.Add(punkt1Y);
        ObjectTrapez.loc_punkt1Y = angeklicktePunkte.Add(punkt1Y);
```

```
        ObjectDreieck.loc_punkt1Y = angeklicktePunkte.Add(punkt1Y);
        ObjectModul.loc_punkt1Y = angeklicktePunkte.Add(punkt1Y);
    }

    //zweiten Klickpunkt merken
    if (klickzaehler == 1)
    {
        punkt2X = e.X;

        ObjectRechteck.loc_punkt2X = angeklicktePunkte.Add(punkt2X);
        ObjectTrapez.loc_punkt2X = angeklicktePunkte.Add(punkt2X);
        ObjectDreieck.loc_punkt2X = angeklicktePunkte.Add(punkt2X);
        ObjectModul.loc_punkt2X = angeklicktePunkte.Add(punkt2X);

        punkt2Y = e.Y;

        ObjectRechteck.loc_punkt2Y = angeklicktePunkte.Add(punkt2Y);
        ObjectTrapez.loc_punkt2Y = angeklicktePunkte.Add(punkt2Y);
        ObjectDreieck.loc_punkt2Y = angeklicktePunkte.Add(punkt2Y);
        ObjectModul.loc_punkt2Y = angeklicktePunkte.Add(punkt2Y);
    }

    //dritten Klickpunkt merken
    if (klickzaehler == 2)
    {
        punkt3X = e.X;
        ObjectTrapez.loc_punkt3X = angeklicktePunkte.Add(punkt3X);

        punkt3Y = e.Y;
        ObjectTrapez.loc_punkt3Y = angeklicktePunkte.Add(punkt3Y);
    }

    klickzaehler++;
}
#endregion

/// <summary>
/// Zeichnung der Dach-Außenkonturen mit Koordinaten der angeklickten Punkte;
/// Sicherstellung, dass genügend Punkte zum Zeichnen der Dachflächen angeklickt wurden
/// </summary>
private void btnZeichnen_Click(object sender, EventArgs e)
{
    #region Rechteck zeichnen und berechnen
    //für Rechteck-Fläche müssen 2 Punkte angeklickt werden; daraus werden Höhe und Breite
    errechnet
    //   P1_____
    //   |         |
    //   |         |
```

```
//      |_____P2
if (checkBox_Rechteck.Checked == true)
{
    //Sicherstellen, dass auch Punkte angeklickt sind
    if (klickzaehler < 2 || klickzaehler > 2)
    {
        MessageBox.Show("Bitte klicken Sie 2 Punkte für ein Rechteck an!");
        //Zeichenfläche wieder leeren, um von vorne anzufangen
        Grafik.Clear(System.Drawing.SystemColors.Control);

        //Modul-Button inaktiv, damit nicht weiter gearbeitet werden kann
        btnModulZeichnen.Enabled = false;

        //Klickzähler wieder auf Null setzen
        klickzaehler = 0;
    }
    else
    {
        //Modul-Button aktiv, damit weiter gearbeitet werden kann
        btnModulZeichnen.Enabled = true;

        //Berechnung von Breite und Höhe des Rechtecks zum Zeichnen und Ausgeben der
Werte
        hoehe = (punkt2Y - punkt1Y);
        breite = (punkt2X - punkt1X);

        //Rechteck auf Zeichenunterlage zeichnen (Höhe und Breite, sowie fehlende Punkte
berechnen)
        Grafik.DrawRectangle(Pens.Red, punkt1X, punkt1Y, breite, hoehe);

        //Breite und Höhe in Pixel in Textbox ausgeben
        textBox_Breite.Text = Convert.ToString(breite);
        textBox_Hoehe.Text = Convert.ToString(hoehe);
        textBox_Seite.Text = "-----";
        textBox_BreiteObenTrapez.Text = "-----";
    }
}
#endregion

#region Trapez zeichnen und berechnen
//3 Punkte für ein Trapez anklicken; Werte (Höhe, Breiten, usw.) werden berechnet
//      P1_____P2
//      /           \
//      /             \
//      /_____P3
else if (checkBox_Trapez.Checked == true)
{
    //Sicherstellen, dass auch Punkte angeklickt sind
    if (klickzaehler < 3 || klickzaehler > 3)
```

```
{
    MessageBox.Show("Bitte klicken Sie 3 Punkte an, um ein Trapez anzulegen!");
    //Zeichenfläche wieder leeren, um von vorne anzufangen
    Grafik.Clear(System.Drawing.SystemColors.Control);

    //Modul-Button inaktiv, damit nicht weiter gearbeitet werden kann
    btnModulZeichnen.Enabled = false;

    //Klickzähler zurück auf Null setzen
    klickzaehler = 0;
}
else
{
    //Modul-Button aktiv, damit weiter gearbeitet werden kann
    btnModulZeichnen.Enabled = true;

    //Trapez mit Hilfe von 3 angeklickten Punkten zeichnen; 4. Punkt errechnen
    //Linie von Punkt 1 nach Punkt 2
    Grafik.DrawLine(Pens.Red, punkt1X, punkt1Y, punkt2X, punkt1Y);
    //Linie von Punkt 2 nach Punkt 3
    Grafik.DrawLine(Pens.Red, punkt2X, punkt1Y, punkt3X, punkt3Y);
    //Linie von Punkt 3 nach Punkt 4
    Grafik.DrawLine(Pens.Red, punkt3X, punkt3Y, (punkt1X - (punkt3X - punkt2X)),
punkt3Y);
    //Linie von Punkt 4 nach Punkt 1
    Grafik.DrawLine(Pens.Red, (punkt1X - (punkt3X - punkt2X)), punkt3Y, punkt1X,
punkt1Y);

    //Berechnung der Bemaßungen
    breite = ObjectTrapez.UntereBreiteBerechnen();

    breiteObenTrapez = ObjectTrapez.ObereBreiteBerechnen();

    hoehe = ObjectTrapez.HoeheBerechnen();

    //Ruft Funktion der Klasse "CTrapezDach" auf, mit der die Seitenlänge berechnet wird
    seitenlaenge = ObjectTrapez.SeitenlaengeTrapezBerechnen();

    //Maße in Textboxen ausgeben (in Pixel angegeben!)
    textBox_Breite.Text = Convert.ToString(breite);
    textBox_BreiteObenTrapez.Text = Convert.ToString(breiteObenTrapez);
    textBox_Hoehe.Text = Convert.ToString(hoehe);
    textBox_Seite.Text = Convert.ToString(seitenlaenge);
}
}
#endregion

#region Dreieck zeichnen und berechnen
```

```
// 2 Punkte für ein Dreieck anklicken; Werte (Höhe, Grundseite, Seitenlänge) werden
berechnet
//      P1
//      /\
//      /\
//      /__P2
else if (checkBox_Dreieck.Checked == true)
{
    //Sicherstellen, dass auch so viele Punkte angeklickt sind, wie benötigt werden
    if (klickzaehler < 2 || klickzaehler > 2)
    {
        MessageBox.Show("Bitte klicken Sie 2 Punkte an, um ein Dreieck anzulegen!");
        //Zeichenfläche wieder leeren, um von vorne anzufangen
        Grafik.Clear(System.Drawing.SystemColors.Control);

        //Modul-Button inaktiv, damit nicht weiter gearbeitet werden kann
        btnModulZeichnen.Enabled = false;

        //Klickzähler auf Null setzen
        klickzaehler = 0;
    }
    else
    {
        //Modul-Button aktiv, damit weiter gearbeitet werden kann
        btnModulZeichnen.Enabled = true;

        //Zeichnet gleichschenkliges Dreieck nach Angabe von 2 Punkten durch klicken
        Grafik.DrawLine(Pens.Red, punkt1X, punkt1Y, punkt2X, punkt2Y);
        Grafik.DrawLine(Pens.Red, punkt2X, punkt2Y, (punkt1X - (punkt2X - punkt1X)),
punkt2Y);
        Grafik.DrawLine(Pens.Red, (punkt1X - (punkt2X - punkt1X)), punkt2Y, punkt1X,
punkt1Y);

        //Berechnung der Bemaßungen
        breite = ObjectDreieck.GrundseiteDreieckBerechnen();
        hoehe = ObjectDreieck.HoeheDreieckBerechnen();
        seitenlaenge = ObjectDreieck.SeitenkanteDreieckBerechnen();

        //Maße in Textboxen ausgeben (in Pixel angegeben!)
        textBox_Breite.Text = Convert.ToString(breite);
        textBox_Hoehe.Text = Convert.ToString(hoehe);
        textBox_Seite.Text = Convert.ToString(seitenlaenge);
        textBox_BreiteObenTrapez.Text = "-----";
    }
}
#endregion
}
```

```
//Nur eine Check-Box ist markiert und angeklickt
#region Absicherung, dass nur eine Checkbox markiert ist
private void checkBox_Rechteck_Click(object sender, EventArgs e)
{
    checkBox_Dreieck.Checked = false;
    checkBox_Trapez.Checked = false;
}

private void checkBox_Trapez_Click(object sender, EventArgs e)
{
    checkBox_Rechteck.Checked = false;
    checkBox_Dreieck.Checked = false;
}

private void checkBox_Dreieck_Click(object sender, EventArgs e)
{
    checkBox_Rechteck.Checked = false;
    checkBox_Trapez.Checked = false;
}
#endregion

//gezeichnete Dachfläche löschen und Eingabewerte wieder auf Anfang setzen
private void btnLoeschen_Click(object sender, EventArgs e)
{
    Graphics Grafik;
    //Zeichenobjekt erzeugen
    Grafik = this.CreateGraphics();

    //Bild löschen
    Grafik.Clear(System.Drawing.SystemColors.Control);

    btnModulZeichnen.Enabled = false;
    btnZeichnen.Enabled = true;

    //Checkboxes leeren
    checkBox_Rechteck.Checked = false;
    checkBox_Trapez.Checked = false;
    checkBox_Dreieck.Checked = false;

    //alle Textboxen leeren
    textBox_Breite.Text = "";
    textBox_Hoehe.Text = "";
    textBox_Seite.Text = "";
    textBox_BreiteObenTrapez.Text = "";
    textBox_Gesamtanzahl_Module.Text = "";
    //Default-Werte wieder einsetzen
    textBox_Modulabstand_X.Text = Convert.ToString(5);
    textBox_Modulabstand_Y.Text = Convert.ToString(5);
}
```



```
textBox_Rand_X.Text = Convert.ToString(10);
textBox_Rand_Y.Text = Convert.ToString(10);
textBox_ModulBreite.Text = Convert.ToString(20);
textBox_ModulHoehe.Text = Convert.ToString(25);

comboBox_Startpunkt.SelectedItem = "oben links";

//Klickzähler auf Null setzen
klickzaehler = 0;
}

//Innenpunkte der Dachfläche berechnen
//Abfrage, ob Modul in der gezeichneten Dachfläche liegt
//anschließend zeichnen der Module und zählen
private void btnModulZeichnen_Click(object sender, EventArgs e)
{
    #region verwendete Variablen mit Erklärungen
    //Zähler der Module bei Trapezen und Dreiecken
    int modulCounter = 0;

    //variable Modul-Breite und -Höhe
    int modulBreite = Convert.ToInt16(textBox_ModulBreite.Text);
    int modulHoehe = Convert.ToInt16(textBox_ModulHoehe.Text);

    //Abstände von den Dachrändern
    int Rand_Abstand_von_X = Convert.ToInt16(textBox_Rand_X.Text);
    int Rand_Abstand_von_Y = Convert.ToInt16(textBox_Rand_Y.Text);

    //Abstände zwischen den Modulen
    int modulAbstand_waagrecht_X = Convert.ToInt16(textBox_Modulabstand_X.Text);
    int modulAbstand_senkrecht_Y = Convert.ToInt16(textBox_Modulabstand_Y.Text);

    //eine Moduleinheit ist die Breite eines Moduls zuzüglich eines Abstandes zwischen 2
    Modulen
    int modulEinheit = (modulBreite + modulAbstand_waagrecht_X);

    //Variablen des Startpunktes (Ausgangspunkt des Raster, das über die Dachfläche gelegt
    wird)
    int startPunkt_X = 0;
    int startPunkt_Y = 0;

    //Variablen für maximal legbare Module
    int maxAnzModule_X = 0;
    int maxAnzModule_Y = 0;

    //Variablen für die inneren Punkte
    int iP_0_X = 0;
    int iP_0_Y = 0;
```

```
int iP_1_X = 0;
int iP_1_Y = 0;

int iP_2_X = 0;
int iP_2_Y = 0;

int iP_3_X = 0;
int iP_3_Y = 0;
#endregion

//Null- und negative Werte abfangen
if (Rand_Abstand_von_X < 0 || Rand_Abstand_von_Y < 0 ||
    modulAbstand_waagrecht_X < 0 || modulAbstand_senkrecht_Y < 0 ||
    modulBreite <= 0 || modulHoehe <= 0)
{
    MessageBox.Show("Bitte geben Sie einen Wert größer 0 ein!");
}
else
{
    #region Schleife zur Vollbelegung des Rechteck-Daches
    //Zeichnung der Module mit Hilfe einer Schleife mit verschiedenen Startpunkten
    //Schleife zur waagerechten Dachbelegung
    if (checkBox_Rechteck.Checked == true)
    {
        #region Koordinaten der Innenpunkte berechnen und Kontrollzeichnung anlegen
        #region Koordinaten der inneren Punkte berechnen
        //Innenpunkt IP(0)
        iP_0_X = punkt1X + Rand_Abstand_von_Y;
        iP_0_Y = punkt1Y + Rand_Abstand_von_X;

        //Innenpunkt IP(1)
        iP_1_X = punkt2X - Rand_Abstand_von_Y;
        iP_1_Y = punkt1Y + Rand_Abstand_von_X;

        //Innenpunkt IP(2)
        iP_2_X = punkt2X - Rand_Abstand_von_Y;
        iP_2_Y = punkt2Y - Rand_Abstand_von_X;

        //Innenpunkt IP(3)
        iP_3_X = punkt1X + Rand_Abstand_von_Y;
        iP_3_Y = punkt2Y - Rand_Abstand_von_X;
        #endregion

        #region Kontrollzeichnung Innenrechteck
        //Rechteck auf Zeichenunterlage zeichnen (mit Hilfe von Höhe und Breite)
        Grafik.DrawRectangle(Pens.Green, iP_0_X, iP_0_Y, (iP_2_X - iP_0_X), (iP_2_Y -
iP_0_Y));
        #endregion
```

```
#endregion

#region Berechnung der maximal legbaren Module
//Berechnung der maximal montierbaren Module in waagerechter Richtung
maxAnzModule_X = (int)((iP_1_X - iP_0_X) / modulEinheit);

//Berechnung der maximal montierbaren Module in senkrechter Richtung
maxAnzModule_Y = (int)(Math.Abs(iP_2_Y - iP_0_Y) / (modulHoehe +
modulAbstand_senkrecht_Y));
#endregion

#region Switch-Case, um Anfangspunkt zu wählen und Startpunkte auszurechnen
//Switch-Case-Abfrage, in welcher Ecke mit Belegung angefangen werden soll
//Berechnung der Startpunkte für den individuellen Fall
switch (comboBox_Startpunkt.SelectedItem.ToString())
{
    case "oben links":
        startPunkt_Y = iP_0_Y;
        startPunkt_X = iP_0_X;
        break;
    case "oben rechts":
        startPunkt_Y = iP_0_Y;
        startPunkt_X = iP_1_X - (maxAnzModule_X * modulEinheit) +
modulAbstand_waagerecht_X;
        break;
    case "unten links":
        startPunkt_Y = iP_3_Y - (maxAnzModule_Y * (modulHoehe +
modulAbstand_senkrecht_Y)) + modulAbstand_senkrecht_Y;
        startPunkt_X = iP_0_X;
        break;
    case "unten rechts":
        startPunkt_Y = iP_3_Y - (maxAnzModule_Y * (modulHoehe +
modulAbstand_senkrecht_Y)) + modulAbstand_senkrecht_Y;
        startPunkt_X = iP_1_X - (maxAnzModule_X * modulEinheit) +
modulAbstand_waagerecht_X;
        break;
}
#endregion

#region Schleife zum Zeichnen der Module
//variablen, die in der Schleife gebraucht werden
int rasterPunkt_X = 0;
int rasterPunkt_Y = 0;

//Schleife für waagerechte Belegung
for (int i = 0; i < maxAnzModule_X; i++)
```

```

    {
        //Schleife für senkrechte Belegung
        for (int j = 0; j < maxAnzModule_Y; j++)
        {
            rasterPunkt_X = startPunkt_X + i * (modulEinheit);
            rasterPunkt_Y = startPunkt_Y + j * (modulHoehe + modulAbstand_senkrecht_Y);

            //iP_0 bis iP_3 sind die inneren Punkte des Trapezes / Rechteckes/ Dreieckes
            //alles, was mit "rasterPunkt" anfängt, sind die Eckpunkte eines Modules
            if (IsInRoof(iP_0_X, iP_0_Y, iP_1_X, iP_1_Y,
                iP_2_X, iP_2_Y, iP_3_X, iP_3_Y,
                rasterPunkt_X, rasterPunkt_Y, (rasterPunkt_X + modulBreite),
rasterPunkt_Y,
                (rasterPunkt_X + modulBreite), (rasterPunkt_Y + modulHoehe),
rasterPunkt_X, (rasterPunkt_Y + modulHoehe)))
            {
                Grafik.DrawRectangle(Pens.DarkBlue, (rasterPunkt_X), (rasterPunkt_Y),
modulBreite, modulHoehe);
                modulCounter++;
            }
        }
    }
    #endregion

}
#endregion
    
```

```

#region Schleife zur Vollbelegung von Trapez-Dächern
if (checkBox_Trapez.Checked == true)
{
    #region Seitenwinkel ("alpha") des Trapezes berechnen und ausgeben
    //
    //      /-----\
    //     /           \
    //    /             \
    //   /alpha         \
    //  /-----\
    //notwendige Variablen zur Berechnung des Seitenwinkels
    double winkelVerhaeltnisRAD = 0;
    double winkelGrad = 0;
    double quotientSeitendreieck = 0;

    //Berechnung des Seitenwinkels ("alpha") des Trapezes
    //Quotient der Seiten berechnen (Vorbereitung, um Co-Tangens anzuwenden
(ArcusTangens)
    quotientSeitendreieck = (double)(punkt3Y - punkt1Y) / (double)(punkt3X - punkt2X);
    
```

```
//ArcusTanges (Co-Tangens) berechnen (ACHTUNG!!! WIRD IN BOGENMASS
(RAD) ANGEGEBEN!!!!)
winkelVerhaeltnisRAD = Math.Atan(quotientSeitendreieck);

//Umrechnung von Bogenmaß in Grad und Ausgabe in MessageBox
winkelGrad = Math.Abs(winkelVerhaeltnisRAD * (180 / Math.PI));
#endregion

#region Koordinaten des inneren Trapezes
#region Koordinaten der inneren Punkte berechnen
//Innenpunkt IP(0)
iP_0_X = Convert.ToInt16(punkt1X + Rand_Abstand_von_Y -
    (Rand_Abstand_von_X / (Math.Tan(winkelVerhaeltnisRAD))));
iP_0_Y = punkt1Y + Rand_Abstand_von_X;

//Innenpunkt IP(1)
iP_1_X = Convert.ToInt16(punkt2X - Rand_Abstand_von_Y +
    (Rand_Abstand_von_X / (Math.Tan(winkelVerhaeltnisRAD))));
iP_1_Y = punkt1Y + Rand_Abstand_von_X;

//Innenpunkt IP(2)
iP_2_X = Convert.ToInt16(punkt3X - Rand_Abstand_von_Y -
    (Rand_Abstand_von_X / (Math.Tan(winkelVerhaeltnisRAD))));
iP_2_Y = punkt3Y - Rand_Abstand_von_X;

//Innenpunkt IP(3)
iP_3_X = Convert.ToInt16((punkt1X - (punkt3X - punkt2X)) + Rand_Abstand_von_Y
+
    (Rand_Abstand_von_X / (Math.Tan(winkelVerhaeltnisRAD))));
iP_3_Y = punkt3Y - Rand_Abstand_von_X;
#endregion

#region Kontrollzeichnung des inneren Trapezes
//Linie von iP(0) zu iP(1)
Grafik.DrawLine(Pens.Green, iP_0_X, iP_0_Y, iP_1_X, iP_1_Y);
//Linie von iP(1) zu iP(2)
Grafik.DrawLine(Pens.Green, iP_1_X, iP_1_Y, iP_2_X, iP_2_Y);
//Linie von iP(2) zu iP(3)
Grafik.DrawLine(Pens.Green, iP_2_X, iP_2_Y, iP_3_X, iP_3_Y);
//Linie von iP(3) zu iP(0)
Grafik.DrawLine(Pens.Green, iP_3_X, iP_3_Y, iP_0_X, iP_0_Y);
#endregion
#endregion

#region Anfangspunkt wählen und Startpunkte berechnen
if (comboBox_Startpunkt.SelectedItem.ToString() == "oben links")
{
    #region Raster-Startpunkt errechnen (oben links)
```

```
//Startpunkt des Rasters ausrechnen
startPunkt_Y = iP_0_Y;

//Breite Seite nach oben (verkehrt herum)
if (iP_0_X < iP_3_X)
{
    startPunkt_X = iP_0_X;
}
//Breite Seite ist Standfläche (richtig rum)
else
{
    //Anzahl der Module zwischen P(0x) und P(3x) (Suche des Raster-
Anfangspunktes)
    int anzahl = (int)(Math.Abs(iP_0_X - iP_3_X) / modulEinheit);

    startPunkt_X = iP_0_X - (anzahl * modulEinheit);
}
#endregion
}
else if (comboBox_Startpunkt.SelectedItem.ToString() == "oben rechts")
{
    #region Raster-Startpunkt errechnen (oben rechts)
    //Startpunkt des Rasters ausrechnen
    startPunkt_Y = iP_0_Y;

    //Breite Seite nach oben (verkehrt herum)
    if (iP_0_X < iP_3_X)
    {
        int anzahl = (int)((iP_1_X - iP_0_X) / modulEinheit);
        startPunkt_X = iP_1_X - (anzahl * modulEinheit) +
modulAbstand_waagrecht_X;
    }
    //Breite Seite ist Standfläche (richtig rum)
    else
    {
        int anzahl = (int)((iP_1_X - iP_3_X) / modulEinheit);
        startPunkt_X = iP_1_X - (anzahl * modulEinheit) +
modulAbstand_waagrecht_X;
    }
    #endregion
}
else if (comboBox_Startpunkt.SelectedItem.ToString() == "unten links")
{
    #region Raster-Startpunkt errechnen (unten links)
    int anzahl_Y = (int)((iP_2_Y - iP_0_Y) / (modulHoehe +
modulAbstand_senkrecht_Y));

    //Startpunkt des Rasters ausrechnen
```

```
startPunkt_Y = iP_2_Y - (anzahl_Y * (modulHoehe + modulAbstand_senkrecht_Y))
+ modulAbstand_senkrecht_Y;

//Breite Seite nach oben (verkehrt herum)
if (iP_0_X < iP_3_X)
{
    startPunkt_X = iP_0_X;
}
//Breite Seite ist Standfläche (richtig rum)
else
{
    //Anzahl der Module zwischen P(0x) und P(3x) (Suche des Raster-
Anfangspunktes)
    int anzahl = (int)(Math.Abs(iP_0_X - iP_3_X) / modulEinheit);

    startPunkt_X = iP_0_X - (anzahl * modulEinheit);
}
#endregion
}
else if (comboBox_Startpunkt.SelectedItem.ToString() == "unten rechts")
{
    #region Raster-Startpunkt errechnen (unten rechts)
    int anzahl_Y = (int)((iP_2_Y - iP_0_Y) / (modulHoehe +
modulAbstand_senkrecht_Y));

    //Startpunkt des Rasters ausrechnen
    startPunkt_Y = iP_2_Y - (anzahl_Y * (modulHoehe + modulAbstand_senkrecht_Y))
+ modulAbstand_senkrecht_Y;

    //Breite Seite nach oben (verkehrt herum)
    if (iP_0_X < iP_3_X)
    {
        int anzahl = (int)((iP_1_X - iP_0_X) / modulEinheit);
        startPunkt_X = iP_1_X - (anzahl * modulEinheit) +
modulAbstand_waagrecht_X;
    }
    //Breite Seite ist Standfläche (richtig rum)
    else
    {
        int anzahl = (int)((iP_1_X - iP_3_X) / modulEinheit);
        startPunkt_X = iP_1_X - (anzahl * modulEinheit) +
modulAbstand_waagrecht_X;
    }
    #endregion
}
#endregion

#region Berechnung der maximal legbaren Module
```

```
//Berechnung der maximal montierbaren Module in waagerechter Richtung
maxAnzModule_X = (int)((Math.Max(iP_2_X, iP_1_X) - startPoint_X +
modulAbstand_waagerecht_X) / modulEinheit);

//Berechnung der maximal montierbaren Module in senkrechter Richtung
maxAnzModule_Y = (int)((iP_2_Y - iP_0_Y) / (modulHoehe +
modulAbstand_senkrecht_Y));

#endregion

#region Schleife zum Zeichnen der Module
//variablen, die in der Schleife gebraucht werden
int rasterPunkt_X = 0;
int rasterPunkt_Y = 0;

//Schleife für waagerechte Belegung
for (int i = 0; i < maxAnzModule_X; i++)
{
    //Schleife für senkrechte Belegung
    for (int j = 0; j < maxAnzModule_Y; j++)
    {
        rasterPunkt_X = startPoint_X + i * (modulEinheit);
        rasterPunkt_Y = startPoint_Y + j * (modulHoehe + modulAbstand_senkrecht_Y);

        //iP_0 bis iP_3 sind die inneren Punkte des Trapezes / Rechteckes/ Dreieckes
        //alles, was mit "rasterPunkt" anfängt, sind die Eckpunkte eines Modules
        if (IsInRoof(iP_0_X, iP_0_Y, iP_1_X, iP_1_Y,
                    iP_2_X, iP_2_Y, iP_3_X, iP_3_Y,
                    rasterPunkt_X, rasterPunkt_Y, (rasterPunkt_X + modulBreite),
rasterPunkt_Y,
                    (rasterPunkt_X + modulBreite), (rasterPunkt_Y + modulHoehe),
rasterPunkt_X, (rasterPunkt_Y + modulHoehe)))
        {
            Grafik.DrawRectangle(Pens.DarkBlue, (rasterPunkt_X), (rasterPunkt_Y),
modulBreite, modulHoehe);
            modulCounter++;
        }
    }
}
#endregion

}
#endregion

#region Schleife zur Vollbelegung von Dreiecks-Dächern
if (checkBox_Dreieck.Checked == true)
```



```
{
    #region Seitenwinkel ("alpha") und Spitzenwinkel ("gamma") des Dreiecks berechnen
    und ausgeben
    //      gamma
    //      / \
    //      / \
    //      / \
    //      /alpha \
    //      /_____\
    //notwendige Variablen zur Berechnung des Seitenwinkels
    double winkelVerhaeltnisRAD = 0;
    double winkelGrad = 0;
    double quotientSeitendreieck = 0;

    double spitzenWinkel = 0;
    double beta = 0;
    double betaRAD = 0;

    //Berechnung des Seitenwinkels ("alpha") des Dreiecks
    //Quotient der Seiten berechnen (Vorbereitung, um Co-Tangens anzuwenden
    (ArcusTangens)
    quotientSeitendreieck = (double)((punkt2Y - punkt1Y) / (double)Math.Abs(punkt2X -
    (punkt1X - (punkt2X - punkt1X))));

    //ArcusTanges (Co-Tangens) berechnen (ACHTUNG!!! WIRD IN BOGENMASS
    (RAD) ANGEGEBEN!!!!)
    winkelVerhaeltnisRAD = Math.Atan(quotientSeitendreieck);

    //Umrechnung von Bogenmaß in Grad und Ausgabe in MessageBox
    winkelGrad = Math.Abs(winkelVerhaeltnisRAD * (180 / Math.PI));

    //Berechnung des Spitzenwinkels "gamma" des Dreiecks
    //"gamma" mit Hilfe der Winkelsumme im Dreieck berechnen
    spitzenWinkel = 180 - (2 * winkelGrad);

    //halber Spitzenwinkel "beta"
    beta = spitzenWinkel / 2;

    betaRAD = ((beta * Math.PI) / 180);

    #endregion

    #region Koordinaten des inneren Dreiecks
    #region Koordinaten der inneren Punkte berechnen
    if (punkt1Y > punkt2Y)
    {
        //Innenpunkt IP(0)
        iP_0_X = Convert.ToInt16(punkt2X - Rand_Abstand_von_Y -
```

```
(Rand_Abstand_von_X / (Math.Tan(winkelVerhaeltnisRAD))));
iP_0_Y = punkt2Y - Rand_Abstand_von_X;

//Innenpunkt IP(1)
iP_1_X = Convert.ToInt16((punkt1X - (punkt2X - punkt1X) +
Rand_Abstand_von_Y +
(Rand_Abstand_von_X / (Math.Tan(winkelVerhaeltnisRAD))));
iP_1_Y = punkt2Y - Rand_Abstand_von_X;

//Innenpunkt IP(2)
iP_2_X = punkt1X;
iP_2_Y = Convert.ToInt16(punkt1Y + Rand_Abstand_von_X +
(Rand_Abstand_von_Y / Math.Sin(betaRAD)));

//Innenpunkt IP(3)
iP_3_X = iP_0_X;
iP_3_Y = iP_0_Y;
}
else
{
    //Innenpunkt IP(0)
    iP_0_X = punkt1X;
    iP_0_Y = Convert.ToInt16(punkt1Y + Rand_Abstand_von_X +
(Rand_Abstand_von_Y / Math.Sin(betaRAD)));

    //Innenpunkt IP(1)
    iP_1_X = iP_0_X;
    iP_1_Y = iP_0_Y;

    //Innenpunkt IP(2)
    iP_2_X = Convert.ToInt16(punkt2X - Rand_Abstand_von_Y -
(Rand_Abstand_von_X / (Math.Tan(winkelVerhaeltnisRAD))));
    iP_2_Y = punkt2Y - Rand_Abstand_von_X;

    //Innenpunkt IP(3)
    iP_3_X = Convert.ToInt16((punkt1X - (punkt2X - punkt1X) +
Rand_Abstand_von_Y +
(Rand_Abstand_von_X / (Math.Tan(winkelVerhaeltnisRAD))));
    iP_3_Y = punkt2Y - Rand_Abstand_von_X;
}
#endregion

#region Kontrollzeichnung des inneren Dreiecks
//Linie von iP(0) zu iP(1)
Grafik.DrawLine(Pens.Green, iP_0_X, iP_0_Y, iP_2_X, iP_2_Y);
//Linie von iP(1) zu iP(2)
Grafik.DrawLine(Pens.Green, iP_2_X, iP_2_Y, iP_3_X, iP_3_Y);
//Linie von iP(2) zu iP(0)
Grafik.DrawLine(Pens.Green, iP_3_X, iP_3_Y, iP_0_X, iP_0_Y);
```

```
#endregion
#endregion

#region Anfangspunkt wählen und Startpunkte berechnen
//Belegung für Dreieck mit Spitze nach oben ist bei "oben links" und "oben rechts"
identisch
//Die Belegung durch das Raster fängt quasi bei x=x(3) und y=y(0) an
if (comboBox_Startpunkt.SelectedItem.ToString() == "oben links" ||
comboBox_Startpunkt.SelectedItem.ToString() == "oben rechts")
{
    //Anzahl der Module zwischen P(0x) und P(3x) (Suche des Raster-Anfangspunktes)
    int anzahl = (int)(Math.Abs(iP_0_X - iP_3_X) / modulEinheit);

    #region Raster-Startpunkt errechnen (oben links)
    //Startpunkt des Rasters ausrechnen
    startPunkt_Y = iP_0_Y;
    startPunkt_X = iP_0_X - (anzahl * modulEinheit);
    #endregion
}

//Die Belegung durch das Raster fängt in der unteren linken Ecke (bei iP(3)) an
else if (comboBox_Startpunkt.SelectedItem.ToString() == "unten links")
{
    //Anzahl der Module zwischen P(0x) und P(3x) (Suche des Raster-Anfangspunktes)
    //int anzahl = (int)(Math.Abs(iP_2_X - iP_3_X) / modulEinheit);
    int anzahl_Y = (int)((iP_2_Y - iP_0_Y) / (modulHoehe +
modulAbstand_senkrecht_Y));

    #region Raster-Startpunkt errechnen (oben links)
    //Startpunkt des Rasters ausrechnen
    startPunkt_Y = iP_3_Y - (anzahl_Y * (modulHoehe + modulAbstand_senkrecht_Y))
+ modulAbstand_senkrecht_Y;
    startPunkt_X = iP_3_X;

    #endregion
}

//Ein Modul muss der unteren rechten Ecke (bei iP(2)) liegen
//Der Startpunkt wird zum Aufziehen des Rasters entsprechen zurückgerechnet
else if (comboBox_Startpunkt.SelectedItem.ToString() == "unten rechts")
{
    //Anzahl der Module zwischen P(0x) und P(3x) (Suche des Raster-Anfangspunktes)
    int anzahl = (int)(Math.Abs(iP_2_X - iP_3_X) / modulEinheit);
    int anzahl_Y = (int)((iP_2_Y - iP_0_Y) / (modulHoehe +
modulAbstand_senkrecht_Y));

    #region Raster-Startpunkt errechnen (oben links)
    //Startpunkt des Rasters ausrechnen
```

```
        startPunkt_Y = iP_3_Y - (anzahl_Y * (modulHoehe + modulAbstand_senkrecht_Y))
+ modulAbstand_senkrecht_Y;
        startPunkt_X = iP_2_X - (anzahl * modulEinheit);
        #endregion

    }
    #endregion

    #region Berechnung der maximal legbaren Module
    //Berechnung der maximal montierbaren Module in waagerechter Richtung
    maxAnzModule_X = (int)((Math.Max(iP_2_X, iP_0_X) - startPunkt_X +
modulAbstand_waagerecht_X) / modulEinheit);

    //Berechnung der maximal montierbaren Module in senkrechter Richtung
    maxAnzModule_Y = (int)((iP_2_Y - startPunkt_Y + modulAbstand_senkrecht_Y) /
(modulHoehe + modulAbstand_senkrecht_Y));
    #endregion

    #region Schleife zum Zeichnen der Module
    //variablen, die in der Schleife gebraucht werden
    int rasterPunkt_X = 0;
    int rasterPunkt_Y = 0;

    //Schleife für waagerechte Belegung
    for (int i = 0; i < maxAnzModule_X; i++)
    {
        //Schleife für senkrechte Belegung
        for (int j = 0; j < maxAnzModule_Y; j++)
        {
            rasterPunkt_X = startPunkt_X + i * (modulEinheit);
            rasterPunkt_Y = startPunkt_Y + j * (modulHoehe + modulAbstand_senkrecht_Y);

            //iP_0 bis iP_3 sind die inneren Punkte des Trapezes / Rechteckes/ Dreieckes
            //alles, was mit "rasterPunkt" anfängt, sind die Eckpunkte eines Modules
            if (IsInRoof(iP_0_X, iP_0_Y, iP_1_X, iP_1_Y,
                iP_2_X, iP_2_Y, iP_3_X, iP_3_Y,
                rasterPunkt_X, rasterPunkt_Y, (rasterPunkt_X + modulBreite),
rasterPunkt_Y,
                (rasterPunkt_X + modulBreite), (rasterPunkt_Y + modulHoehe),
rasterPunkt_X, (rasterPunkt_Y + modulHoehe)))
            {
                Grafik.DrawRectangle(Pens.DarkBlue, (rasterPunkt_X), (rasterPunkt_Y),
modulBreite, modulHoehe);
                modulCounter++;
            }
        }
    }
}
```

```
        #endregion

    }

    #endregion

    textBox_Gesamtanzahl_Module.Text = Convert.ToString(modulCounter);

    //Zeichnen- und Modul-Button ausgrauen, damit nicht übereinander gezeichnet werden
kann!
    btnModulZeichnen.Enabled = false;
    btnZeichnen.Enabled = false;
}

//Abfrage, ob Modul auf der zeichenbaren Dachfläche liegt
#region Boolesche Abfrage, ob Modul auf Trapez-Dachfläche liegt
///<summary>
/// Boolesche Abfrage, ob Modul auf Dachfläche liegt oder nicht          rp0_____rp1
/// iP = Innenpunkte des Trapezes, die ausgerechnet wurden              |   |
/// Rasterpunkte sind die Eckpunkte der Module, die auf dem Dach liegen  rp3_____rp2
///</summary>
private bool IsInRoof(int iP_0_X, int iP_0_Y, int iP_1_X, int iP_1_Y,
                    int iP_2_X, int iP_2_Y, int iP_3_X, int iP_3_Y,
                    int rasterPunkt_0_X, int rasterPunkt_0_Y, int rasterPunkt_1_X, int
rasterPunkt_1_Y,
                    int rasterPunkt_2_X, int rasterPunkt_2_Y, int rasterPunkt_3_X, int
rasterPunkt_3_Y)
{
    //notwendige Variablen für diese Funktion
    int xPos_erlaubt_links = 0;
    int xPos_erlaubt_rechts = 0;

    //Y-Wert des Punktes liegt oberhalb der Dachoberkante
    if (rasterPunkt_0_Y < iP_0_Y)
    {
        return false;
    }

    //Y-Wert liegt unterhalb der Dachunterkante
    if (rasterPunkt_3_Y > iP_2_Y)
    {
        return false;
    }

    //schmale Seite des Trapezes ist unten (Trapez umgedreht)
    //unterer linker Punkt ragt über das Trapez hinaus
```

```
    if (iP_0_X < iP_3_X)
    {
        xPos_erlaubt_links = iP_0_X + (((iP_3_X - iP_0_X) * (rasterPunkt_3_Y - iP_0_Y)) /
(iP_3_Y - iP_0_Y));

        if (xPos_erlaubt_links > rasterPunkt_3_X)
        {
            return false;
        }
    }
    //Trapez mit breiter Seite nach unten
    //oberer linker Punkt ragt über das Trapez hinaus
    else
    {
        xPos_erlaubt_links = iP_0_X + (((iP_3_X - iP_0_X) * (rasterPunkt_0_Y - iP_0_Y)) /
(iP_3_Y - iP_0_Y));

        if (xPos_erlaubt_links > rasterPunkt_0_X)
        {
            return false;
        }
    }

    //schmale Trapezseite ist unten (Trapez umgedreht)
    //untere rechte Ecke ragt über das Trapez hinaus
    if (iP_1_X > iP_2_X)
    {
        xPos_erlaubt_rechts = iP_1_X - (((iP_1_X - iP_2_X) * (rasterPunkt_2_Y - iP_1_Y)) /
(iP_2_Y - iP_1_Y));

        if (xPos_erlaubt_rechts < rasterPunkt_2_X)
        {
            return false;
        }
    }
    //Trapez mit breiter Seite nach unten
    //obere rechte Ecke ragt über das Trapez hinaus
    else
    {
        xPos_erlaubt_rechts = iP_1_X + (((iP_2_X - iP_1_X) * (rasterPunkt_1_Y - iP_1_Y)) /
(iP_2_Y - iP_1_Y));

        if (xPos_erlaubt_rechts < rasterPunkt_1_X)
        {
            return false;
        }
    }
    return true;
}
```

#endregion

```
//löscht alle gezeichneten Module
//Dachfläche bleibt bestehen
//Abstands-Werte sind änderbar und Module neu zeichnenbar
private void btnModuleLoeschen_Click(object sender, EventArgs e)
{
```

```
    //Gesamte Zeichenfläche löschen
    Grafik.Clear(System.Drawing.SystemColors.Control);
```

```
    #region Module auf Rechteck-Dach löschen
```

```
    //Rechteck-Dachfläche neu zeichnen
    if (checkBox_Rechteck.Checked == true)
    {
        btnModulZeichnen.Enabled = true;
```

```
        //Berechnung der Bemaßungen
        hoehe = ObjectRechteck.HoeheBerechnen();
        breite = ObjectRechteck.BreiteBerechnen();
```

```
        //Rechteck auf Zeichenunterlage zeichnen (mit Hilfe von Höhe und Breite)
        Grafik.DrawRectangle(Pens.Red, punkt1X, punkt1Y, breite, hoehe);
    }
```

```
    #endregion
```

```
    #region Module auf Trapez-Dach löschen
```

```
    //Trapez-Dachfläche neu zeichnen
    if (checkBox_Trapez.Checked == true)
    {
        btnModulZeichnen.Enabled = true;
```

```
        //Trapez mit Hilfe von 3 angeklickten Punkten zeichnen; 4. Punkt errechnen
        Grafik.DrawLine(Pens.Red, punkt1X, punkt1Y, punkt2X, punkt1Y);
        Grafik.DrawLine(Pens.Red, punkt2X, punkt1Y, punkt3X, punkt3Y);
        Grafik.DrawLine(Pens.Red, punkt3X, punkt3Y, (punkt1X - (punkt3X - punkt2X)),
punkt3Y);
        Grafik.DrawLine(Pens.Red, (punkt1X - (punkt3X - punkt2X)), punkt3Y, punkt1X,
punkt1Y);
    }
```

```
    #endregion
```

```
    #region Module auf Dreiecks-Dach löschen
```

```
    //Dreieck-Dachfläche neu zeichnen
    if (checkBox_Dreieck.Checked == true)
    {
        btnModulZeichnen.Enabled = true;
```

```
        //Zeichnet gleichschenkliges Dreieck nach Angabe von 2 Punkten durch klicken
```

```
        Grafik.DrawLine(Pens.Red, punkt1X, punkt1Y, punkt2X, punkt2Y);
        Grafik.DrawLine(Pens.Red, punkt2X, punkt2Y, (punkt1X - (punkt2X - punkt1X)),
punkt2Y);
        Grafik.DrawLine(Pens.Red, (punkt1X - (punkt2X - punkt1X)), punkt2Y, punkt1X,
punkt1Y);
    }
    #endregion

    //Modulanzahl-Felder wieder leeren
    textBox_Gesamtanzahl_Module.Text = "";
    //Combobox-Startpunkt wieder auf Anfang setzen
    comboBox_Startpunkt.SelectedItem = "oben links";
}
}
```

//Klasse RechteckDach merkt die geklickten Punkte und berechnet Breite und Höhe des Rechtecks

```
class CRechteckDach
{
    #region Punkte Rechteck merken
    //Punkt 1 merken
    public int loc_punkt1X;
    public int loc_punkt1Y;

    //Punkt 2 merken
    public int loc_punkt2X;
    public int loc_punkt2Y;
    #endregion

    public int breiteRechteck;
    public int hoeheRechteck;

    #region getter und setter für Rechteck-Variablen
    //Variablen mit get und set belegen
    public int BreiteDerFigur
    {
        get
        {
            return breiteRechteck;
        }
        set
        {
            breiteRechteck = value;
        }
    }

    public int HoeheDerFigur
    {
```



```
    get
    {
        return hoeheRechteck;
    }
    set
    {
        hoeheRechteck = value;
    }
}
#endregion

#region Methoden zur Berechnung der Maße und Zeichnung
/// <summary>
/// Berechnung der Höhe des Rechtecks
/// </summary>
public int HoeheBerechnen()
{
    hoeheRechteck = (loc_punkt2Y - loc_punkt1Y);
    return hoeheRechteck;
}

/// <summary>
/// Berechnung der Breite des Trapezes
/// </summary>
public int BreiteBerechnen()
{
    breiteRechteck = (loc_punkt2X - loc_punkt1X);
    return breiteRechteck;
}
#endregion

#region Konstruktor und Destruktor
//Default-Konstruktor
public CRechteckDach()
{
}

//Default-Destruktor
~CRechteckDach()
{
}
#endregion
}

/// <summary>
/// Klasse TrapezDach merkt die geklickten Punkte und berechnet Höhe,
/// Breite (oben und unten), sowie die Seitenlänge
/// </summary>
```

```
class CTrapezDach
{
    #region Punkte von Trapez merken
    //Punkt 1 merken
    public int loc_punkt1X;
    public int loc_punkt1Y;

    //Punkt 2 merken
    public int loc_punkt2X;
    public int loc_punkt2Y;

    //Punkt 3 merken
    public int loc_punkt3X;
    public int loc_punkt3Y;
    #endregion

    private int obereBreiteTrapez = 0;
    private int untereBreiteTrapez = 0;
    private int hoeheTrapez = 0;
    private int seitenlaenge = 0;

    #region getter und setter für Trapez-Variablen
    public int BreiteObenTrapez
    {
        get
        {
            return obereBreiteTrapez;
        }
        set
        {
            obereBreiteTrapez = value;
        }
    }

    public int BreiteUntenTrapez
    {
        get
        {
            return untereBreiteTrapez;
        }
        set
        {
            untereBreiteTrapez = value;
        }
    }

    public int HoeheTrapez
    {
        get
```

```
{
    return hoeheTrapez;
}
set
{
    hoeheTrapez = value;
}
}

public int SeitenlaengeTrapez
{
    get
    {
        return seitenlaenge;
    }
    set
    {
        seitenlaenge = value;
    }
}
#endregion

#region Methoden zur Berechnung der Maße und Zeichnung
/// <summary>
/// Berechnung der Grundseite des Trapezes
/// </summary>
public int UntereBreiteBerechnen()
{
    untereBreiteTrapez = (loc_punkt3X - (loc_punkt1X - (loc_punkt3X - loc_punkt2X)));
    return untereBreiteTrapez;
}

/// <summary>
/// Berechnung der oberen Seite des Trapezes
/// </summary>
public int ObereBreiteBerechnen()
{
    obereBreiteTrapez = (loc_punkt2X - loc_punkt1X);
    return obereBreiteTrapez;
}

/// <summary>
/// Berechnung der Höhe des Trapezes
/// </summary>
public int HoeheBerechnen()
{
    hoeheTrapez = (loc_punkt3Y - loc_punkt1Y);
    return hoeheTrapez;
}
```

```
/// <summary>
/// Berechnung der Seitenlänge eines Trapezes mit Pythagoras-Formel
/// </summary>
public int SeitenlaengeTrapezBerechnen()
{
    //Pythagoras zum Berechnen der Seitenkante
    seitenlaenge = Convert.ToInt16(Math.Sqrt(((loc_punkt3X - loc_punkt2X) * (loc_punkt3X -
loc_punkt2X)) + ((loc_punkt3Y - loc_punkt1Y) * (loc_punkt3Y - loc_punkt1Y))));
    return seitenlaenge;
}
#endregion

#region Konstruktor und Destruktor
//Default-Konstruktor
public CTrapezDach()
{
}

//Default-Destruktor
~CTrapezDach()
{
}
#endregion
}

/// <summary>
/// Klasse DreieckDach merkt die geklickten Punkte und berechnet Höhe, Breite (Grundseite)
/// und Seitenlängen
/// </summary>
class CDreieckDach
{
    #region Punkte Dreieck merken
    //Punkt 1 merken
    public int loc_punkt1X;
    public int loc_punkt1Y;

    //Punkt 2 merken
    public int loc_punkt2X;
    public int loc_punkt2Y;
    #endregion

    private int grundseiteDreieck = 0;
    private int hoehe = 0;
    private int seitenkanteDreieck = 0;

    #region getter und setter für Rechteck-Variablen
    //Variablen mit get und set belegen
```

```
public int GrundseiteVomDreieck
{
    get
    {
        return grundseiteDreieck;
    }
    set
    {
        grundseiteDreieck = value;
    }
}

public int hoeheDreieck
{
    get
    {
        return hoehe;
    }
    set
    {
        hoehe = value;
    }
}

public int SeitenkanteDreieck
{
    get
    {
        return seitenkanteDreieck;
    }
    set
    {
        seitenkanteDreieck = value;
    }
}
#endregion

#region Methoden zur Berechnung der Maße und Zeichnung
/// <summary>
/// Berechnung der Grundseite eines Dreieck
/// </summary>
public int GrundseiteDreieckBerechnen()
{
    grundseiteDreieck = Math.Abs((loc_punkt2X - loc_punkt1X));
    return grundseiteDreieck;
}

/// <summary>
/// Berechnung der Seitenlänge eines Dreiecks mit Pythagoras-Formel
```

```
/// </summary>
public int SeitenkanteDreieckBerechnen()
{
    //Pythagoras zum Berechnen der Seitenkante
    seitenkanteDreieck = Convert.ToInt16(Math.Abs(Math.Sqrt(((loc_punkt2X - loc_punkt1X)
* (loc_punkt2X - loc_punkt1X)) + ((loc_punkt2Y - loc_punkt1Y) * (loc_punkt2Y -
loc_punkt1Y)))));
    return seitenkanteDreieck;
}

/// <summary>
/// Berechnung der Höhe eines Dreiecks
/// </summary>
public int HoeheDreieckBerechnen()
{
    hoehe = Math.Abs((loc_punkt2Y - loc_punkt1Y));
    return hoehe;
}
#endregion

#region Konstruktor und Destruktor
//Default-Konstruktor
public CDreieckDach()
{
}

//Default-Destruktor
~CDreieckDach()
{
}
#endregion
}

//Klasse Modul speichert die Breite und die Höhe der Module, damit diese gezeichnet werden
können
class CModul
{
    //Punkt 1 merken
    public int loc_punkt1X = 0;
    public int loc_punkt1Y = 0;

    //Punkt 2 merken
    public int loc_punkt2X = 0;
    public int loc_punkt2Y = 0;

    private int breiteModul = 0;
    private int hoeheModul = 0;
```

```
private int modulAnzahl = 0;

#region Getter und setter für die Variablen
//getter und setter für Modul-Variablen
public int BreiteDesModuls
{
    get
    {
        return breiteModul;
    }
    set
    {
        breiteModul = value;
        //breiteModul = Convert.ToInt16(TextBox_ModulBreite.Text);
    }
}

public int HoeheDesModuls
{
    get
    {
        return hoeheModul;
    }
    set
    {
        hoeheModul = value;
    }
}

public int ModulAnzahl
{
    get
    {
        return modulAnzahl;
    }
    set
    {
        modulAnzahl = value;
    }
}
#endregion

#region Konstruktor und Destruktor für die Module
//Konstruktor, der Breite und Höhe übergibt
public CModul()
{
    breiteModul = 20;
    hoeheModul = 25;
}
```

```
    }  
  
    ~CModul()  
    {  
    }  
    #endregion  
}  
  
}
```




Entwicklerdokumentation

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 12.03.2013

1. Modulbeschreibung

Der „Algorithmus zur Vollbelegung von Dächern mit Photovoltaik-Modulen“ ist keine eigenständige Anwendung. Es handelt sich um einen [Prototypen](#) mit vollständig implementierter Berechnung, aber einer vorläufigen Benutzer-Oberfläche zur Eingabe der Startparameter und zur Visualisierung der Ergebnisse. Um die Eingabe der Dachflächen, der Modul-Daten und der Abstände nachvollziehbar simulieren zu können, muss eine Klasse erstellt werden, die sämtliche Zeichenfunktionen und Button-Aktivitäten enthält. Dies ist die Klasse „Zeichenunterlage“, die an die Form der Benutzer-Oberfläche gekoppelt ist. In dieser Klasse sind die Aktionen integriert, die durch den Anwender vorgenommen werden können. Sie enthält die Funktionen, die bei der Betätigung der Schaltflächen „btnZeichnen“, „btnLöschen“, „btnModuleZeichnen“ und „btnModuleLöschen“ aufgerufen werden. Weiter ist in dieser Klasse die Anlage und Typenbezeichnung der allgemein verwendeten Variablen verzeichnet. Zudem wird hier die Combobox mit Inhalten gefüllt und sichergestellt, dass nur eine Dachform mit einem Haken in der Checkbox versehen ist. Die jeweils nicht gewählten beiden Checkboxes müssen leer sein. Die Methode „IsInRoof“ ist eine Abfrage, die sicherstellt, dass das Raster der berechneten Module auch tatsächlich innerhalb der gezeichneten Dachfläche (unter Berücksichtigung der Randabstände) liegt, denn nur dann dürfen die Module an der Benutzer-Oberfläche gezeichnet werden.

Um die Dachflächen abbilden und berechnen zu können, wird für jede Dachform eine eigene Klasse angelegt: „CRechteckDach“, „CTrapezDach“ und „CDreieckDach“. Diese enthalten jeweils Variablen, in welchen die Werte der angeklickten Koordinaten lokal gespeichert werden. Mit diesen werden fehlende Angaben (z.B. Höhe und Breite der Dachfläche) berechnet. Die Ergebnisse stehen wiederum der Haupt-Klasse (Zeichenunterlage) zur Verfügung.

Die Module, mit denen die Dachflächen belegt werden, werden durch die Klasse „CModul“ beschrieben. Hier finden keine Berechnungen statt.

2. Methodenbeschreibung

2.1 Zeichenunterlage

Beim Start des Prototyps wird die Methode „Comboboxfuellen“ aufgerufen, die eine Combobox zur Auswahl des Startpunktes zur Vollbelegung erzeugt und die vorgegebenen Startpunkte zur Darstellung an der Benutzer-Oberfläche angezeigt. Die Methoden „checkBox_Rechteck()“, „checkBox_Trapez()“ und „checkBox_Dreieck()“ stellen sicher, dass bei der Auswahl einer Dachfläche wirklich nur eine Form ausgewählt ist. „Zeichenunterlage_MouseClick()“ ist eine Methode, die zur Erfassung der Koordinaten der Punkte benutzt wird, die vom Anwender für die Dachflächen ausgesucht wurden. „btnZeichnen()“ errechnet mit Hilfe der zuvor angeklickten

Eckpunkte der Dachflächen und der gewählten Dachform die fehlenden Punkte und Abmessungen, um diese an der Benutzeroberfläche grafisch (die Kontur) bzw. numerisch (die Bemaßungen) anzuzeigen.

Die Methode „btnModuleZeichnen()“ ruft die auf der Benutzeroberfläche eingegebenen Werte für Modul, Abstände und Wahl des Startpunktes ab. Beim Zeichnen wird die Methode „IsInRoof()“ verwendet. Bei „btnModuleZeichnen()“ wird ein rechteckiges Raster von Modulen mit den gewünschten Größen erstellt. Mit „IsInRoof()“ wird sichergestellt, dass die Module auch tatsächlich in der verfügbaren Dachfläche liegen. Liegt nur ein Eckpunkt eines Moduls minimal außerhalb der vom Anwender festgelegten Dachgrenzen, wird das Modul nicht gezeichnet. Beim Zeichnen der Module zählt eine Variable die positionierten Module und zeigt die Gesamtzahl für den Anwender an der Benutzer-Oberfläche an. Mit „btnModuleLoeschen()“ werden das Modulfeld und die Kontroll-Zeichnung des Randabstandes entfernt. Die Dachfläche und die vom Anwender eingegebenen Daten bleiben an der Benutzer-Oberfläche stehen. Die Methode „btnLoeschen()“ löscht alle Eingaben und Zeichnungen. Vom Anwender veränderbare Werte werden auf festgesetzte [Default-Werte](#) zurückgesetzt.

2.2 CRechteckDach

Diese Klasse enthält Methoden, die Höhe („HoeheBerechnen()“) und Breite („BreiteBerechnen()“) einer rechteckigen Dachfläche ermitteln. Die Ergebnisse sind von der Haupt-Klasse „Zeichenunterlage“ aus aufrufbar und stehen zur Weiterverarbeitung bereit.

2.3 CTrapezDach

Diese Klasse enthält Methoden, die Höhe („HoeheBerechnen()“), Grundseite („UntenBreiteBerechnen()“), Oberkante („ObereBreiteBerechnen()“) und Seitenkante („SeitenlaengeTrapezBerechnen()“) einer trapezförmigen Dachfläche ermitteln. Die Ergebnisse sind von der Haupt-Klasse „Zeichenunterlage“ aus aufrufbar und stehen zur Weiterverarbeitung bereit.

2.4 CDreieckDach

Diese Klasse enthält Methoden, die Höhe („HoeheDreieckBerechnen()“), Grundseite („GrundseiteDreieckBerechnen()“) und Seitenkante („SeitenkanteDreieckBerechnen()“) einer dreieckigen Dachfläche ermitteln. Die Ergebnisse sind von der Haupt-Klasse „Zeichenunterlage“ aus aufrufbar und stehen zur Weiterverarbeitung bereit.



Benutzer-Hilfe

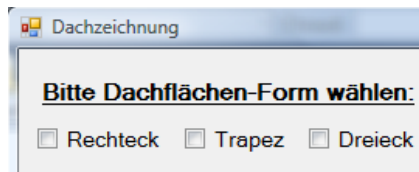
Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 11.03.2013

1. Dachflächen-Form auswählen

Im oberen linken Bereich der Benutzeroberfläche ist eine Dachflächen-Form auszuwählen.



2. Eckpunkte der Dachflächen festlegen und zeichnen

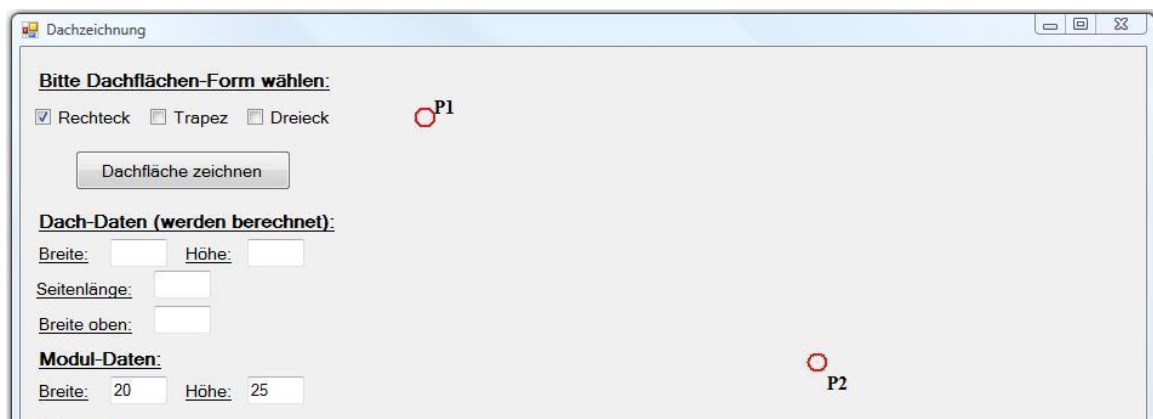
2.1 Rechteckige Dachfläche

Bitte Dachflächen-Form wählen:

☒ Rechteck ☐ Trapez ☐ Dreieck

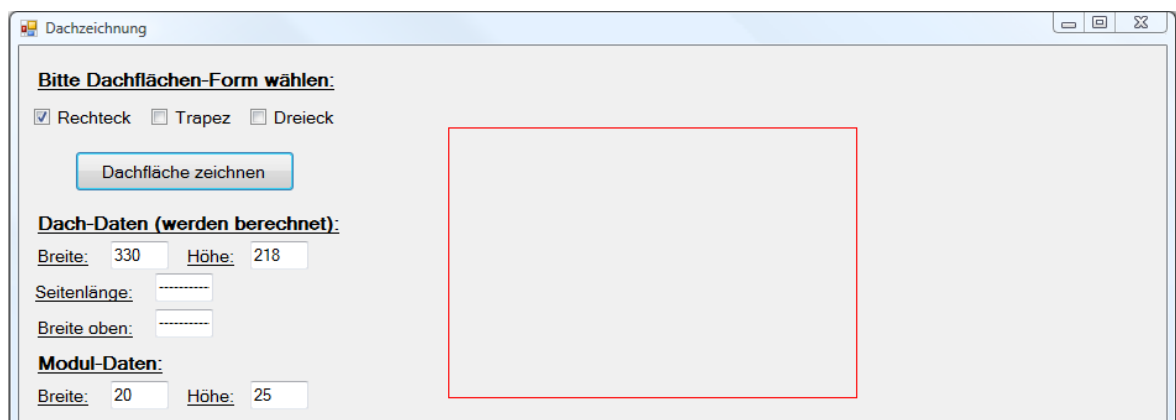
Dachfläche zeichnen

Wird eine rechteckige Dachfläche gewählt, müssen in einem nächsten Schritt zwei Eckpunkte dieser Fläche auf der Zeichenoberfläche festgelegt werden.

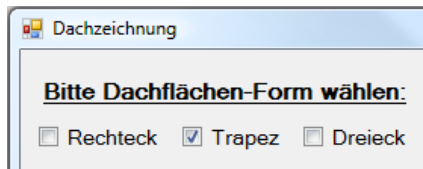


Auf der verfügbaren Zeichenfläche sind für eine rechteckige Dachfläche zwei Punkte zu wählen, mit denen Breite und Höhe der Dachfläche gekennzeichnet werden.

Über die Schaltfläche „*Dachfläche zeichnen*“ wird die Dachfläche angezeigt. Die berechneten Maße der Dachkanten werden berechnet und angezeigt.



2.2 Trapezförmige Dachfläche

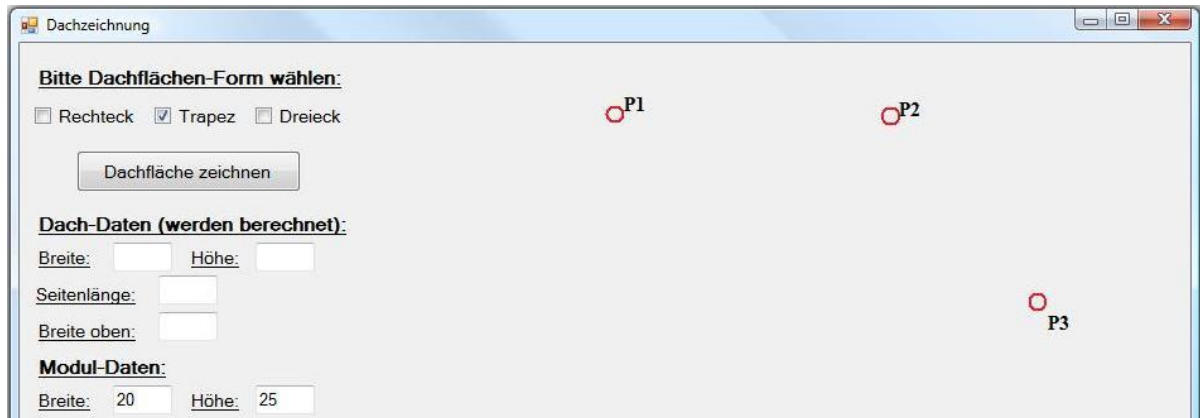


Dachzeichnung

Bitte Dachflächen-Form wählen:

☐ Rechteck ☒ Trapez ☐ Dreieck

Wird eine trapezförmige Dachfläche gewählt, müssen in einem nächsten Schritt drei Eckpunkte dieser Fläche auf der Zeichenoberfläche festgelegt werden.



Dachzeichnung

Bitte Dachflächen-Form wählen:

☐ Rechteck ☒ Trapez ☐ Dreieck

Dachfläche zeichnen

Dach-Daten (werden berechnet):

Breite: Höhe:

Seitenlänge:

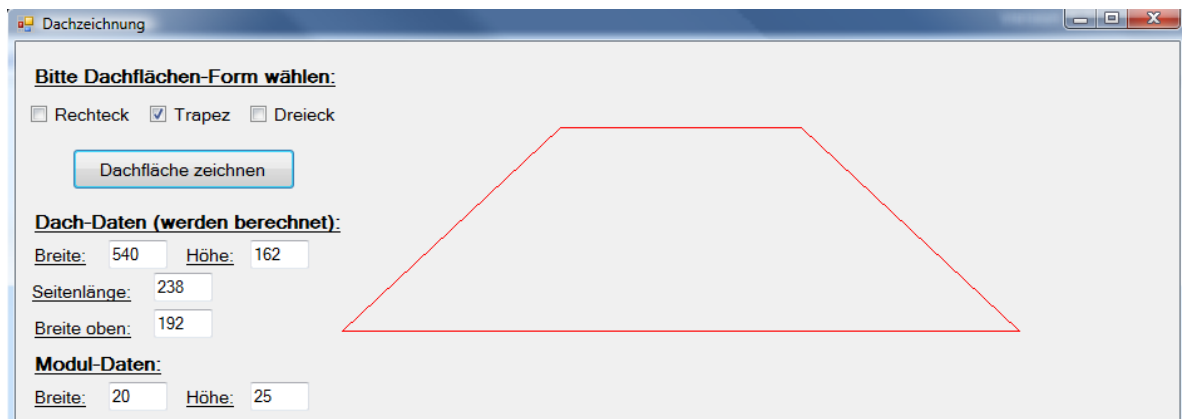
Breite oben:

Modul-Daten:

Breite: 20 Höhe: 25

Auf der verfügbaren Zeichenfläche sind für eine trapezförmige Dachfläche drei Punkte zu wählen, mit denen die Breite der schmalen Kante und die Position des seitlich ausgestellten Punktes der Dachfläche gekennzeichnet werden.

Über die Schaltfläche „*Dachfläche zeichnen*“ wird die Dachfläche angezeigt. Die berechneten Maße der Dachkanten werden berechnet und angezeigt.



Dachzeichnung

Bitte Dachflächen-Form wählen:

☐ Rechteck ☒ Trapez ☐ Dreieck

Dachfläche zeichnen

Dach-Daten (werden berechnet):

Breite: 540 Höhe: 162

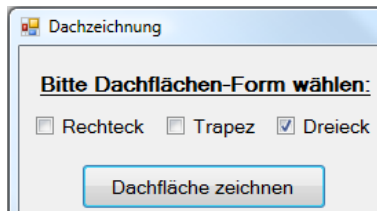
Seitenlänge: 238

Breite oben: 192

Modul-Daten:

Breite: 20 Höhe: 25

2.3 Dreieckige Dachfläche



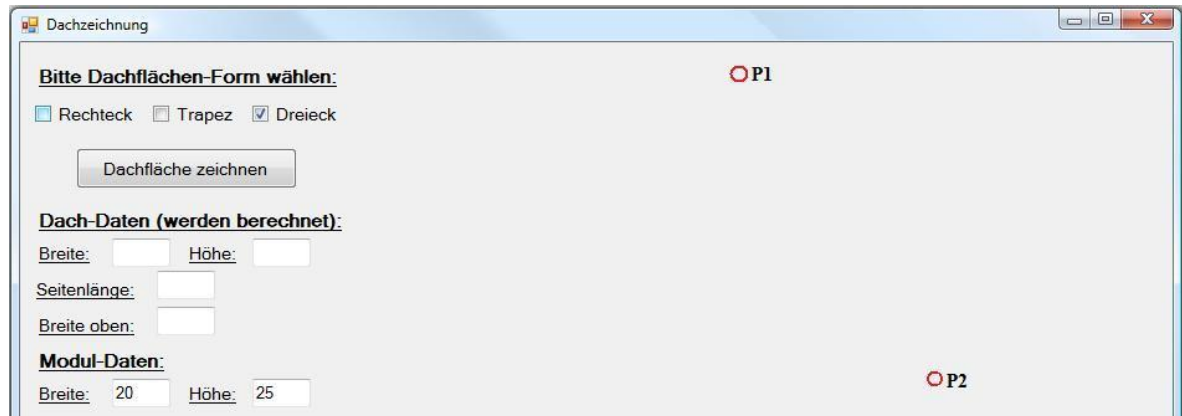
Dachzeichnung

Bitte Dachflächen-Form wählen:

☐ Rechteck ☐ Trapez ☒ Dreieck

Dachfläche zeichnen

Wird eine dreieckige Dachfläche gewählt, müssen in einem nächsten Schritt zwei Eckpunkte dieser Fläche auf der Zeichenoberfläche festgelegt werden.



Dachzeichnung

Bitte Dachflächen-Form wählen:

☐ Rechteck ☐ Trapez ☒ Dreieck

Dachfläche zeichnen

Dach-Daten (werden berechnet):

Breite: Höhe:

Seitenlänge:

Breite oben:

Modul-Daten:

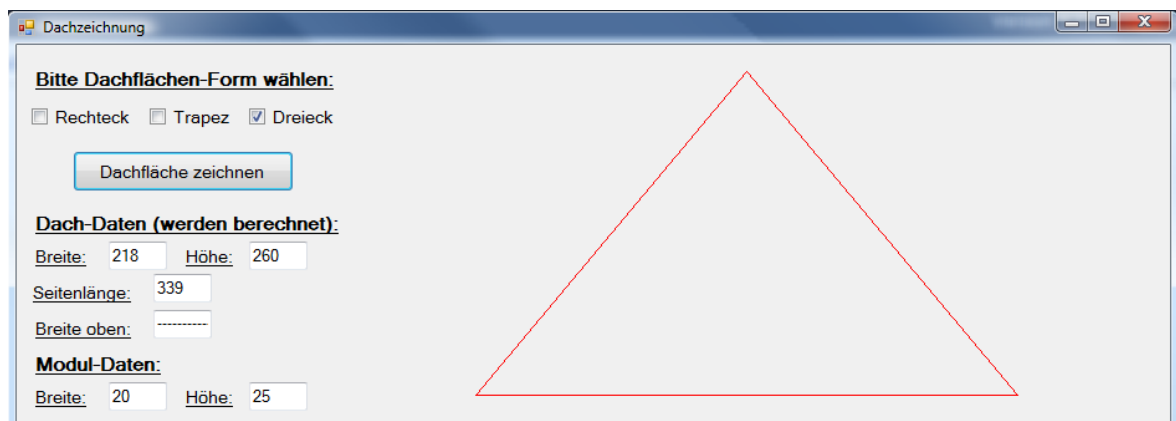
Breite: 20 Höhe: 25

P1

P2

Auf der verfügbaren Zeichenfläche sind für eine dreieckige Dachfläche zwei Punkte zu wählen, mit denen die Spitze und die Position eines seitlich ausgestellten Punktes der Dachfläche gekennzeichnet werden.

Über die Schaltfläche „*Dachfläche zeichnen*“ wird die Dachfläche angezeigt. Die berechneten Maße der Dachkanten werden berechnet und angezeigt.



Dachzeichnung

Bitte Dachflächen-Form wählen:

☐ Rechteck ☐ Trapez ☒ Dreieck

Dachfläche zeichnen

Dach-Daten (werden berechnet):

Breite: 218 Höhe: 260

Seitenlänge: 339

Breite oben: -----

Modul-Daten:

Breite: 20 Höhe: 25

3. Eingabe der Modul-Daten, Abstände zu den Dachkanten bzw. zwischen den Modulen, Startpunkt-Auswahl

Die Vorgehensweise zur Vollbelegung der Dachflächen mit Modulen ist für alle drei Dachflächen-Typen identisch. Aus diesem Grund wird an dieser Stelle exemplarisch die rechteckige Dachfläche ausgeführt.

Modul-Daten:
Breite: 20 Höhe: 25

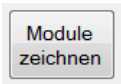
Abstände:
Abstand zur Dach-Kante:
horizontal: 10
vertikal: 10
Abstand zwischen den Modulen:
horizontal: 5
vertikal: 5

Startpunkt der Vollbelegung:
oben links

Bitte ändern Sie die vorgegebenen Werte in diesem Bereich Ihren Anforderungen entsprechend.

Sollten keine abweichenden Werte eingegeben werden, so werden die voreingestellten Default-Werte verwendet.

4. Automatische Vollbelegung

Mit Klick auf die Schaltfläche  wird das Ergebnis der automatischen Vollbelegung mit Photovoltaik-Modulen grafisch dargestellt.

Dachzeichnung

Bitte Dachflächen-Form wählen:
☒ Rechteck ☐ Trapez ☐ Dreieck
Dachfläche zeichnen

Dach-Daten (werden berechnet):
Breite: 542 Höhe: 344
Seitenlänge:
Breite oben:
Modul-Daten:
Breite: 20 Höhe: 25

Abstände:
Abstand zur Dach-Kante:
horizontal: 10
vertikal: 10
Abstand zwischen den Modulen:
horizontal: 5
vertikal: 5

Startpunkt der Vollbelegung:
oben links

Module zeichnen Module löschen

Anzahl der legbaren Module: 200

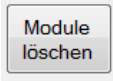
Alle Eingaben zurücksetzen

Alle Bemaßungen und Größen-Angaben sind in Pixel angegeben!

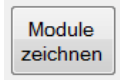
Zur optischen Kontrolle des Startpunktes wird der eingegebene Randabstand zu den Dachkanten mit einer grünen Linie dargestellt.

Die berechnete und gezeichnete Anzahl der Module wird angezeigt.

5. Module löschen

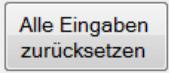
Mit der Schaltfläche  können die Module und die Zeichnung des Randabstandes entfernt werden. Die Dachfläche und die eingegebenen Werte für Abstände, Modul-Daten und Startpunkt-Auswahl bleiben erhalten.

Durch Wahl eines anderen Startpunktes kann gegebenenfalls die Anzahl der Module für die gewählte Dachfläche optimiert werden.

Die erneute Vollbelegung wird mit der Schaltfläche  gestartet.

6. Alles löschen und alle Eingaben zurücksetzen

Soll eine neue Dachfläche mit Modulen belegt werden, so muss zuerst die Schaltfläche

 angeklickt werden. Die Zeichnung wird gelöscht und auch die vom Anwender zuvor geänderten Daten werden auf die Ausgangswerte zurückgesetzt.



**Soll-Ist-Vergleich
und
Kosten-Nutzen-Analyse**

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 18.03.2013

1. Soll-Ist-Vergleich

Phase	Arbeitspaket	Soll-Zeit in h	Ist-Zeit in h
Definition & Analyse	IST-Analyse	1	1
	SOLL-Konzept	2	2
	Durchführbarkeitsanalyse	1	1
	Pflichtenheft	6	6
Planung	Arbeitspakete	1	1,5
	Planungsdokumente	5	4,5
	Qualitätsplan	1,5	1,5
Entwurf	Testfallkatalog	1,5	1,5
	DV-Entwurfsdokumente	8	9
	Entwurf Programmoberfläche	3	3
Implementierung	Erstellung der Programmoberfläche	1	1
	Programmierung	13	13,5
Test	Whitebox-Test	5	4
	Fehlerkorrektur	1	0,5
Abschluss	SOLL-IST-Vergleich	1,5	1,5
	Qualitätsbericht	1,5	1,5
	Abnahmeprotokoll inkl. Übergabe	0,5	0,5
Dokumentation	Entwicklerdokumentation	2,5	2,5
	Benutzer-Hilfe	1	1
	Prozessorientierter Projektbericht	10	10
Summe		67	67

In obiger Tabelle werden die tatsächlich benötigten Zeiten den ursprünglich geplanten Zeiten für die einzelnen Arbeitspakete gegenübergestellt. Arbeitsschritte, die schneller abgearbeitet werden konnten als geplant, sind grün (in Schwarz-Weiß-Ausdrucken hell-grau) hervorgehoben. Punkte, die mehr Zeit in Anspruch genommen haben als vorgesehen, sind hingegen rot (in Schwarz-Weiß-Ausdrucken dunkel-grau) unterlegt. Die im Verlauf der Projektdurchführung eingesparten Zeiten wurden an anderen Stellen benötigt, sodass die eingeplante Pufferzeit nicht benötigt wurde.

Nach Rücksprache mit Herrn H. Meier haben auch die vorgesehenen 4 Stunden für die Durchführung des [Blackbox-Testes](#) ausgereicht.

2. Kosten-Nutzen-Analyse

2.1 Kosten

Für die Bearbeitung des Projektes sind ursprünglich 67 Stunden angesetzt worden. Diese Gesamtstundenzahl wurde trotz einzelner Verzögerungen durch rascheres Vorankommen an

anderer Stelle aufgefangen, sodass die Gesamt-Stundenanzahl von 67 Stunden nicht überschritten wurde.

Die Pufferzeit musste nicht in Anspruch genommen werden. Somit sind auch keine weiteren Kosten bei dem durchgeführten Projekt entstanden.

Da sich auch beim nicht von der Antragstellerin durchgeführten [Blackbox-Test](#) kein zeitlicher Mehraufwand festgestellt worden ist, sind die veranschlagten Stunden auch hier eingehalten worden.

Lediglich der Zeitaufwand von Max Mustermann musste aufgestockt werden, da bei der Erstellung des Soll-Konzeptes, des Pflichtenheftes und bei der Abnahme des Projektes kurze Unterredungen notwendig waren. Daher wurde an dieser Stelle eine halbe Stunde mehr benötigt als ursprünglich vorgesehen.

Somit ergeben sich nach Abschluss des Projektes nachfolgende Kosten, die vom veranschlagten [Kostenplan](#) (siehe Anhang, S. 59 ff.) minimal abweichen:

Name	Zeitaufwand in Stunden	Stundensatz	Kosten
Elvira Musterfrau	67	89,00 €	5.963,00 €
F. Schulze	1	89,00 €	89,00 €
Max Mustermann	1	89,00 €	89,00 €
H. Meier	4	89,00 €	356,00 €
Gesamt			6.497,00 €

2.2 Nutzen

Der Nutzen, der durch die Entwicklung des Vollbelegungs-Algorithmus entsteht, ist monetär nicht messbar. Es entsteht zwar ein Zeitersparnis für den Anwender, da die Vollbelegung automatisch erfolgt, allerdings ist dieser Nutzen für die Anwendung nicht allein ausschlaggebend. Zudem nutzt der Algorithmus dem Endanwender, da ihm diese zusätzliche Funktion, eine schnelle, zuverlässige und vor allen Dingen auch optimale Belegung von verschiedenen Dachflächen ermöglicht.

Wichtiger für die Firma Hottgenroth GmbH & Co. KG ist die Kundenbindung, was durch die Weiterentwicklung von Funktionen und die Optimierung der Benutzerfreundlichkeit erreicht wird. Denn nur ein zufriedener Kunde bleibt einem Produkt oder auch einer Firma dauerhaft treu.



Qualitätsbericht

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 19.03.2013

1. Projektqualität

Das vorliegende Projekt wurde anhand des erweiterten Wasserfallmodells, das als Vorgehensmodell favorisiert wurde, durchgeführt. Dadurch wurde die Kontrolle eines strukturierten Ablaufes gewährleistet.

Anhand des erstellten Zeit- und Meilensteinplanes konnten die Abschnitte des Projektes begleitend kontrolliert werden. Reviews, die das Erreichen eines Meilensteins begleiteten, wurden durchgeführt. Zeitliche Verzögerungen beziehungsweise zeitliche Gewinne fanden nicht in den gleichen Phasen statt, in denen für ein Arbeitspaket länger oder kürzer gebraucht wurde. Über das gesamte Projekt hinweg betrachtet, gleichen sich die Differenzen jedoch aus, sodass der vierte Meilenstein und das Ende des Projektes planmäßig erreicht werden konnte.

Die veranschlagte Pufferzeit musste nicht in Anspruch genommen werden. Somit entstanden auch keine weiteren Kosten für den Auftraggeber.

2. Produktqualität

Der Fehler, der im Verlauf des [Whitebox-Testes](#) gefunden wurde, konnte durch einen Rücksprung in die vorangehende Implementierungsphase behoben werden. Trotz des Rücksprungs musste die vorhandene Pufferzeit nicht in Anspruch genommen werden. Durch eine erneute Testrunde inklusive [Blackbox-Test](#) konnte die Korrektheit und die Funktionstüchtigkeit des Produktes sichergestellt werden.

Das Kommentieren und Einrücken des Quellcodes anhand der firmeninternen Programmierrichtlinien sowie das Einhalten der vorhandenen Namenskonventionen tragen zur besseren Wartbarkeit und der geforderten Erweiterbarkeit bei. Die Kontrolle der Softwaregestaltung durch den Qualitätsmanagementbeauftragten entfiel bei diesem Projekt, da ein [Prototyp](#) erstellt wurde und keine Endanwendung für den Benutzer.

3. Fazit

Da die Ziele des Qualitätsplans eingehalten werden konnten, ist festzustellen, dass dieser vollständig erfüllt wurde.

Unter Umständen ist die Zeitplanung bei zukünftigen Projekten minimal anzupassen, sofern einige Arbeitspakete regelmäßig schneller abgearbeitet werden können oder generell mehr Zeit benötigen (z.B. erstellen von UML-Diagrammen). Ansonsten war ein reibungsloser Ablauf gegeben, wodurch das Projekt zu einem guten Abschluss gebracht werden konnte.



Abnahmeprotokoll

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 28.03.2013

1. Abnahme-Voraussetzungen

Der Algorithmus zur Vollbelegung von Dachflächen mit Photovoltaik-Modulen wurde am 28.03.2013 an den Auftraggeber, Herrn Max Mustermann, übergeben.

Die einwandfreie Funktion der Berechnungen sowie die Vollständigkeit der verfassten Dokumente wurden vorausgesetzt.

2. Abnahme

Die einwandfreie Funktionstüchtigkeit und die Vollständigkeit der übergebenen Protokolle werden hiermit von Auftraggeber (Max Mustermann) und Auftragnehmer (Elvira Musterfrau) bestätigt.

Köln, den 28.03.2013

Max Mustermann
(Auftraggeber)

Elvira Musterfrau
(Auftragnehmer)



Glossar

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 27.03.2013

1. Algorithmus

Ein Algorithmus ist eine aus endlich vielen Schritten bestehende, eindeutige und ausführbare Handlungsvorschrift zur Lösung eines Problems oder einer Klasse von Problemen. (Quelle: <http://de.wikipedia.org/wiki/Algorithmus> ; abgerufen am 27.03.2013)

2. Aufständering

Eine Aufständering ist die Montage von Photovoltaik-Modulen auf Gestellen mit optimalem Neigungswinkel. Durch sie soll der Ertrag der einzelnen Module optimiert werden. So können Sonnenwinkel berechnet werden und die Module so aufgebaut werden, dass der größtmögliche Nutzen bei der Gewinnung von Energie aus Sonne erzielt wird.

3. Blackbox-Test

Programm-Test ohne Kenntnis des Quellcodes anhand eines Testfallkataloges. Wird von einer dritten Person durchgeführt.

4. CAD

CAD = computer-aided design (deutsch: rechnerunterstütztes Konstruieren). Mit Hilfe von Software werden grafische Objekte (Produkte / Gegenstände) virtuell konstruiert.

5. Default-Werte

Festgelegte Standard-Werte, die übernommen werden, wenn keine benutzerspezifischen Eingaben erfolgt sind.

6. Energieberater

Anwendung zur energetischen Planung und Bewertung von Wohn- und Nichtwohn-Gebäuden nach aktuellen Verordnungen und Normen (z.B. EnEV; DIN V 4108-6; DIN V 18599; DIN V 4701-10). [in Anlehnung an den „Gesamtkatalog 2013“ der Firma Hottgenroth Software GmbH & Co. KG]

7. Einfaches Wasserfallmodell

Lineares, nicht iteratives Vorgehensmodell. Die Phasen dieses Vorgehensmodells werden chronologisch durchlaufen und abgearbeitet. Jedes Phasenergebnis ist eine bindende Voraussetzung für den Beginn der nächsten Phase.

8. Erweitertes Wasserfallmodell

Grundlage ist das lineare Durchlaufen der Phasen des einfachen Wasserfallmodells. Allerdings ist ein schrittweises Zurückspringen in die vorherige Phase möglich, wenn festgestellt wird, dass Unwägbarkeiten durch abweichende Ziele oder unvorhersehbare personelle oder technische

Ausfälle aufgetreten sind. Auch hier ist ein festgeschriebenes Phasenergebnis bindend, bevor in die nächste Phase gewechselt werden kann.

9. Extreme Programming

Vorgehensmodell, das durch fortlaufende Iteration und den Einsatz mehrerer Einzelmethoden gekennzeichnet ist. Es handelt sich um ein strukturiertes Vorgehensmodell. Dabei werden Kunden-Anforderungen in kleinen Schritten umgesetzt.

10. Gantt-Diagramm

Instrument des Projektmanagements, das die zeitliche Abfolge von Aktivitäten in Form von Balken auf einer Zeitachse grafisch darstellt.

11. GUI

Graphical-User-Interface (deutsch: Grafische Benutzeroberfläche); bietet dem Benutzer eines Computers die Möglichkeit, ein Programm mit Maus und Tastatur zu bedienen.

12. HS Code-Dokumentationsrichtlinien

Die HS Code-Dokumentationsrichtlinien enthalten Anweisungen darüber, wie die einzelnen Code-Abschnitte kommentiert werden sollen.

13. HS-GUI-Style-Guide

Der HS-GUI-Style-Guide enthält Vorgaben für das äußere Erscheinungsbild (Look) und die Bedienmöglichkeiten (Feel) der hauseigenen Software. So sind hier unter anderem Vereinbarungen bezüglich eines allgemein üblichen Windows-Look&Feel-Layout und auch Bedienkomponenten definiert, die spezifisch für Hottgenroth Software sind.

14. HS-Programmierrichtlinien C#

Die HS-Programmierrichtlinien C# enthalten Vorgaben zur Generierung von Quellcode in der Entwicklungsumgebung. Die Vorgaben betreffen unter anderem die Klassen-, Methoden-, Variablen- und Konstanten-Benennung.

15. Inverter

Ein Inverter ist ein Solarwechselrichter und Teil einer jeden Solaranlage. Er wandelt Gleichstrom in Wechselstrom um. Dieser kann dann verbraucht, oder in das öffentliche Netz eingespeist werden.

16. Netzplan

Ein Netzplan ist eine grafische Darstellung eines Projektverlaufs. Logische Abhängigkeiten zwischen den im Projekt verankerten Arbeitspaketen und deren zeitliche Positionierung und Dauer werden hieraus ersichtlich.

17. Nominalphrasierung

Aus einer Erzählung werden die auftauchenden Nomen (Hauptwörter z.T. mit Artikel) herausgezogen. Diese bilden im Fall des Software-Engineerings die Grundlage zu den Klassen eines Klassendiagrammes.

18. Photovoltaik-Module / PV-Module

Das Licht der Sonne wird durch Solar- / Photovoltaik-Module direkt in elektrischen Strom umgewandelt.

19. Prototyp

19.1 generell:

Ein Prototyp ist ein generelles erstes Entwurfsmuster (z.B. für Aussehen, Funktionen oder Geräte), der schnell zu ersten Ergebnissen führt. Somit ist ein frühzeitiges Feedback bezüglich der Eignung eines Lösungsvorschlages möglich.

19.2 im Projekt:

Im Fall des Projektes ist der Prototyp ein Entwurfsmuster für die Benutzeroberfläche. Die Ausgabe der durch den Algorithmus berechneten Module erfolgt an der Benutzeroberfläche. Die endgültige Version nach Integration des Algorithmus in die Anwendung „Grafische Erfassung“ (CAD) wird anders aussehen.

20. Prototyping

Prototyping ist ein Vorgehensmodell, das schnell zu sichtbaren Ergebnissen führt und frühzeitig ein Feedback ermöglicht. Prototypen sind die Basis für eine bessere Kommunikation mit dem Kunden über die Benutzerführung und Funktionsweise. Das bietet eine bessere Veranschaulichung, statt mit abstrakten Modellen zu arbeiten.

21. Review

Manuelle Überprüfung von Arbeitsergebnissen in der Software-Entwicklung. Die erzielten Ergebnisse werden zusammen mit einer anderen Person überprüft. Ein Review ist eine statische Testmethode und wird zu den analytischen Qualitätssicherungsmaßnahmen gezählt.

22.Sperrfläche

Bereiche, die für festgelegte Aktionen verboten sind. Bei der Dachvollbelegung sind Bereiche, die Dachfenster, Dachgauben oder Schornsteine enthalten, für das Positionieren von Photovoltaik-Modulen gesperrt.

23.Spiralmodell

Dieses Vorgehensmodell betrachtet den Entwicklungsprozess als iterativen Prozess. Es handelt sich um eine Weiterentwicklung des "[einfachen Wasserfallmodells](#)". Hier wird jede einzelne Phase mehrfach (spiralförmig) durchlaufen. Mit dieser Methode wird versucht, das Scheitern großer Software-Projekte zu verhindern.

24.TGA-Heizung

Software zum Auslegen und Berechnen von Heizungsanlagen (TGA: Technische Gebäude-Ausrüstung).[in Anlehnung an den „Gesamtkatalog 2013“ der Firma Hottgenroth Software GmbH & Co. KG]

25.U-Wert

Wärmedurchgangskoeffizient (früher "k-Wert"); Maß für den Wärmedurchgang durch ein Bauteil. Angabe in $W/(m^2K)$. Der U-Wert beschreibt die Wärmemenge, die bei einem Grad Temperaturunterschied durch einen Quadratmeter des Bauteils entweicht. Je kleiner der U-Wert ist, desto weniger Wärme wird durch ein Bauteil geleitet.

26.Vorgehensmodell

Ein auf die Software-Entwicklung angepasstes Verfahren für den Bereich der Anwendungsentwicklung. Hiermit soll der Prozess der Software-Entwicklung übersichtlicher gestaltet und auch beherrschbar werden.

27.Wartbarkeit

Kriterium der Software-Entwicklung. Die Wartbarkeit zeigt an, mit welchem Erfolg, mit welcher Qualität und mit welchem Programmieraufwand Änderungen an einem Programm / Projekt durchgeführt werden können.

28.Whitebox-Test

Programm-Test anhand eines Testfallkataloges mit Kenntnis des Quellcodes und der inneren Funktionsweise des Programms. Wird meist von der entwickelnden Person selber durchgeführt.



Persönliche Erklärung

Projekt:

**„Algorithmus zur automatischen Vollbelegung von Dächern mit
Photovoltaik-Modulen“**

Autor: Elvira Musterfrau
Datum: 27.03.2013

1. Erklärung zur Projektarbeit und deren Dokumentation

Vor- / Name des Prüfungsteilnehmers	Elvira Musterfrau
Ausbildungsberuf	Fachinformatikerin, Bereich Anwendungsentwicklung
Ausbildungsbetrieb	Hottgenroth Software GmbH & Co. KG
Thema der Projektarbeit	Entwicklung eines Algorithmus zur automatischen Vollbelegung von Dächern mit Photovoltaik-Modulen
Termin der Abschlussprüfung	Sommer 2013

2. Persönliche Erklärung der Prüfungsteilnehmerin

Ich versichere durch meine Unterschrift, dass ich dieses Projekt und alle dazugehörigen Dokumente selbstständig und ohne fremde Hilfe angefertigt habe.

Nicht von mir entwickelte und verfasste Dokumente habe ich als solche entsprechend kenntlich gemacht.

Köln, den _____

Elvira Musterfrau
(Auszubildende)

Einverständniserklärung des Ausbildungsbetriebes:

Köln, den _____

F. Schulze
(Projektverantwortlicher)

Köln, den _____

Anna Ausbilder
(Ausbildungsleiterin)