Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Zentrum für GeoInformatik (Z_GIS)
der Paris Lodron Universität Salzburg

zum Thema

Spatial Decision Support System (SDSS) für die Windkraft-Planung

vorgelegt von

Dipl. Geogr. Matthias Jochem

U102589, UNIGIS MSc Jahrgang 2012

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc(GIS)”

Gutachter:
Ao. Univ. Prof. Dr. Josef Strobl

Regensburg, 19.08.2014

I. Eigenständigkeitserklärung

Ich versichere diese Master Thesis ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben und, dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Alle Ausführungen der Arbeit, die wörtlich oder sinngemäß übernommen wurden, sind entsprechend gekennzeichnet.

Regensburg, 19.08.2014 _____

Unterschrift

II. Kurzfassung

Für das Auffinden und Bewerten potenzieller Flächen für Windenergieanlagen (WEAs) kommen zumeist multikriterielle Bewertungsverfahren in Verbindung mit GI-Systemen zum Einsatz. Häufig werden diese ‚gewöhnlichen' GIS-basierten Analysen in der Literatur unter dem Begriff *Spatial Decision Support System* (SDSS) gehandelt. Ein SDSS ist jedoch nicht mit GI-Systemen oder GIS-basierten Analysen gleichzusetzen, wenngleich hier enge Verbindungen bestehen. Kennzeichen eines SDSS ist u.a. dessen Ausrichtung auf ein spezifisches Entscheidungsproblem. Zudem bietet es dem Nutzer die Möglichkeit die Datenprozessierung zu verändern.

Mittels der Skriptsprache *Python* wird daher innerhalb von *ArcGIS 10.1* ein ‚richtiges' SDSS entwickelt, das den Prozess der Potenzialflächenanalyse für WEAs automatisiert. Die Bewertung der Flächen erfolgt dabei mittels *Weighted Linear Combination* (WLC). Darüber hinaus kann der Nutzer die Untersuchungsregion selbst bestimmen sowie Abstands- und Ausschlussflächen definieren. Dadurch sind Flächenanalysen auch für mehrere Gebiete mit unterschiedlichen regionalen Rahmenbedingungen in kurzer Zeit möglich. Das SDSS wird letztlich anhand einer ausgewählten Untersuchungsregion in Thüringen getestet und die Ergebnisse erfolgreich auf Plausibilität geprüft.

Schlagwörter: Flächenanalyse, multikriterielle Entscheidungsanalyse, *Python*, SDSS, *Spatial Decision Support System*, *Weighted Linear Combination*, Windenergie, Windpark, WLC

III. Abstract

The selection and suitability assessment of potential sites for wind farms typically employs multiple criteria evaluation models combined with GIS. The literature often describes these 'common' GIS-based analyses as Spatial Decision Support Systems (SDSS). GIS or GIS-based analysis and SDSS, however, are not identical although they share many characteristics. A SDSS is designed for a specific decision problem and offers the user the possibility to change the processing of input data.

This study develops a 'true' SDSS within ArcGIS 10.1 that automates the site selection and evaluation process using the scripting language Python. A Weighted Linear Combination (WLC) is used to evaluate the sites and users are able to select the region under study and define excluded areas. This makes it possible to assess site suitability of several areas with distinct regional characteristics and regulatory frameworks within a short timeframe. The study applies the developed SDSS to a selected case region in Thuringia and successfully confirms the SDSS results in a plausibility analysis.

Keywords: multi criteria evaluation, Python, SDSS, site suitability, Spatial Decision Support System, Weighted Linear Combination, windfarm, windpower, WLC

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AHP	Analytical Hierarchy Process
ASCII	American Standard Code for Information Interchange
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
BRD	Bundesrepublik Deutschland
DWD	Deutscher Wetterdienst
FFH-Gebiete	Flora-Fauna-Habitat-Gebiete
GIS	Geographisches Informationssystem
GUI	Graphical User Interface
kg/m^3	Kilogramm pro Kubikmeter
kW	Kilowatt
kWh/a	Kilowattstunden pro Jahr
MC-SDSS	Mulicriteria Spatial Decision Support System
MKE	multikriterielle Entscheidungsanalyse
MKRE	multikriterielle räumliche Entscheidungsanalyse
MW	Megawatt
MWh/a	Megawattstunden pro Jahr
OSM	Open Street Map
PDF	Portable Document Format
pbf	Protocolbuffer Binary Format
SDSS	Spatial Decision Support System(s)
shp	Shape
SRTM	Shuttle Radar Topographic Mission
TK	Topographische Karte
UML	Unified Modelling Language
VGI	Volunteered Geographic Information
WEA	Windenergieanlage
WLC	Weighted Linear Combination
WMS	Web Map Service

1 Einleitung

1.1 Hintergrund und Ziel der Arbeit

Die Planung und Projektierung von Windparks ist mit vielen Unsicherheiten verbunden. Eine der größten Unsicherheiten stellt hierbei das Auffinden jener Gebiete dar, welche sowohl technisch als auch ökonomisch für die Entwicklung von Windenergie-Projekten geeignet sind, zumal sich diese von Jahr zu Jahr weiter reduzieren. Das Ausfindigmachen neuer und rentabler Flächen wird somit zu einem entscheidenden Faktor bei der Windpark-Planung (Grassi et al. 2012). Für das Selektieren und Bewerten dieser Areale ist es notwendig, eine Vielzahl von Kriterien in Betracht zu ziehen (Gorsevski et al. 2013). Die Standortplanung und -bewertung für Windenergieanlagen (WEAs) kann somit als multikriterieller Prozess der Entscheidungsfindung angesehen werden (Mari et al. 2011). *Spatial Decision Support Systems* (SDSS) sollen diesen Entscheidungsprozess unterstützen, indem sie multikriterielle Bewertungsverfahren, zur Selektion und Gewichtung von Entscheidungskriterien und Alternativen, mit GI-Systemen kombinieren (Malczewski 2006). Zahlreiche Studien haben sich bisher auf unterschiedliche Weise und unter Berücksichtigung verschiedenster Kriterien mit dieser Thematik auseinandergesetzt (Al-Yahyai et al. 2012, Aydin et al. 2010, Baban & Parry 2001, Bennui et al. 2007, Hansen 2005, Janke 2010, Ouammi et al. 2012, Ramirez-Rosado 2008, Rodman & Meentemeyer 2006, Tegou et al. 2010, Voivontas et al. 1998). Gemein ist diesen Studien, dass der Entscheidungsprozess bzw. die Datenverarbeitung nicht automatisiert ist, die Selektions- und Bewertungskriterien starr sind und sich die Flächenanalyse nicht auf benutzerdefinierte geographische Regionen begrenzen lässt. Eine Ausnahme stellt hier zum Teil die Arbeit von van Haaren & Fthenakis (2011) dar. Sie entwickelten innerhalb von *ArcGIS-Desktop 9.3.1* ein Tool, das dem Nutzer die individuelle Eingabe von Ausschlussflächen sowie das Definieren des Untersuchungsgebiets ermöglicht. Gerade in Ländern mit unterschiedlichen regionsspezifischen gesetzlichen Anforderungen und naturräumlichen Gegebenheiten ist es notwendig, sowohl das zu untersuchende Gebiet als auch Ausschlussflächen sowie Bewertungskriterien und deren Gewichtung flexibel definieren zu können. Dadurch kann man unmittelbar zu einer für die jeweilige Region passenden Flächenselektion und

-bewertung gelangen, welche die regionsspezifischen Eigenheiten adäquat berücksichtigt. Ziel dieser Arbeit ist es daher, den Prozess der Flächenselektion und -bewertung durch die prototypische Entwicklung eines SDSS, in Form eines GIS-basierten Tools, das die beschriebenen Anforderungen erfüllt, zu automatisieren. Auf mögliche politische, gesetzliche oder technische Anforderungen kann somit schnell reagiert werden. Ergebnis dieses Prozesses stellt eine Karte dar, welche die selektierten Gebiete sowie deren ökonomisch-technischen Eignungsgrad für die Windkraftnutzung wiederspiegelt. Das SDSS wird exemplarisch für ein Gebiet in Thüringen konzipiert. Wenngleich im Zuge der Plausibilitätsprüfung eine grobe Flächenpotenzialanalyse in der Untersuchungsregion durchgeführt wird, zielt die Arbeit nicht darauf ab das juristische, ökonomische oder technische Potenzial für die Windkraftnutzung in dieser Region zu analysieren oder zu bewerten. Bei der vorliegenden Arbeit geht es lediglich um die Konzeption und Umsetzung des SDSS. Grund für die Wahl des Projektgebiets war das Vorhandensein eines für die Tool-Entwicklung wichtigen und kostenpflichtigen Datensatzes in dieser Region.

1.2 Methodik und Aufbau der Arbeit

Wie der Name schon sagt, unterstützen SDSS den Entscheidungsprozess. Zu Beginn der Arbeit wird daher auf die theoretischen Grundlagen der Entscheidungsfindung eingegangen. Hierzu gehört auch die multikriterielle Bewertung, als integraler Bestandteil des zu entwickelnden Systems. Es werden deshalb ebenfalls die Grundlagen und Funktionsweisen ausgewählter Bewertungsverfahren erläutert. Zudem erfolgt eine Definition des Begriffs SDSS. Dies ist vor allem für ein detaillierteres Verständnis der Thematik als auch für die Überprüfung einiger in Kapitel 1.4 aufgestellter Hypothesen von Bedeutung. In Kapitel 3 wird schließlich auf den Prozess der Standortanalyse für Windkraftanlagen eingegangen. Dieser Prozess bildet die Grundlage für die Konzeption des SDSS. Kapitel 4 setzt sich mit der Berechnung des Windenergiepotenzials auseinander. Grund hierfür ist die außerordentliche ökonomische Bedeutung dieses Faktors im Planungsprozess (van Haaren & Fthenakis 2011). Die verwendeten Datenquellen sowie die Datenaufbereitung werden in Kapitel 5 beschrieben. Die Datenaufbereitung basiert zum Teil auf theoretischen Grundlagen der Kapitel 2 bis 4. Selbiges gilt für die in Kapitel 6 folgende Konzeption und Umsetzung des SDSS. Die Umsetzung erfolgt als GIS-Tool innerhalb von *ArcGIS 10.1* mittels der Skriptsprache *Python* über das *Arcpy-Site-Paket*. Hier kommen vor allem Werkzeuge des *Spatial Analyst* zum Einsatz. Kapitel 7 beinhaltet schließlich die Ergebnisanalyse. Hierzu gehört zum einen die Verifizierung bzw. Falsifizierung der Hypothesen, zum anderen die Prüfung des Outputs auf Plausibilität. Die Arbeit schließt mit einer kritischen Zusammenfassung sowie einem Ausblick auf mögliche zukünftige Forschungsarbeiten ab.

1.3 Projektgebiet

Das SDSS wird für ein ca. 50 km x 50 km großes Gebiet in Thüringen entwickelt. In dessen Zentrum liegt die Landeshauptstadt Erfurt. Im Nordosten wird es durch die Landesgrenze nach Sachsen-Anhalt begrenzt. Im Südwesten befinden sich Ausläufer des Thüringer Waldes. Ein Blick in *Google Earth* zeigt, dass der Untersuchungsraum im Norden flach und von landwirtschaftlichen Nutzflächen geprägt ist. Im Süden dominieren teils landwirtschaftliche Flächen, teils Waldgebiete. Innerhalb dieses Projektgebietes wird ein zweites Untersuchungsgebiet definiert: Da das SDSS die flexible Auswahl des Untersuchungsraums ermöglichen soll, sind zumindest zwei Untersuchungsräume zu definieren, um die Ergebnisse auf Plausibilität prüfen zu können. Das zweite Untersuchungsgebiet wird durch die Außengrenzen mehrerer Gemeinden definiert. Die Daten hierfür stammen vom Bundesamt für Kartographie und Geodäsie[1].

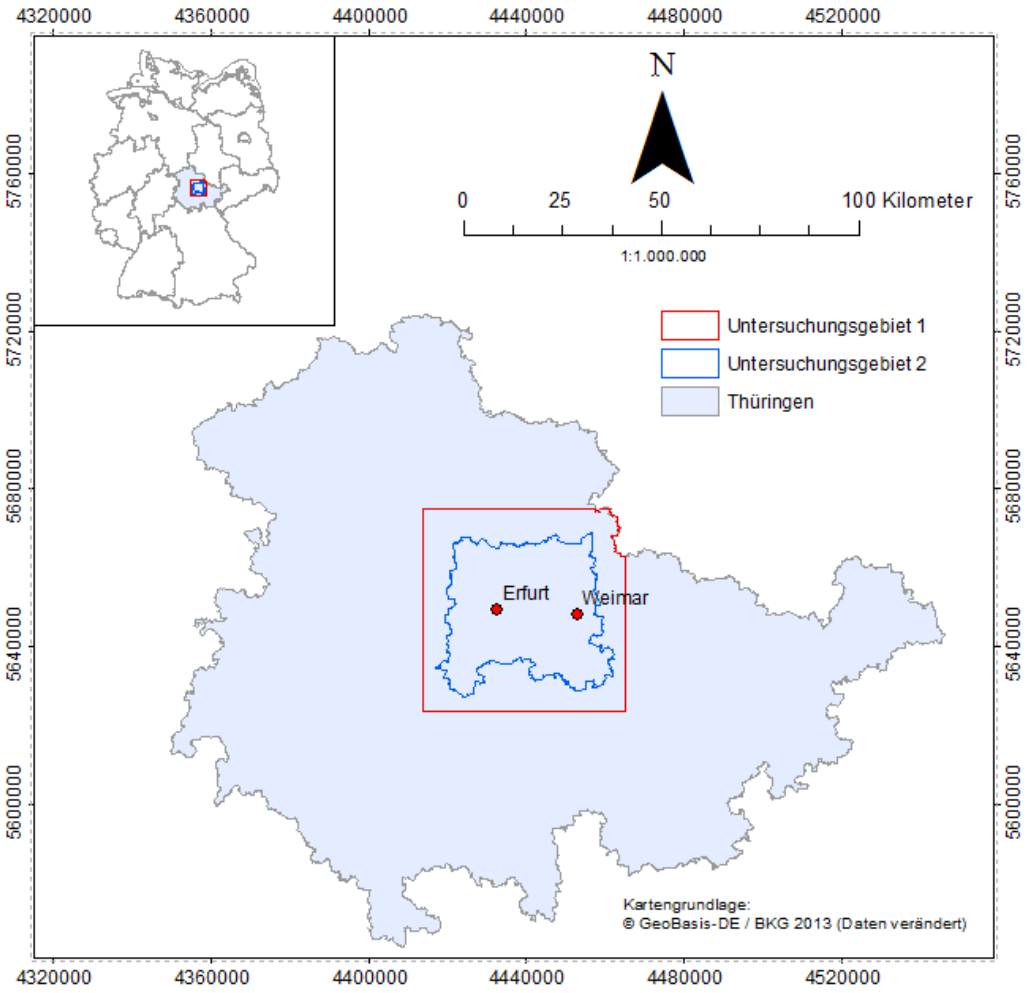


Abbildung 1: Untersuchungsgebiete

[1] www.bkg.bund.de (Stand: 15.12.2013)

1.4 Hypothesen

Die Gültigkeit nachfolgender Hypothesen wird im Zuge dieser Arbeit überprüft. Zugleich stellen diese Hypothesen Anforderungen an das zu entwickelnde SDSS, die teils noch spezifiziert werden müssen.

Hypothese A:

Das SDSS erfüllt alle Kriterien, die sich aus der Definition in Kapitel 2 ergeben.

Hypothese B:

Der Anwender kann das zu untersuchende Gebiet flexibel festlegen, Ausschluss- und Abstandsflächen selbst definieren sowie Eignungskriterien individuell auswählen und gewichten.

Hypothese C:

Das gesamte SDSS lässt sich als *Script-Tool* innerhalb von *ArcGIS 10.1* mittels der Skriptsprache *Python* über das *Arcpy-Site-Paket* umsetzen.

2 Grundlagen der Entscheidungsfindung

2.1 Entscheidungsprozess

Bei einer Entscheidung handelt es sich um eine Auswahl, die zwischen zwei oder mehr Alternativen getroffen wird. Simon (1960) geht davon aus, dass es sich dabei um einen Prozess handelt, der sich in drei Phasen unterteilen lässt: *Intelligence-*, *Design-* und *Choice-Phase.* Die *Intelligence-Phase* besteht aus der Problemdefinition bzw. der Definition des zu erreichenden Ziels sowie der Auswahl von Kriterien, die für die Problemlösung relevant sind (Sugumaran & Degroote 2011, Malczewski 1999). In der *Design-Phase* kommt es zur Datenauswahl und -verarbeitung. Diese Phase beinhaltet typischerweise die Entwicklung einer formalen Struktur, welche den Entscheidungsträger bei der Auswahl von Alternativen unterstützt (Drobne & Lisec 2009). In der *Choice-Phase* wird die eigentliche Entscheidung getroffen. Hier kommen spezifische Entscheidungsregeln zum Bewerten und Einschätzen der Alternativen zum Einsatz (Karnatak et al. 2007). Da dem Entscheidungsträger häufig nicht alle notwendigen Informationen für die Problemlösung bekannt sind oder nicht genau definiert werden können, sind Entscheidungen meist mit Unsicherheiten behaftet (Malczewski 1999). Eine Entscheidung wird unter Bedingungen der Sicherheit getroffen, falls der Entscheidungsträger über ein vollständiges Wissen der Entscheidungssituation verfügt (Malczewski 2006). Folglich wird eine Entscheidung umso eher unter der Bedingung der Unsicherheit getroffen, je geringer dieses Wissen ausfällt.

Letztlich muss erwähnt werden, dass die drei erläuterten Phasen keineswegs linear ablaufen müssen, sondern häufig, aufgrund neuer Erkenntnisse oder Erfahrungen, zu vorherigen Phasen zurückgesprungen wird (Sugumaran & Degroote 2011).

Da sich die vorliegende Arbeit mit räumlichen Entscheidungen auseinandersetzt, wird im Folgenden auf die räumlichen Aspekte der Entscheidungsfindung eingegangen. Der oben beschriebene Entscheidungsprozess wird hierzu weiterentwickelt und konkretisiert.

2.2 Räumliche Entscheidungsfindung

Häufig erfordern räumliche Entscheidungsprobleme, dass Alternativen durch die Betrachtung mehrerer Kriterien bewertet werden müssen (Drobne & Lisec 2009). Räumliche Entscheidungsprobleme besitzen somit einen multikriteriellen Charakter, bzw. sind nur unter Zuhilfenahme von Methoden der multikriteriellen räumlichen Entscheidungsanalyse (MKRE) adäquat zu lösen. MKREs können als Prozess angesehen werden, der multidimensionale geographische Daten und Informationen kombiniert und in eine eindimensionale Bewertung der Alternativen umwandelt (Malczewski 1999). Für einen Überblick über die vielfältigen Methoden im Bereich der MKREs sei auf die Arbeit von Sugumaran & Degroote (2011) verwiesen. Bei der hier vorliegenden Arbeit werden zwei dieser Methoden, die bei der Umsetzung des SDSS Anwendung finden, genauer beleuchtet (Kapitel 2.3).

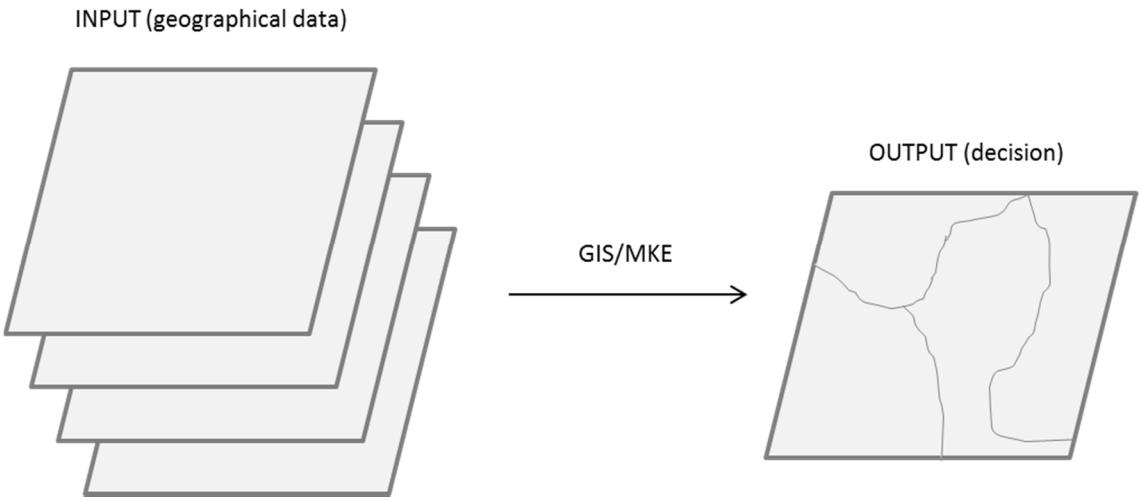


Abbildung 2: Datentransformation durch MKRE (Quelle: Malczewski 1999)

Im Allgemeinen zielen multikriterielle Entscheidungsanalysen (MKE) darauf ab, Alternativen auf Basis sich widersprechender Kriterien zu analysieren (Malczewski 1999). Im räumlichen Kontext bedeutet dies, die Eignung von Flächen für eine bestimmte Nutzung, anhand der Gewichtung spezifischer und teils entgegengesetzter Kriterien, zu bewerten. Die Flächen stellen dabei die zur Auswahl stehenden Alternativen dar. Malczewski (1999) entwickelte einen theoretischen Rahmen, welcher die MKRE in das von Simon (1960) aufgestellte Phasenmodel integriert:

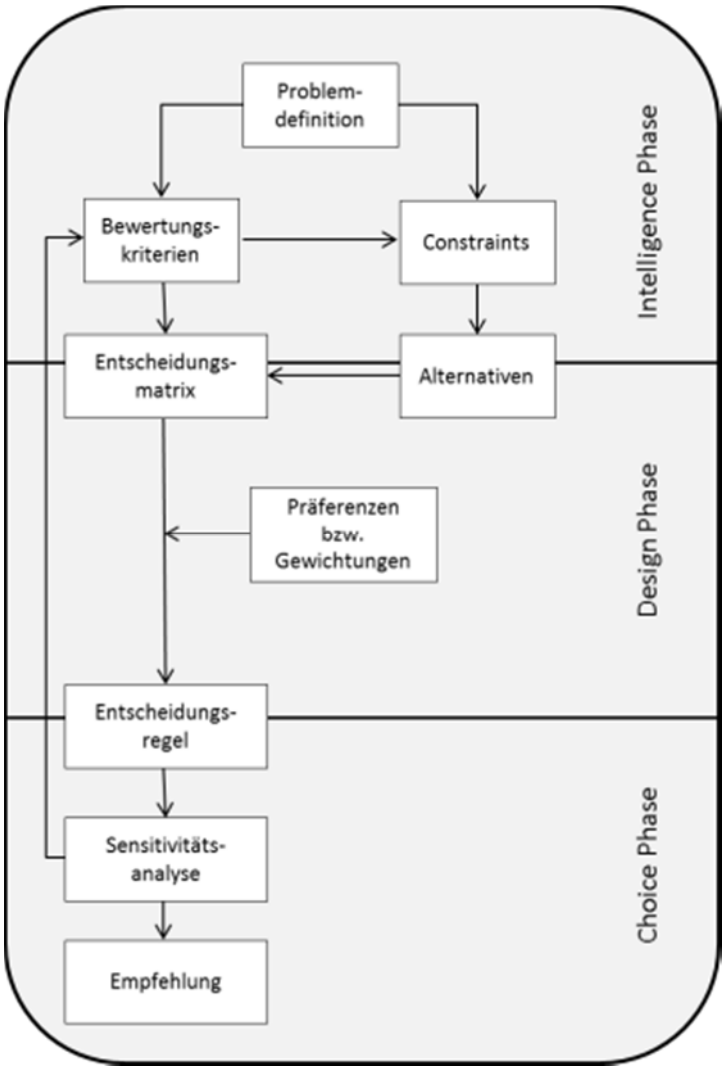


Abbildung 3: Theoretischer Rahmen der MKRE (Quelle: Malczewski 1999)

Nach der Definition des Problems und zu erreichenden Ziels erfolgt in einem zweiten Schritt die Auswahl aller für die Problemlösung bzw. Zielerfüllung notwendigen und zu bewertenden Kriterien. Zudem wird eine Werteskala definiert, die den Grad der Zielerfüllung wiederspiegelt. Die Werte selbst nennt Malczewski Attribute. Sie dienen der Bewertung der Alternativen. Anschließend wird jeder Alternative in Verbindung mit dem jeweiligen Kriterium eine Entscheidungsvariable zugeordnet. Diese definieren einen Entscheidungsraum. Jede Variable spiegelt dabei den Grad der Erfüllung eines Teilziels wieder, das durch das jeweilige Kriterium bestimmt wird. Der Entscheidungsraum lässt sich auch in Form einer Matrix veranschaulichen (Tab.: 1). Jede Alternative wird in Relation zum jeweiligen Teilziel gesetzt. Die Werte in den Zellen stellen dabei die Entscheidungsvariablen dar. Restriktionen, die den Entscheidungsraum begrenzen, werden als *Constraints* bezeichnet. Bezogen auf die Entscheidungsmatrix bedeutet

dies, dass die Anzahl der Zeilen reduziert wird. Flächen, die aufgrund natürlicher oder legislativer Faktoren für die Windkraftnutzung nicht verfügbar sind, stellen beispielsweise *Constraints* dar.

Tabelle 1: Entscheidungsmatrix (Quelle: Malczewski 1999, verändert)

	Teilziel 1	**Teilziel 2**	**Teilziel 3**	**…**	**Teilziel n**
Alternative 1	Variable 11	Variable 12	Variable 13	…	Variable 1n
Alternative 2	Variable 21	Variable 22	Variable 23	…	Variable 2n
Alternative 3	Variable 31	Variable 32	Variable 33	…	Variable 3n
…	…	…	…	…	…
Alternative m	Variable m1	Variable m2	Variable m3	…	Variable mn

Die eigentliche Datenverarbeitung beginnt durch das Zuordnen von Präferenzen bzw. Gewichtungen zu den Kriterien durch den Entscheidungsträger. Die Gewichtung der Kriterien und deren Aggregation erfolgt über festgesetzte Regeln (Kapitel 2.3), die die Bewertung der Alternativen bestimmen (Hocevar & Riedl 2003).

Durch eine anschließende Sensitivitätsanalyse erhält der Entscheidungsträger ein tieferes Verständnis der Problemstruktur. Sensitivitätsanalysen zielen darauf ab, den Einfluss der Kriterien und Gewichtungen auf das Endergebnis zu analysieren. Kriterien und Gewichtungen sind stets mit Unsicherheiten verbunden, welche die Unsicherheit des Endergebnisses determinieren. Durch Sensitivitätsanalysen kann das Endergebnis besser interpretiert und eine verlässlichere und sicherere Entscheidung getroffen oder Empfehlung abgegeben werden (Feizizadeh & Blaschke 2014, Lilburne & Tarantola 2009).

2.3 Methoden multikriterieller räumliche Entscheidungsanalysen

2.3.1 *Boolean Overlay*

Wie oben erwähnt wird bei MKREs ein multidimensionaler Input in einen eindimensionalen Output umgewandelt. In der hier vorliegenden Arbeit werden verschiedene Rasterdatensätze aggregiert. Die Pixel der Raster repräsentieren dabei die Alternativen, die Datensätze selbst die Bewertungskriterien. Bei der Methode des *Boolean Overlay* oder *Boolean Intersection* erfolgt eine Bewertung lediglich zwischen den Kategorien ‚geeignet' und ‚ungeeignet'. Durch diese Dichotomie können vorher definierte *Constraints* aus der weiteren Analyse ausgeschlossen werden. Die *Constraints* lassen sich über logische Operatoren (*AND, OR*) verknüpfen (Jiang & Eastman 2000). In der Praxis werden ungeeigneten Flächen Pixelwerte von 0 zugewiesen. Durch eine anschließende Multiplikation aller Raster untereinander (*OR-Operator*) oder deren Addition (*AND-Operator*) erhält man einen Rasterdatensatz, welcher Ausschlussflächen durch den Wert 0 repräsentiert (Abb.4).

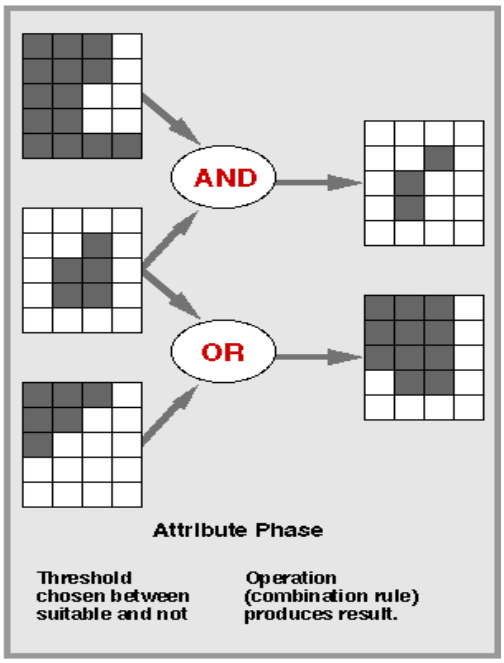


Abbildung 4: *Boolean Overlay*[2]

[2] Quelle: http://loi.sscc.ru/gis/G460/Lec08.html (Stand: 21.06.2014, verändert)

2.3.2 *Weighted Linear Combination*

Eines der häufigsten Verfahren um Datensätze zu kombinieren, ist die Methode der *Weighted Linear Combination* (WLC) (Malczewski 2004). Im Gegensatz zur Methode des *Boolean Overlay* ist es nicht das Ziel der WLC eine dichotome Abgrenzung zwischen geeigneten und ungeeigneten Flächen zu erreichen. Vielmehr geht es darum, den Grad der Eignung für eine bestimmte Nutzung anhand verschiedener Kriterien zu bewerten. Da die Kriterien in verschiedenen Einheiten gemessen werden, ist es notwendig, diese auf eine einheitliche Skala (Werteskala) zu transformieren (Drobne & Lisec 2009). Voogd (1983) erläutert mehrere Möglichkeiten, wie diese Standardisierung zu bewerkstelligen ist. Die einfachste ist die lineare Skalierung, die durch folgende Gleichung ausgedrückt wird (Drobne & Lisec 2009):

$$x_i = \frac{(R_i - R_{min})}{(R_{max} - R_{min})} * SR \qquad (1)$$

R_i stellt dabei den Ursprungswert, R_{min} den Minimalwert und R_{max} den Maximalwert innerhalb des zu standardisierenden Rasters dar. SR ist die Spannweite, auf die die Ursprungswerte standardisiert werden sollen. x_i ist der standardisierte Wert. Da der Wert 0 für die geringstmögliche Erfüllung des Teilziels steht, ist das Ergebnis der Berechnung von 1 zu subtrahieren, falls hohe Ursprungswerte eines Kriteriums bedeuten, dass das Teilziel eher nicht erfüllt wird und umgekehrt:

$$x_i = 1 - \frac{(R_i - R_{min})}{(R_{max} - R_{min})} * SR \qquad (2)$$

Jeder standardisierte Wert wird anschließend mit einem Faktor multipliziert, der den relativen Einfluss der einzelnen Kriterien repräsentiert. Die Vergabe der Gewichte erfolgt subjektiv durch den Entscheidungsträger und beeinflusst maßgeblich das Endergebnis (Feizizadeh & Blaschke 2014). Die Ergebnisse werden letztlich aufsummiert.

Nach Hansen (2005), der sich auf die Ausführungen von Jiang & Eastman (2000) stützt, kann dieses Vorgehen durch folgende Formel dargestellt werden:

$$S^k = \sum w_i x_i^k \quad (3)$$

w_i steht hierbei für die zugeordneten Gewichte und x_i^k für den Wert der Entscheidungsvariablen von Kriterium i im k-ten Pixel. S^k gibt dann den Grad der Eignung im k-ten Pixel wieder. Abbildung 5 verdeutlicht dieses Vorgehen nochmals graphisch:

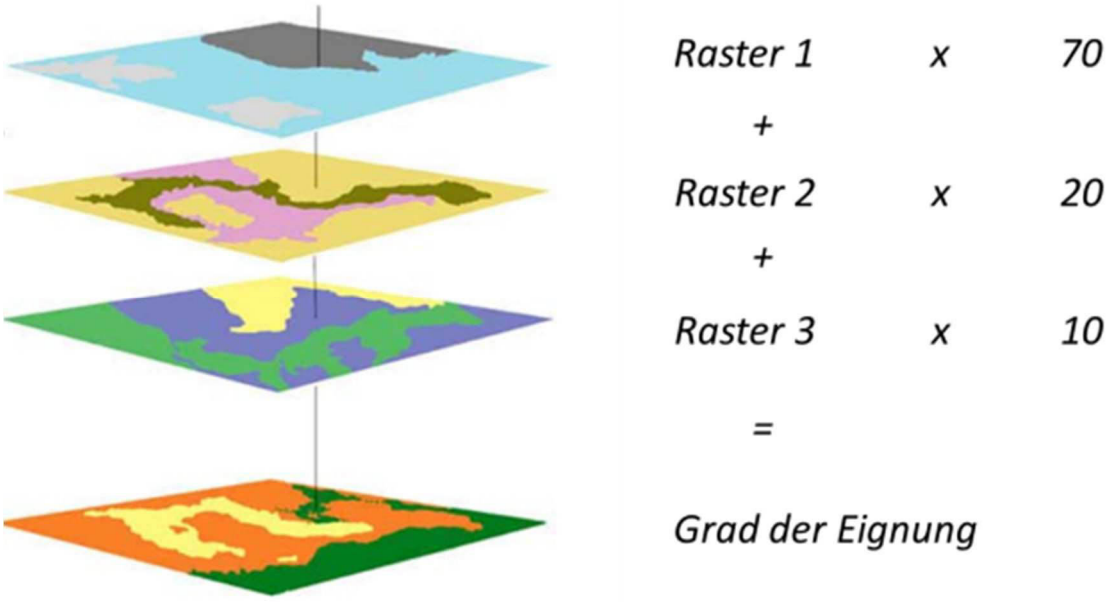


Abbildung 5: Prinzip *Weighted Linear Combination*[3]

Hansen (2005) betont, dass die Summe der Gewichte 1 ergeben muss. Dies kann so nicht stehen gelassen werden, da die Gewichte lediglich in Relation zu ihrer Summe gesehen werden müssen. Für die Funktionsweise der WLC hat dies keine Bedeutung. Dennoch sollten sich die Gewichte zu einer leicht fassbaren Zahl aufaddieren, um die Interpretation des Ergebnisses nicht unnötig zu erschweren. Hilfreich für die Interpretation ist ebenfalls eine Standardisierung des Ergebnisrasters.

[3]Quelle: http://www.fao.org/docrep/006/y4816e/y4816e0g.htm (Stand: 27.05.2014, verändert)

Zusammenfassend lässt sich der Prozess der WLC in 6 Schritte gliedern (Malczewski 1999):

1. Definieren der Bewertungskriterien (*Map Layer*)
2. Standardisieren der *Map Layer*
3. Zuordnen von Gewichten zu jedem standardisierten *Map Layer*
4. Multiplizieren der *Map Layer* mit den zugeordneten Gewichten
5. Aufaddieren der multiplizierten *Map Layer*
6. Bewerten und klassifizieren der Alternativen

Kritik an der WLC richtet sich vor allem gegen drei Aspekte. Zum einen wird die Standardisierung der Kriterien bemängelt. Häufig erfolgt eine einfache lineare Transformation, ohne dieses Vorgehen genauer zu begründen. Eine nicht-lineare Transformation könnte in einigen Fällen jedoch besser geeignet sein (Eastman 1999, Feizizadeh & Blaschke 2014). Zum anderen geht die Methode davon aus, dass ein niedriger Wert eines Kriteriums durch einen hohen Wert eines anderen Kriteriums ersetzt werden kann. Dies wird auch als *„trade off“* oder *„substitutability“* bezeichnet *(*Jiang & Eastman 2000). Die Kriterien werden somit als unabhängig voneinander betrachtet und eine Korrelation ausgeschlossen. Folge kann eine fehlerhafte Flächenbewertung sein (Malczewski 1999). Der dritte Kritikpunkt betrifft das Entscheidungsrisiko. Während bei *boolschen* Entscheidungsregeln das Risiko eine falsche Entscheidung zu treffen, über Messfehler abgeschätzt werden kann, ist dies bei den kontinuierlichen Kriterien der WLC nicht der Fall (Jiang & Eastman 2000, Drobne & Lisec 2009).

Trotz der aufgeführten Kritikpunkte ist die Methode der WLC weit verbreitet. Malczewski (2004) führt dies auf die leichte Verständlichkeit und intuitive Anwendbarkeit zurück. Zudem lässt sich die Methode leicht durch den Einsatz von *Map Algebra* in einer GIS-Umgebung implementieren. Aus diesen Gründen findet sie auch in der vorliegenden Arbeit Anwendung. Hinzu kommt die Tatsache, dass sich mit der WLC auch die Entscheidungsregel des *Analytical Hierarchy Process* (AHP) problemlos umsetzen ließe[4]. Hier müssten vorab die relativen Gewichte über den paarweisen Vergleich der

[4] Eine Erläuterung des AHP sowie weitere Methoden von MKREs finden sich u.a. bei Malczewski (1999) und Sugumaran & Degroote (2011).

Kriterien generiert werden. Die Aggregation der gewichteten Kriterien ist schließlich ident zur Methode der WLC.

2.4 *Spatial Decision Support Systems*

Ein Blick in die Literatur verdeutlicht, dass der Begriff SDSS nicht eindeutig definiert ist (Sugumaran & Degroote 2011). Grob lässt sich ein SDSS als ein interaktives computerbasiertes System definieren, welches zum Ziel hat, den Entscheidungsträger bei der Lösung semistrukturierter räumlicher Entscheidungsprobleme zu unterstützen (Densham 1991, Malczewski 1999). Folge dieser unpräzisen Definition ist, dass GIS-basierte Analysen im Allgemeinen als SDSS bezeichnet werden bzw. SDSS häufig durch Aspekte gekennzeichnet werden, welche auch GI-Software charakterisieren (Keenan 2006). Bottero et al. (2013) verweisen beispielsweise auf die Fähigkeit eines SDSS zur Datenmanipulation, Datenhaltung und Visualisierung und setzen damit den Begriff SDSS mit GIS gleich. Obgleich der Begriff SDSS von Bottero et al. (2013) unvollständig charakterisiert wird, kommt dennoch die fundamentale Rolle zum Vorschein, die GIS innerhalb eines SDSS einnimmt (Sugumaran & Degroote 2011).

Laudien et al. (2007) verweisen darauf, dass ein SDSS „eine Entscheidungsunterstützung, basierend auf einem definierten Entscheidungsbaum“ generiert und es dem Experten ermöglicht die Prozessierung der Daten durch sein Wissen verändern zu können. Die individuelle Veränderung der Datenprozessierung ermöglicht Szenario-Analysen und trägt zur iterativen Problemlösung bei. Des Weiteren erfolgt eine Fokussierung auf den Anwender bzw. Entscheidungsträger als Experte für ein spezifisches Problem. Der Nutzer muss somit kein GIS-Experte sein, was eine leichte Bedienbarkeit des Systems voraussetzt. Sugumaran & Degroote (2011) betonen zudem die Fähigkeit eines SDSS Berichte in Form von Karten, Tabellen und Grafiken zu generieren.

Da keine einheitliche Defintion des Begriffs SDSS existiert, ist es für die hier vorliegende Arbeit nötig, den Begriff an dem Vorhandensein bestimmter Kriterien festzumachen, an welchen das Ergebnis der Arbeit gemessen werden kann. Auf die GIS-spezifischen Bestandteile, wie Datenmanagement, Datenanalyse und Visualisierung, wird hierbei nicht eingegangen, da das geplante Tool innerhalb einer GI-Software entwickelt wird, die diese Komponenten bereitstellt. Diese Sichtweise impliziert, dass

die notwendige Technik für ein SDSS durch ein GIS geliefert wird. Demnach ist ein SDSS eine GIS-Modifikation, welche für ein spezifisches Problem entwickelt wurde (Keenan 2006).

Aufbauend auf der vorangehenden Diskussion des Begriffs und in Anlehnung an Sugumaran & Degroote (2011) wird ein SDSS für diese Arbeit folgendermaßen definiert:

> Ein SDSS ist ein interaktives computerbasiertes System, welches zum Ziel hat den Entscheidungsträger bei der Lösung konkreter semistrukturierter räumlicher Entscheidungsprobleme zu unterstützen und durch folgende Aspekte charakterisiert ist:
>
> - Leichte Bedienbarkeit,
> - Möglichkeit zu Szenario-Analysen (Wenn-Dann-Analysen) bzw. individuelle Veränderung der Datenprozessierung,
> - Beitrag zur iterativen Problemlösung,
> - und Berichterstellung.

Die Bezeichnung ‚semistrukturiert' verweist darauf, dass nur Teilbereiche des Entscheidungsproblems algorithmierbar sind und sich strukturiert lösen lassen. Für die endgültige Entscheidungsfindung sind jedoch Beurteilungsschritte notwendig, die von einem menschlichen Problemlöser auszuführen sind (Schiemenz & Schönert 2005). Letztlich impliziert der Begriff SDSS selbst den semistrukturierten Charakter des Entscheidungsproblems: Das System unterstützt den Prozess der Entscheidungsfindung, kann dem Entscheidungsträger die Entscheidung allerdings nicht abnehmen.

Einige Autoren verwenden auch den Begriff *Multi Criteria SDSS* (MC-SDSS), um auf den multikriteriellen Charakter ihrer Analysen hinzuweisen (Simão et al. 2009, Bottero et al. 2013). Karnatak et al. (2007) verweisen darauf, dass es sich hierbei um einen Teil des weiten Feldes von SDSS handelt. Nach der Argumentation von Drobne & Lisec (2009) ist diese Unterscheidung allerdings überflüssig, da räumliche Entscheidungen grundsätzlich multikriterieller Natur seien. Aus diesem Grund wird in dieser Arbeit auf den Begriff MC-SDSS verzichtet und lediglich der Begriff SDSS verwendet.

3 Standortanalyse für Windkraftanlagen

3.1 Standortkriterien

Eine der größten Schwierigkeiten bei der Entwicklung von Windenergieprojekten stellt das Auffinden geeigneter Flächen dar, zumal sich die Gebiete, welche sowohl technisch als auch ökonomisch für die Windenergienutzung geeignet sind, Jahr für Jahr reduzieren. Die Größe und Anzahl der Gebiete hängt von spezifischen lokalen und regionalen Eigenheiten und Beschränkungen ab (Grassi et al. 2012). Eine Vielzahl von Studien hat sich bisher mit der Standortanalyse für Windkraftanlagen beschäftigt und dafür verschiedenste Kriterien in Betracht gezogen (Al-Yahyai et al. 2012, Baban & Parry 2001, Bennui et al. 2007, Gorsevski et al. 2013, Grassi et al. 2012, Janke 2010, Rodman & Meentemeyer 2006, Sliz-Szkliniarz & Vogt 2011, van Haaren & Fthenakis 2011, Voivontas et al. 1998). Eine mögliche Klassifikation dieser Kriterien stellt jene in politische, ökonomische und technische dar (Grassi et al. 2012). Einige Studien führen auch umweltspezifische bzw. ökologische und/oder anthropogene Kriterien an (Gorsevski et al. 2013, Rodman & Meentemeyer 2006, van Haaren & Fthenakis 2011). Da das zu entwickelnde Tool für die Flächenanalyse und -bewertung innerhalb der BRD gedacht ist, ist eine solche zusätzliche Unterscheidung in dieser Arbeit nicht notwendig, da dies bereits durch politische Faktoren geregelt ist. Als politische Kriterien gelten dabei solche, welche auf einer legislativen Grundlage basieren. Diese bestimmen die primären Ausschlussflächen und sind somit als *Constraints* zu behandeln. Die Regelungen hierzu sind teils regionalspezifisch. Zu den *Constraints* gehören beispielsweise FFH-Gebiete, Naturschutzgebiete, Abstandsflächen zu Wohngebäuden oder Gewerbegebieten. Die rechtliche Grundlage für die Ausweisung von Windeignungsgebieten bildet in Deutschland u.a. das Raumordnungsgesetz, das Bundesimmissionsschutzgesetz, das Bundesnaturschutzgesetz, das Baugesetzbuch, sowie bundeslandspezifische raumplanerische Gesetzgebungen (Heier 2008).

Da nicht alle politisch verfügbaren Gebiete technisch und/oder ökonomisch für die Realisierung von Windenergieprojekten geeignet sind, kommt es durch die Betrachtung der technischen und ökonomischen Aspekte zu einer weiteren Reduzierung der verfügbaren Flächen. Eine eindeutige Unterscheidung zwischen ökonomischen und

technischen Kriterien ist durch die Literatur nicht gegeben. Van Haaren & Fthenakis (2011) rechnen beispielsweise die Windressourcen zu den ökonomischen Faktoren, wohingegen Grassi et al. (2012) und Sliz-Szkliniarz & Vogt (2011) diese den technischen Aspekten zuordnen, ohne dies jedoch genauer zu erläutern. Grund dürfte jedoch sein, dass die Windressourcen keine direkten Investitionen per se erforderlich machen und zudem die voraussichtliche Energieproduktion von der technischen Spezifikation des geplanten Anlagentyps abhängig ist. Aus diesen Gründen werden die Windressourcen bzw. der Energieertrag in dieser Arbeit zu den technischen Kriterien gerechnet. Die enorme ökonomische Bedeutung dieses Faktors im Hinblick auf Profit und mögliche Investitionsentscheidungen soll dadurch nicht negiert werden. Eine Zuordnung der Windressourcen zu den umweltspezifischen Kriterien, wie sie von einigen Autoren vorgenommen wird (Gorsevski et al. 2013), erscheint für diese Arbeit nicht sinnvoll. Auch wenn es sich um ein natürliches Phänomen handelt und eine solche Klassifizierung deshalb nachvollzogen werden kann, geht es in dieser Arbeit um die Entwicklung eines SDSS zur Bewertung von Flächen hinsichtlich ihrer technischen und ökonomischen Eignung für die Windenergienutzung.

Zu den ökonomischen Kriterien zählt der Abstand zu Straßen und zu Umspannwerken, die als Einspeisepunkte in das Stromnetz dienen. Der Bau von Zuwegungen und die Verlegung von Kabeltrassen zur Netzeinspeisung sind mit entsprechenden Investitionen verbunden (Gorsevski et al. 2013, Grassi 2012, Mann et al. 2012, van Haaren & Fthenakis 2011). Auch die Hangneigung kann durch etwaig anfallende Erdarbeiten für die Anlagenerrichtung zu Mehrkosten führen. Zudem erschwert unwegsames Gelände den Bau von Zuwegungen und Kabeltrassen (Voivontas et al. 1998).

Zusammenfassend sind in Tabelle 2 sämtliche Kriterien aufgelistet, die in das zu entwickelnde SDSS implementiert werden. Da das SDSS für ein Gebiet in Thüringen umgesetzt wird, wurden die politische Kriterien den Empfehlungen des Thüringer Ministeriums für Bau, Landesentwicklung und Verkehr (2005) sowie den von der Bund-Länder-Initiative ‚Windenergie' herausgegebenen Abstandsempfehlungen (2013) entnommen. Aufgrund der Datenlage konnten in der vorliegenden Arbeit jedoch nicht alle in diesen Dokumenten aufgeführten Kriterien Berücksichtigung finden.

Tabelle 2: Kriterien der Standortanalyse

Ökonomische	**Technische**	**Politische**
Abstand zu Straßen Abstand zu Umspannwerken Hangneigung	Windressourcen bzw. Energieertrag eines spezifischen Anlagentyps	Primäre Abstands- bzw. Ausschlussflächen: Vogelschutz-, Landschaftsschutz-, FFH-, Naturschutzgebiete, Wohn-, Gewerbe-, Industriegebiete, Bahnnetz, Autobahnen, Bundes-, Landes-, Kreisstraßen

3.2 Prozess der Standortanalyse

Der Prozess der Standortanalyse hat zum Ziel, geeignete Flächen für die Installation von Windkraftanlage zu finden und diese zu bewerten. Grassi et al. (2012) sprechen von geeigneten Flächen („eligible areas“), wenn es dort sowohl formaljuristisch als auch technisch möglich ist, Windkraftanlagen zu errichten. Standortspezifische Windverhältnisse finden hierbei jedoch noch keine Berücksichtigung. Für eine umfassende und sinnvolle Bewertung ist dieser Aspekt jedoch unerlässlich. Das Auffinden jener Gebiete, in welchen alle erforderlichen Voraussetzungen für die erfolgreiche Verwirklichung eines Windparks gegeben sind, beschreiben die Autoren als komplexen und iterativen Prozess, der als erster Schritt in der Bewertung des Windpotenzials einer Region angesehen werden kann.

Der Prozess der Standortanalyse lässt sich in vier Schritte gliedern:

1. Definieren von Kriterien. Bottero et. al (2013) untergliedern diese Kriterien weiter in Faktoren und *Constraints*. Faktoren sind für gewöhnlich kontinuierlicher Natur und dienen dem Bewerten der Eignung einer Fläche für einen bestimmten Zweck. *Constraints* dienen, wie bereits erwähnt, dem Definieren von Ausschlussflächen.
2. Dem Gewichten der Faktoren (Abschnitt 2.3.2).
3. Ausschluss der definierten *Constraints* (Abschnitt 2.3.1).
4. Interpretieren des Ergebnisses.

Abbildung 6 ist an Gorsevski et al. (2013) angelehnt und an den Prozess der Windkraftplanung angepasst, der in der vorliegenden Arbeit zur Anwendung kommt. Es handelt sich hierbei um einen Wertebaum, der das Entscheidungsproblem strukturiert. Übergeordnetes Ziel *(Goal)* stellt das Selektieren der idealsten Gebiete für die Windkraftnutzung dar. Eine strikte Trennung zwischen *Constraints* und Faktoren erfolgt in diesem Fall nicht, da das zu entwickelnde SDSS dem Benutzer ermöglichen soll, Faktorwerte aus der Analyse auszuschließen. Deshalb sind sämtliche Faktoren zusätzlich als *Constraints* aufgeführt. *Objectives* spiegeln Teilziele wieder, die durch mehrere Faktoren repräsentiert werden können.

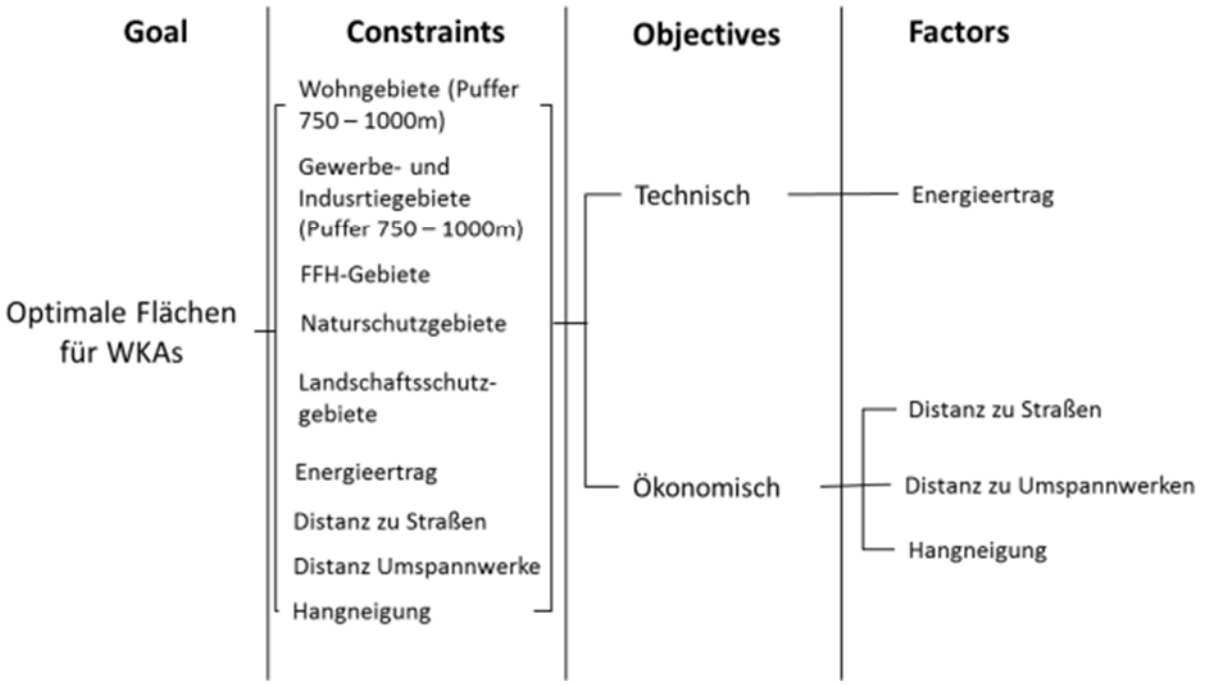


Abbildung 6: Wertebaum des Entscheidungsproblems (Quelle: Gorsevski et al. 2013, verändert)

Abbildung 7 stellt den Prozess der Standortanalyse nochmals als Flussdiagramm dar: Zu Beginn steht die Auswahl der Untersuchungsregion und das Definieren der Faktoren. Die ausgewählten Faktoren werden nach zuvor durchgeführter Standardisierung über die Methode der WLC kombiniert. Zwischenergebnis ist eine erste Eignungskarte. Anschließend folgt das Definieren der *Constraints*. Diese werden mittels *Boolean Overlay* mit dem Ergebnis der WLC verschnitten. Somit erhält man eine Karte, welche den Grad der Eignung und die technisch-ökonomisch hochwertigen Gebiete innerhalb der „eligible areas“ wiederspiegelt. Können geeignete Flächen, im Sinne von technisch-

ökonomisch sinnvollen Gebieten, ausfindig gemacht werden, endet der Prozess. Anderenfalls muss eine neue Untersuchungsregion ausgewählt werden. Sinnvoll ist stets eine Sensitivitätsanalyse, welche das Risiko einer Fehlentscheidung minimiert.

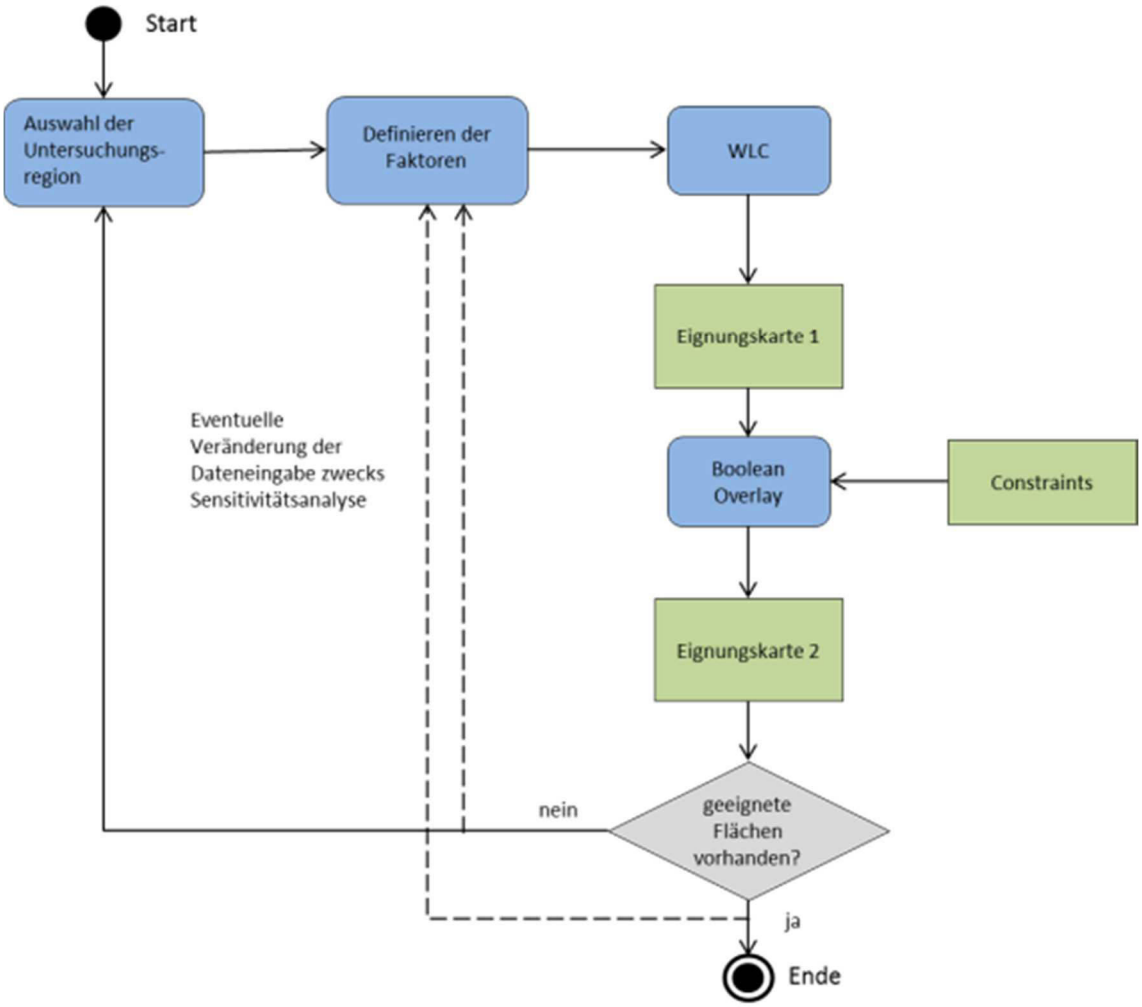


Abbildung 7: Aktivitätsdiagramm Standortplanung

4 Windenergiepotenzial

4.1 WEA-Leistungskurve

Die meisten Studien, die sich mit der Bewertung von Flächen für die Windenergienutzung beschäftigen, verwenden für die Bestimmung des Energiepotenzials die mittlere Windgeschwindigkeit (Almoataz et al. 2012, Baban & Parry 2001, Gass et al. 2013, Gorsevski et al. 2013, Mari et al. 2011, Rodman & Meentemeyer 2006, Tegou et al. 2010, Voivontas et al. 1998). Das tatsächliche Energiepotenzial eines Standorts ist jedoch nicht von der mittleren Windgeschwindigkeit, sondern von der Häufigkeitsverteilung der Windgeschwindigkeiten (Kapitel 4.2) und dem geplanten WEA-Typ abhängig. Dies ist auf die Leistungskennlinie oder Leistungskurve zurückzuführen, die für jede WEA durch den Anlagenhersteller bestimmt wird. Sie stellt den Zusammenhang zwischen Windgeschwindigkeit in Nabenhöhe und produzierter Leistung dar. Zudem wird durch die Leistungskurve sowohl jene Windgeschwindigkeit definiert, bei welcher die WEA mit der Energieproduktion beginnt (Einschaltgeschwindigkeit), als auch jene, bei welcher sich die Anlage, aufgrund möglicher Überlastung, abschaltet (Abschaltgeschwindigkeit). Abbildung 8 stellt den Verlauf der Leistungskennlinie eines spezifischen Anlagentyps dar. Es handelt sich um eine Anlage des Herstellers ENERCON, die auch bei der späteren Bestimmung des Energiepotenzials herangezogen wird. Die Absolutwerte der Leistungskurve sind Anhang A bzw. Tabelle 4 zu entnehmen.

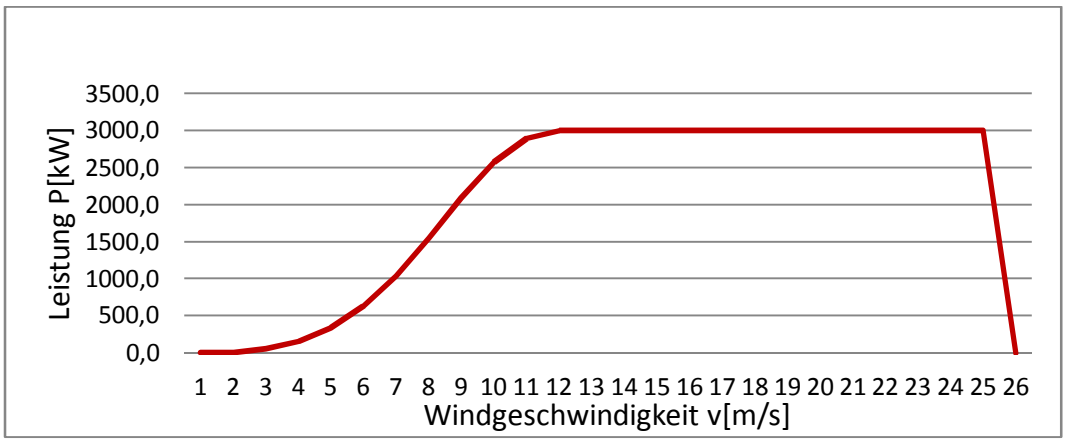


Abbildung 8: Leistungskurve E115 (3.0 MW), (Quelle: Enercon 2013, eigene Darstellung)

In nachfolgender Tabelle sind die wichtigsten Daten der ENERCON E115 (3.0 MW) aufgeführt.

Tabelle 3: Anlagenspezifikation E115 (3.0 MW), (Quelle: Enercon 2013, eigene Darstellung)

ENERCON E115 (3.0 MW)	
Hersteller	ENERCON
Nennleistung	3.0 MW
Rotordurchmesser	115 m
Einschaltgeschwindigkeit	2 m/s
Abschaltgeschwindigkeit	26 m/s
Erreichen der Nennleistung	12 m/s
Nabenhöhe	149 m

Die Energieproduktion beginnt bei einer Windgeschwindigkeit von 2 m/s. Die Nennleistung von 3 MW wird ab 12 m/s erreicht. Bei Windgeschwindigkeiten größer als 25 m/s schaltet sich die Anlage ab.

4.2 Weibullverteilung

Wie oben dargestellt ist für die Bestimmung der Energieproduktion einer WEA die Häufigkeitsverteilung der Windgeschwindigkeiten heranzuziehen. Diese Häufigkeitsverteilung wird im Allgemeinen mittels der sog. Weibullverteilung beschrieben (Emeis 2001, Gass et al. 2013, Ouammi et al. 2012):

$$f(u) = \frac{k}{A} * \left(\frac{u}{A}\right)^{k-1} \exp\left[-\left(\frac{u}{A}\right)^{k}\right] \quad (4)$$

Die Verteilung wird durch zwei Parameter definiert: Den A-Faktor oder Skalenfaktor. Dieser ist direkt proportional zur mittleren Windgeschwindigkeit und besitzt die Einheit m/s sowie den dimensionslosen k-Faktor, der auch als Formfaktor bezeichnet wird. f(u) steht für die Häufigkeit des Auftretens einer bestimmten Windgeschwindigkeit u. In Abbildung 9 sind unterschiedliche Weibullverteilungen für variierende k-Werte und einen konstanten A-Wert dargestellt.

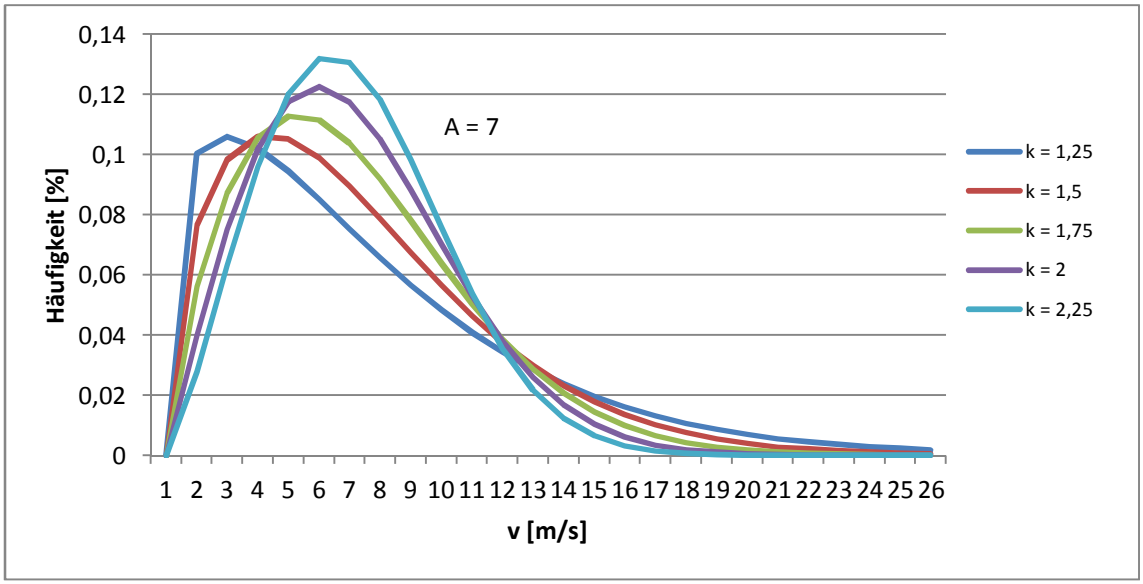


Abbildung 9: Weibullverteilung für verschiedenen k-Werte und konstanten A-Wert

Eine Vergrößerung des k-Faktors hat eine Erhöhung des Scheitelpunktes sowie eine Verringerung der Rechtsschiefe der Kurve zur Folge. Mit größerem k-Wert nehmen die Häufigkeiten geringerer Windgeschwindigkeiten sowie die Häufigkeiten sehr hoher Windgeschwindigkeiten ab. Eine Vergrößerung des A-Wertes führt zu einer Verschiebung der Kurve nach rechts. Der Scheitelpunkt wird durch eine Erhöhung des A-Wertes jedoch niedriger. Die Häufigkeiten geringerer Windgeschwindigkeiten nehmen dadurch ab und jene hoher Windgeschwindigkeiten zu (Abb. 10).

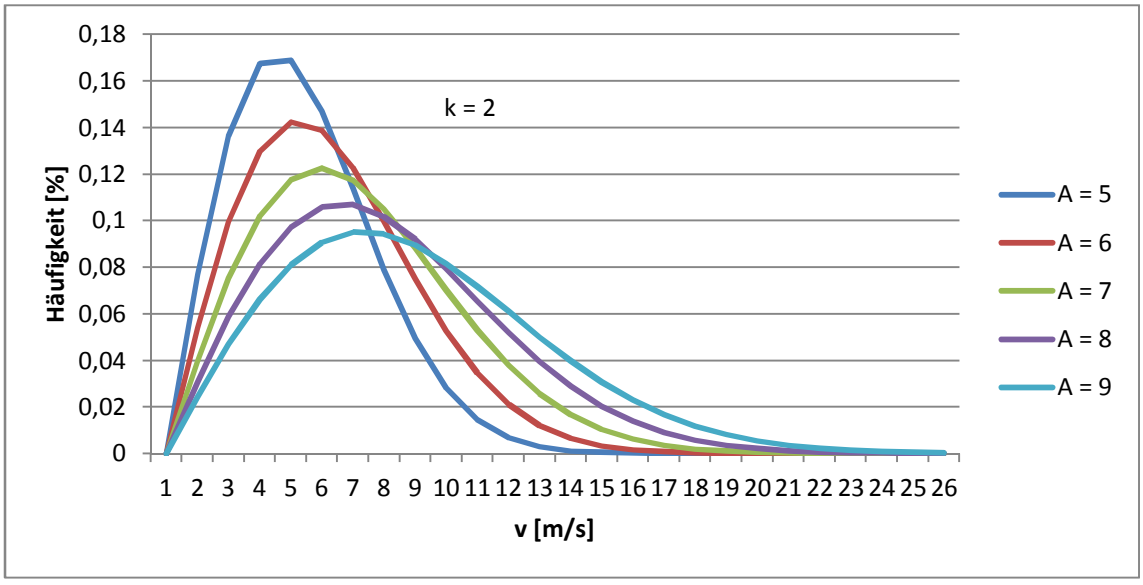


Abbildung 10: Weibullverteilung für verschiedene A-Werte und konstanten k-Wert

4.3 Vertikale Extrapolation der Weibullverteilung

Da aufgrund der Datenlage die Weibullparameter nicht in jeder beliebigen Höhe verfügbar sind, ist es notwendig diese auf die Nabenhöhe der gewünschten WEA zu extrapolieren. Für die Extrapolation der mittleren Windgeschwindigkeit auf größere Höhen wird häufig auf das hellmannsche Potenzgesetz zurückgegriffen (Cabello & Orza 2010, Emeis 2001, McWilliam et al. 2012, Yeh & Wang 2008):

$$\frac{v_1}{v_2} = \left(\frac{h_1}{h_2}\right)^{\alpha} \quad (5)$$

v_1 steht für die Windgeschwindigkeit in der Höhe h_1 und v_2 für die Windgeschwindigkeit in der Höhe h_2. Der Exponent α ist eine von der Bodenrauigkeit abhängige Konstante, die Werte zwischen 0 und 1 annehmen kann. Zudem gilt $v_2 > v_1$.

Justus & Mikhail (1976) zeigen, dass sich der A-Wert mit zunehmender Höhe genauso verhält wie die Windgeschwindigkeit, sodass analog zum Potenzgesetz gilt:

$$\frac{A_1}{A_2} = \left(\frac{h_1}{h_2}\right)^{\alpha} \quad (6)$$

Nach Yeh & Wang (2008) ist das Potenzgesetz bis in Höhen von 150 m sinnvoll anwendbar. Für Emeis (2001) gilt dieser Zusammenhang allerdings nur für die unterste atmosphärische Schicht, die sog. Prandtl-Schicht, welche nach seinen Untersuchungen bereits in Höhen zwischen 50 und 80 m (atmosphärische Grenzschicht) in die sog. Ekmann-Schicht übergeht. Folglich müssten in Höhen über 80 m die Gesetze der Ekmann-Schicht zur Anwendung kommen. Der Autor erwähnt jedoch, dass dieser Bereich der Atmosphäre erst wenig erforscht sei. Auch die durch ihn vorgeschlagenen Lösungsansätze für die Abbildung der Windverhältnisse in einer beliebigen Höhe der Ekmann-Schicht sind für die hier vorliegende Arbeit ungeeignet, da die notwendigen Parameter sowohl über als auch unter der atmosphärischen Grenzschicht bekannt sein bzw. erst durch Messung gewonnen werden müssten. Ohne die Aussage von Yeh & Wang bzw. Emeis hinsichtlich der Höhe der Anwendbarkeit des Potenzgesetzes bestätigen noch verwerfen zu können, bleibt bei der vorliegenden Arbeit aufgrund der Datenlage nichts anderes übrig als das Potenzgesetz für die Extrapolation des A-Wertes heranzuziehen.

Was den Exponenten α betrifft, so nimmt dessen Wert mit zunehmender Bodenrauigkeit ebenfalls zu. Grassi et al. (2012) und Rozsavolgyi (2009) bestimmen den Wert von α über Landbedeckungs-Klassen. Kritisch anzumerken ist bei diesem Vorgehen, dass so lediglich die Rauigkeit an einem Punkt wiedergegeben werden kann. Die Windscherung an einem Standort ist jedoch von der Bodenrauigkeit in der weiteren Umgebung abhängig. Boudia et al. (2012) und Cabello & Orza (2010) greifen zur Festlegung von α auf Untersuchungen von Justus & Mikhail (1976) zurück. Diese definieren den Wert des Rauigkeitsexponenten über folgende Formel:

$$\alpha = \frac{0{,}37 - 0{,}0881 * \ln A_1}{1 - 0{,}0881 * \ln(h_1/10)} \qquad (7)$$

Doran & Verholek (1977) weisen darauf hin, dass diese Formel auf einem begrenzten Datensatz beruht und deshalb mit Vorsicht zu verwenden sei. McWilliam et al. (2012) ziehen bei ihrer Arbeit einen Standardwert für α von 1/7 heran. Cabello & Orza (2010) erwähnen, dass dieser Wert für gewöhnlich gebraucht wird. Einer Betrachtung der spezifischen Standorteigenschaften kann bei diesem Vorgehen allerdings nicht Rechnung getragen werden, was die Unsicherheiten bei der Extrapolation weiter erhöht. Für eine exaktere Bestimmung des Rauigkeitsexponenten ist das Vorhandensein von Windgeschwindigkeiten bzw. A-Werten in zwei verschiedenen Höhen notwendig. Das Potenzgesetz kann dann folgendermaßen umgewandelt werden:

$$\frac{A_1}{A_2} = \left(\frac{h_1}{h_2}\right)^{\alpha} \rightarrow \alpha = \frac{\ln(A_1/A_2)}{\ln(h_1/h_2)} \qquad (8)$$

Dieser Ansatz kommt auch bei der vorliegenden Arbeit zur Anwendung. Mit dem Bekanntsein von α lässt sich dann der A-Wert, durch die Auflösung des hellmannschen Potenzgesetzes, in Nabenhöhe, wie folgt, berechnen:

$$A_3 = \frac{A_2}{\left(\frac{h_2}{h_3}\right)^{\alpha}} \qquad (9)$$

Für die Extrapolation des k-Werts schlagen Justus & Mikhail (1976) folgende Formel vor:

$$\frac{k_2}{k_1} = \frac{1-0{,}0881*\ln(h_1/10)}{1-0{,}0881*\ln(h_2/10)} \qquad (10)$$

Die von Doran & Verholek (1977) angebrachte Kritik an dieser Formel bezieht sich dabei auf die gleichen Punkte wie bei der Bestimmung des Rauigkeitsexponenten. Boudia et al. (2012) extrapolieren mit diesem Ansatz den k-Wert auf 50 m Höhe. Für eine Extrapolation des k-Werts in den Bereich der Ekmann-Schicht ist dieses Vorgehen allerdings ungeeignet. Der k-Wert müsste nach dieser Formel mit zunehmender Höhe ebenfalls kontinuierlich zunehmen. Emeis (2001) zeigt, dass der Parameter seinen Maximalwert zwischen 40 und 80 m Höhe im Bereich der atmosphärischen Grenzschicht erreicht und danach wieder abnimmt. Nach Tong (2010) kann der k-Wert bis zu Höhen von 100 m zunehmen. Auch Justus & Mikhail (1976) ist die Tatsache des abnehmenden k-Werts ab einer gewissen Höhe bewusst. Sie weisen deshalb darauf hin, dass die oben aufgeführte Formel lediglich bis 100 m Höhe anwendbar sei. Da für das zu entwickelnde SDSS der k-Wert als Rasterdatensatz in 80 m über Grund vorliegt, und sich somit im Bereich seines möglichen Maximum befindet, erscheint eine Extrapolation des k-Parameters als unsinnig. Zudem verändern sich die k-Werte zwischen 50 und 200 m nur geringfügig (Abb. 11). Folglich wird in der vorliegenden Arbeit nur der A-Wert auf Nabenhöhe extrapoliert.

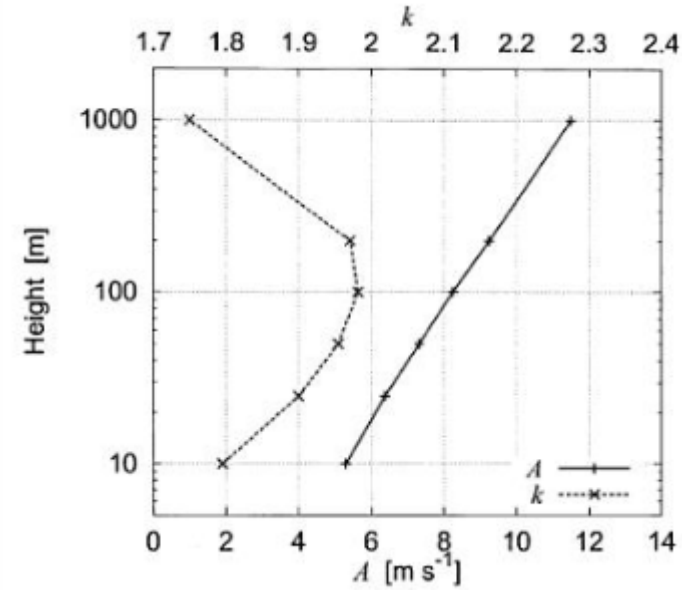


Abbildung 11: Verlauf des A- und k-Parameters mit der Höhe (Petersen et al. 1997)

4.4 Energieertragsberechnung

Zur Berechnung des Energieertrags einer WEA an einem spezifischen Standort muss die Weibullverteilung in Nabenhöhe mit der Leistungskurve der Windkraftanlage in Beziehung gesetzt werden. Der jährliche Energieertrag E_a berechnet sich über das Produktintegral aus den Funktionen der Leistungskurve und der Weibullverteilung (Ramirez-Rosado et al. 2008, Tong 2010):

$$\mathrm{E_a} = \int_2^{25} \mathrm{P(u)f(u)du} * 8760\mathrm{h/a} \qquad (11)$$

Das Integral reicht von der Einschalt- bis zur Abschaltgeschwindigkeit der Anlage. P(u) steht für die Funktion der Leistungskurve, f(u) für die Funktion der Weibullverteilung. 8760 ist die Anzahl der Stunden pro Jahr. Tong (2010) weist darauf hin, dass sich dieses Integral analytisch nicht lösen lässt. Daher müssen die Klassenhäufigkeiten der einzelnen Windgeschwindigkeiten mit der Leistung der Windkraftanlage bei der jeweiligen Windgeschwindigkeit multipliziert und die resultierenden Klassenerträge aufaddiert werden. Die Berechnung des Energieertrags ist in Abbildung 12 veranschaulicht. Tabelle 4 zeigt ebenfalls das Vorgehen bei der Energieertragsberechnung. In Tabelle 4 wurde eine Weibullverteilung mit einem A-Wert von 7 und einem k-Wert von 2 sowie die im Anhang A befindliche Leistungskurve der E115-Anlage mit 3.0 MW Nennleistung verwendet. Der Tabelle ist auch zu entnehmen, dass sich die Häufigkeiten der einzelnen Windgeschwindigkeitsklassen nicht zu 1 aufaddieren. Dies liegt daran, dass Windgeschwindigkeiten unterhalb der Einschaltgeschwindigkeit bzw. oberhalb der Abschaltgeschwindigkeit bei der Energieertragsberechnung keine Berücksichtigung finden. In diesem Beispiel kann die WEA 95.66% des auftretenden Windes für die Stromerzeugung heranziehen.

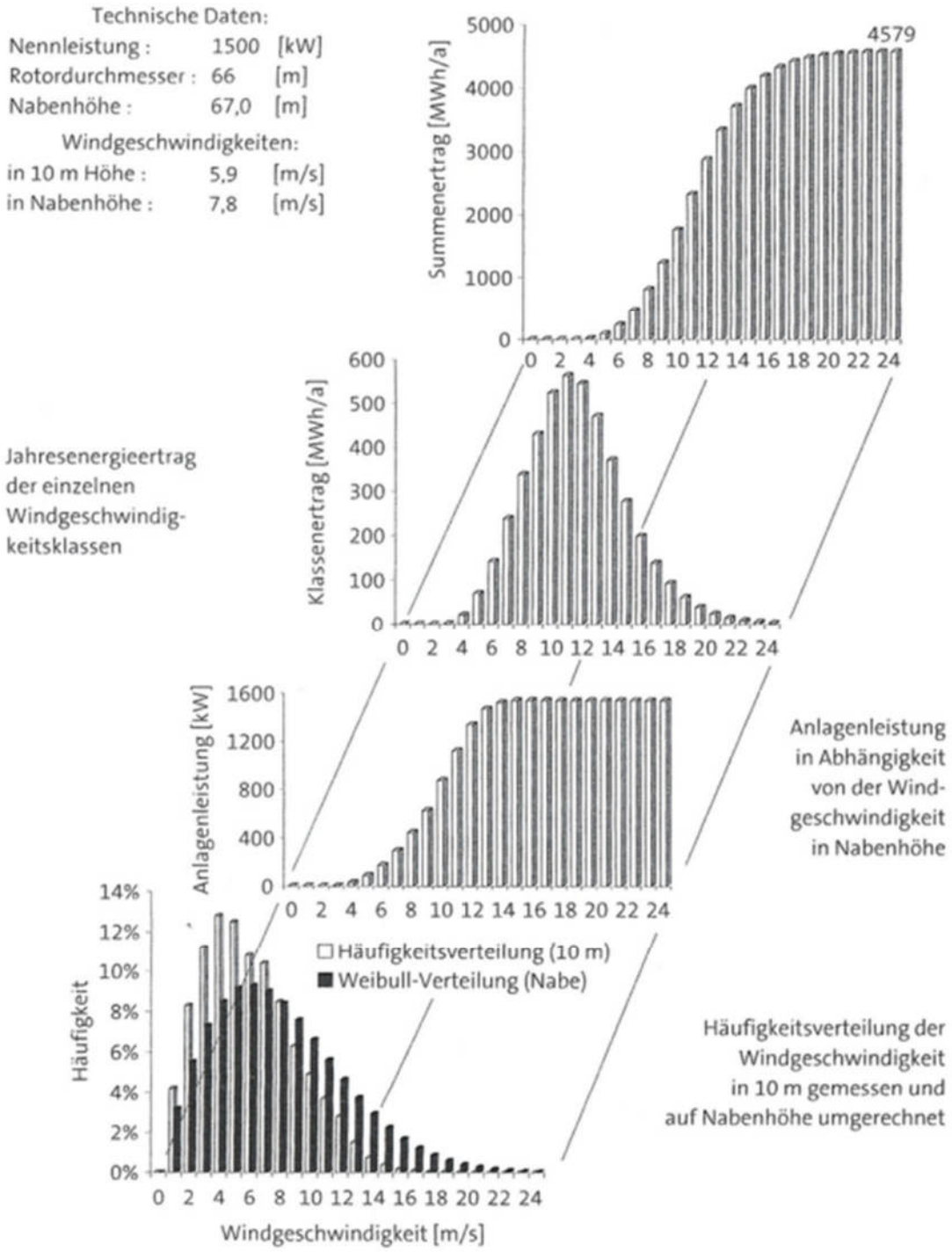


Abbildung 12: Energieertragsberechnung (Heier 2008)

Tabelle 4: Energieertragsberechnung (A-Wert = 7, k-Wert = 2)

Wind-geschwindig-keit [m/s]	Häufigkeit pro Wind-geschwindigkeits-klasse	Anlagenleistung [kW] in Abhängigkeit von Windgeschwindigkeit in Nabenhöhe	Klassenertrag [kWh/a]
2	0,075234	3,0	1.977,14
3	0,101903	48,5	43.294,49
4	0,117783	155,0	159.926,07
5	0,122525	339,0	363.855,47
6	0,117466	627,5	645.698,10
7	0,105108	1.035,5	953.436,30
8	0,088447	1.549,0	1.200.155,87
9	0,070333	2.090,0	1.287.690,37
10	0,053030	2.580,0	1.198.512,20
11	0,038000	2.900,0	965.355,40
12	0,025925	3.000,0	681.312,78
13	0,016862	3.000,0	443.128,64
14	0,010466	3.000,0	275.048,57
15	0,006205	3.000,0	163.057,65
16	0,003515	3.000,0	92.387,28
17	0,001905	3.000,0	50.056,11
18	0,000987	3.000,0	25.945,99
19	0,000490	3.000,0	12.871,07
20	0,000233	3.000,0	6.112,63
21	0,000106	3.000,0	2.779,89
22	0,000046	3.000,0	1.210,92
23	0,000019	3.000,0	505,33
24	0,000008	3.000,0	202,07
25	0,000003	3.000,0	77,43
Summe:	**0,9566**		**8.574.597,78**

5 Datengrundlage

5.1 Datenquellen

Für die Umsetzung des SDSS wurde auf Daten aus unterschiedlichsten Quellen zurückgegriffen. Nachfolgende Tabelle gibt einen Überblick über die bezogenen Rohdaten.

Tabelle 5: Datengrundlage

Datensatz	Datenquelle	Ausdehnung	Datum	Format
Deutscher Wetterdienst (DWD)				
Weibullparameter in 10m und 80m Höhe über Grund (200m Auflösung)	Deutscher Wetterdienst	Gauß-Krüger (Bessel): Hoch min: 5624000 Hoch max: 5675000 Rechts min: 4414000 Rechts max: 4465000	Statistischer Bezugszeitraum: 1981 - 2000	ASCII
Open Street Map (OSM)				
OSM-Daten der Umspannwerke	Open Street Map, Download von www.geofabrik.de	Bundesrepublik Deutschland	Download am 12.11.2013	.pbf
OSM-Daten der Wohn-, Gewerbe- und Industriegebiete	Geofabrik: www.geofabrik.de	Bundesländer Thüringen und Sachsen-Anhalt	Download Thüringen am 10.09.2013 Download Sachsen-Anhalt am 08.11.2013	.shp
OSM-Daten des Straßen- und Eisenbahnnetzes	Geofabrik: www.geofabrik.de	Bundesländer Thüringen und Sachsen-Anhalt	Download Thüringen am 10.09.2013 Download Sachsen-Anhalt am 08.11.2013	.shp
Shuttle Radar Topography Mission (SRTM)				
SRTM-Daten (90m Auflösung)	Consortium for Spatial Information http://srtm.csi.cgiar.org	WGS 84: Latitude min: 50° N Latitude max: 55° N Longitude min: 10° E Longitude max: 15° E	Download am 11.11.2013	GeoTiff (16 Bit)
Schutzgebiete				
FFH-Gebiete	Thüringer Landesanstalt für Umwelt und Geologie	Bundesland Thüringen	Bearb. Stand: 07.12.2012 Stand: 18.01.2005	.shp
Landschaftsschutzgebiete			Bearb. Stand: 21.05.2013 Stand: 21.05.2013	
Naturpark			Bearb. Stand: 15.12.2006 Stand: 07.12.2012	
Naturschutzgebiete			Bearb. Stand: 18.11.2013 Stand: 05.11.2013	
Vogelschutzgebiete			Bearb. Stand: 07.12.2012 Stand: 26.03.2007	

Weibullparameter

Die Weibullparameter basieren auf dem statistischen Windfeldmodel des DWD und wurden, wie in Kapitel 4 beschrieben, für die Berechnung des Energiepotenzials herangezogen. Das statistischen Windfeldmodel des DWD wird von Gerth & Christoffer (1994) ausführlich beschrieben und diskutiert: Insgesamt fließen zur Bestimmung der mittleren Windgeschwindigkeit Daten aus 225 festinstallierten Windmessstationen sowie aus 30 temporären Windmessstationen in die Berechnung ein. Diese Daten wurden hindernisbereinigt und über einen regressionsanalytischen Ansatz mit der Geländehöhe, der geographischen Lage, der topographischen Form sowie der Oberflächenrauigkeit in Beziehung gesetzt. Die in dieser Arbeit vorliegenden Daten wurden für einen Bezugszeitraum von 1981-2000 berechnet. Was die Güte des Windfeldmodels angeht, so beträgt der Korrelationskoeffizient zwischen den gemessenen mittleren Windgeschwindigkeiten an den Stationen und den Modelwerten 0.993, die Streuung beträgt 0.14 m/s. Bei den Weibullparametern erhält man für beide Parameter Korrelationskoeffizienten von mehr als 0.98 und eine mittlere Streuung von 0.09 für den Formparameter bzw. 0.21 für den Skalenfaktor. Bei der Berechnung der Weibullparameter wurden jedoch Daten von nur 162 Messstationen herangezogen, da sämtliche Messstationen, deren Messdauer weniger als 6 Jahre betrug, die Häufigkeitsverteilung der Windgeschwindigkeiten nicht adäquat abbilden.

OSM-Daten

Die Straßendaten sowie Daten der Wohn-, Gewerbe- und Industriegebiete wurden für die Bundesländer Thüringen und Sachsen-Anhalt von der Geofabrik GmbH in aufbereiteter Form als shp-Format bezogen. Ebenfalls konnte dort die pbf-Datei für das Gebiet der BRD heruntergeladen werden.

SRTM-Daten

Die SRTM-Daten stammen vom *Consortium for Spatial Information*. Das bezogene GeoTiff deckt eine Fläche von 5° geographischer Länge und 5° geographischer Breite ab. Im Gegensatz zu ASTER-Daten besitzen SRTM-Daten zwar eine größere Auflösung, weisen aber eine durchschnittlich höhere Genauigkeit auf. Zudem sind ASTER-Daten eher für gebirgige Regionen geeignet (Jacobsen 2010). Aus diesen Gründen kommen die gröber aufgelösten SRTM-Daten in der vorliegenden Arbeit zur Anwendung.

Schutzgebiete

Sämtliche Schutzgebietsdaten wurden von der Thüringer Landesanstalt für Umwelt und Geologie kostenfrei im shp-Format zur Verfügung gestellt. Digitalisiergrundlage bildete die TK 25 (2003) des Thüringer Landesamtes für Vermessung und Geoinformation. Zu beachten ist der Unterschied des Datums zwischen Stand und Bearbeitungsstand (Tab. 5): Während sich der Stand auf die Aktualität hinsichtlich der Ausweisung der Schutzgebiete bezieht, ist mit dem Bearbeitungsstand das Datum der letzten Überarbeitung bereits digital vorhandener Datensätze gemeint.

5.2 Datenaufbereitung

Weibullparameter

Das Tool *ASCII to Raster* diente zur Umwandlung der ASCII-Datensätze der A-Werte in 10 und 80 m Höhe sowie der k-Wert in 80 m Höhe in Rasterdatensätze. Anschließend erfolgte die Definition des Projektionssystems: Die Daten wurden vom DWD im System Gauß-Krüger (Bessel, Zone 4) bereitgestellt, allerdings war diese Projektionsinformation im Datensatz noch nicht enthalten.[5]

Da die A-Werte im Ursprungsdatensatz um den Faktor 10 und die k-Werte um den Faktor 1000 erhöht waren, mussten die Rasterdatensätze durch die entsprechenden Faktoren dividiert werden, um die korrekten Werte der Weibullparameter zu erhalten. Abbildung 13 stellt den Prozess der Datenaufbereitung für den A-Wert exemplarisch dar.

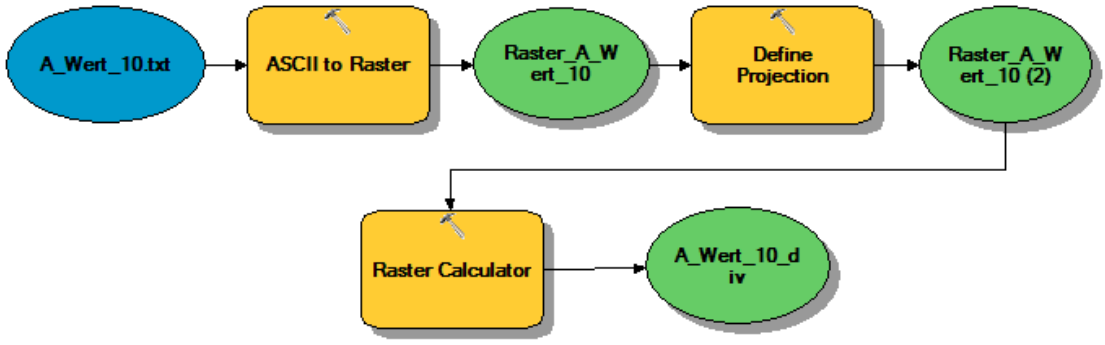


Abbildung 13: Aufbereitung Rohdaten der Weibullparameter

Mittels der aufbereiteten Weibullparameter konnte über den *Raster Calculator*, unter Verwendung von Formel (8), ein neuer Rasterdatensatz, welcher die Bodenrauigkeit wiederspiegelt, berechnet werden. Durch diesen Datensatz war es möglich, unter Verwendung von Formel (9), den A-Wert in 149 m Höhe zu bestimmen.

Die Berechnung des Energieertragsrasters erfolgte über das in Kapitel 4.4 beschriebene Verfahren: Nach der Berechnung der Ertragsraster pro Windgeschwindigkeitsklasse wurden diese addiert. Um die Einheit MWh/a zu erhalten, musste das Ergebnis noch durch 1000 dividiert werden. Zudem wurde das Raster auf eine Pixelgröße von 25 m

[5] Sämtliche in dieser Arbeit zur Anwendung kommenden Daten werden in das System Gauß-Krüger (Bessel, Zone 4) projiziert.

überführt. Formel (1) diente anschließend zur Standardisierung des Ertragsrasters auf Werte zwischen 0 und 1. Da das nicht-standardisierte Energieertragsraster als *Floating-Point-Raster* vorlag, innerhalb des späteren SDSS bzw. *Python-Scripts* jedoch die Funktion *Extract by Attributes* Anwendung findet, welche diesen Pixeltyp nicht verarbeiten kann, erfolgte eine Umwandlung in ein *Integer-Raster*, wobei das *Floating-Point-Raster* zuvor noch mit 0.5 addiert wurde, um korrekt gerundete Werte zu erhalten. Letztlich wurde das Raster noch auf die Ausdehnung des Untersuchungsgebiets zugeschnitten.

Ergebnis des Prozesses der Aufbereitung der Weibullparameter sind somit zwei Rasterdatensätze, die als Eingangsdatensätze für das SDSS zur Verfügung stehen: zum einen ein standardisiertes Energieertragsraster, das für die WLC herangezogen werden kann, zum anderen ein Raster, das die absoluten Energieerträge darstellt und zum Definieren von Ausschlussflächen dient (Abb. 14).

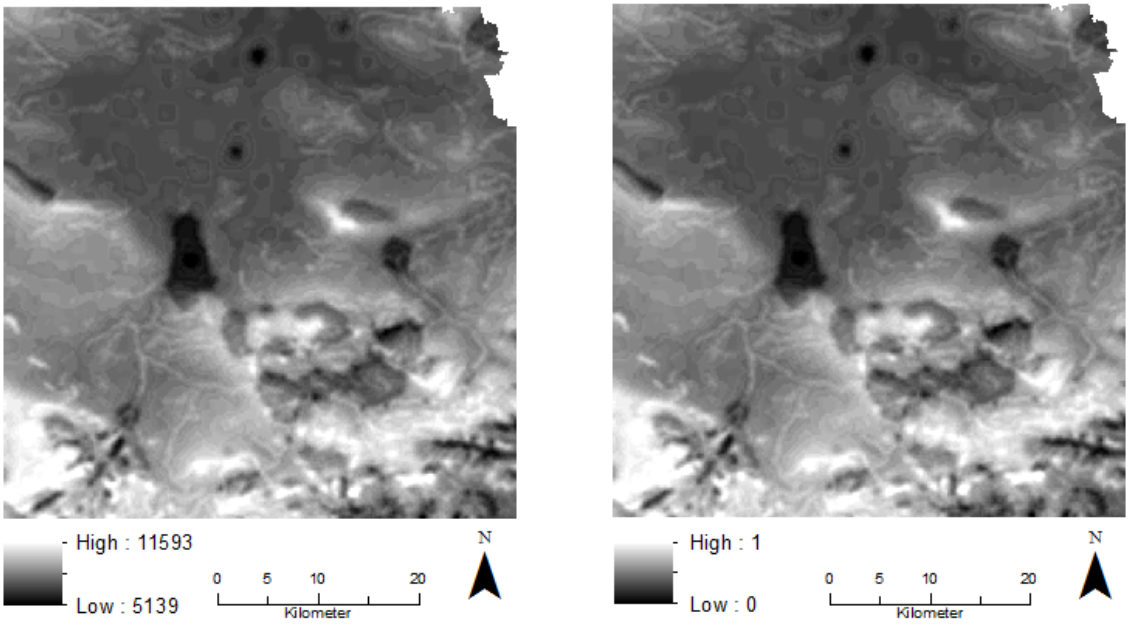


Abbildung 14: Raster Energieertrag

OSM-Daten

- Umspannwerke

  Mittels der Kommandozeilen-Tools *osmconvert* und *osmfilter* wurde die pbf-Datei der BRD in eine osm-Datei umgewandelt und die Umspannwerke herausgefiltert. Die Filterung erfolgte über das *Tag ‚power=station'*. Zwar wird seitens des *OpenStreetMap-Wiki* darauf verwiesen, dass Umspannwerke seit Januar 2008 mit dem *Tag ‚power=substation'* ausgewiesen werden sollten[6], allerdings waren in dem Datensatz erst 161 Umspannwerke mit diesem Tag vorhanden, 117 davon im Raum Berlin. Innerhalb und in der weiteren Umgebung des Untersuchungsgebiets konnte über dieses Attribut kein Umspannwerk ausfindig gemacht werden.

  Das Tool *Load OSM File* der *OpenStreetMap-Toolbox* diente zur Umwandlung der osm-Datei in ein *Feature Dataset*. Für die weitere Verarbeitung wurden nur die *Polygon Features* herangezogen.

  Vor der weiteren Aufbereitung des Datensatzes wurde um das eigentliche Projektgebiet ein Einzugsgebiet definiert (Abb. 15): Um innerhalb des Projektgebiets ein korrektes Ergebnis zu erhalten, ist es notwendig, dessen weitere Umgebung zu betrachten. Gerade am Rand des Untersuchungsraums kann sich das nächstgelegene Umspannwerk außerhalb des zu untersuchenden Gebiets befinden. Die Ausdehnung des Einzugsgebiets beträgt 70 km x 70 km und ist Abbildung 15 zu entnehmen.

[6] http://wiki.openstreetmap.org/wiki/Tag:power%3Dstation (Stand: 01.12.2013, zuletzt geändert 22.10.2013)

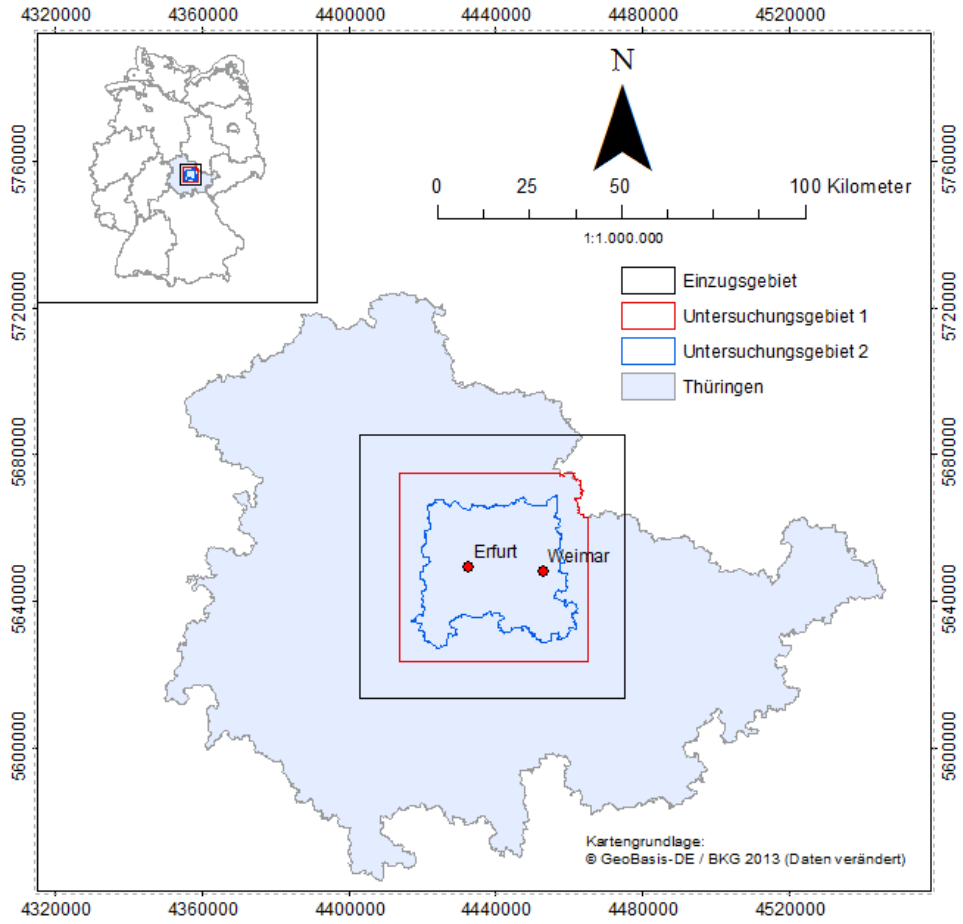


Abbildung 15: Einzugsgebiet

Für die Umspannwerke fand innerhalb des Einzugsgebiets ein Abgleich mit Goo*gle Earth* statt. Es stellte sich heraus, dass teilweise auch kleine Transformatorhäuschen als Umspannwerke gekennzeichnet wurden. Diese Daten wurden manuell herausgefiltert, sodass von den ursprünglich 38 im Einzugsgebiet befindlichen *Features* 30 zur weiteren Bearbeitung zur Verfügung standen. Mit Hilfe dieser Daten konnte eine euklidische Distanzoberfläche berechnet werden, welche anschließend auf die Ausdehnung des Projektgebiets zugeschnitten wurde. Gemäß dem Vorgehen bei den Energieertragsrastern wurde auch dieser Rasterdatensatz standardisiert bzw. in ein *Integer*-Raster umgewandelt und auf eine Pixelgröße von 25 m überführt (Abb. 16).

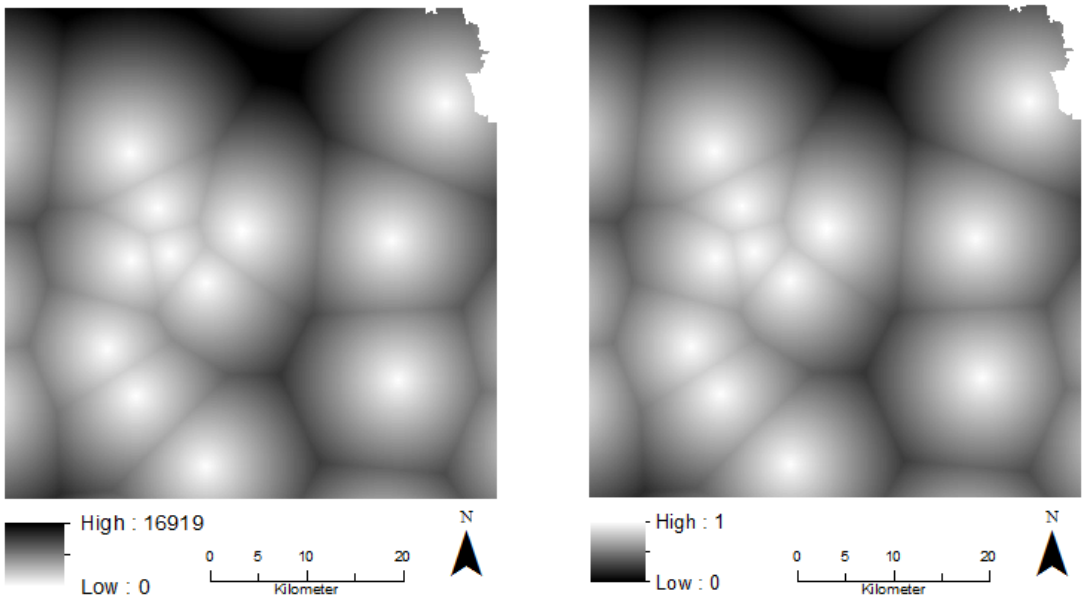


Abbildung 16: Distanzraster Umspannwerke

- Straßennetz, Wohn-, Gewerbe- und Industriegebiete

  Straßen wurden in Anlehnung an die von Bergmann & Höfle (2013) verwendeten Attribute bzw. OSM-*Tags* jeweils für die Bundesländer Thüringen und Sachsen-Anhalt gefiltert, mittels *Merge* zusammengeführt, auf das Einzugsgebiet zugeschnitten und letztlich wie bei den Umspannwerken ein Distanzraster sowie ein standardisiertes Raster mit 25 m Pixelgröße und identer Ausdehnung erzeugt (Abb. 17).

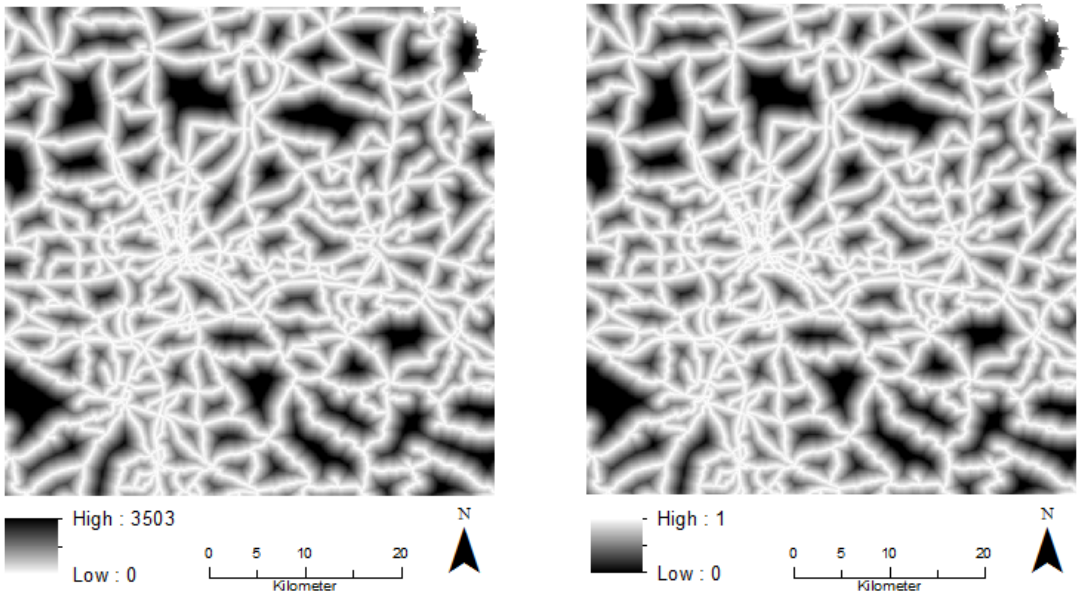


Abbildung 17: Distanzraster Straßennetz

Ebenso fand eine Umwandlung der einzelnen Straßenkategorien sowie des Schienennetzes in eigene *Feature Classes* statt. So ist es später möglich, innerhalb des SDSS um die einzelnen Straßenkategorien Abstandsflächen zu definieren. Selbiges gilt für die Wohn-, Gewerbe- und Industriegebiete. Tabelle 7 ist zu entnehmen, nach welchen Attributen die einzelnen Nutzungen gefiltert wurden.

Tabelle 6: OSM-*Tags* verschiedener Nutzungstypen

Nutzung	OSM-*tag*
Wohnbebauung	*landuse = residential*
Gewerbegebiete	*landuse = commercial*
Industriegebiete	*landuse = industrial*
Eisenbahn	*railway = rail*
Autobahn	*highway = motorway, highway = motorway_link*
Bundesstraße	*highway = trunk, highway = trunk_link, highway = primary, highway = primary_link*
Landesstraße	*highway = secondary*
Kreisstraße	*highway = tertiary*

SRTM-Daten

Nachdem der Datensatz umprojiziert und zugeschnitten war, wurde das Raster mittels *Focal Statistics* geglättet und daraus ein Hangneigungsraster erzeugt. Das Hangneigungsraster gibt die Steigung in Prozent an. Wie bei den vorherigen Datensätzen wurde zudem ein standardisiertes Raster erzeugt (Abb. 18).

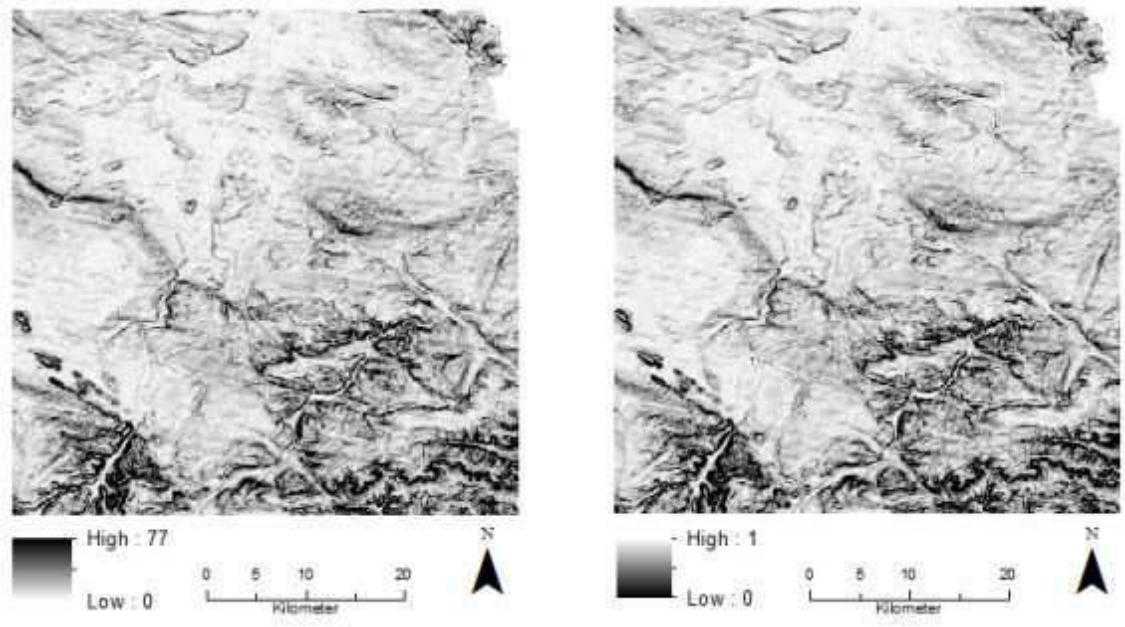


Abbildung 18: Raster Hangneigung

Schutzgebiete

Die Daten der Schutzgebiete wurden lediglich umprojiziert. Eine weitere Datenaufbereitung fand nicht statt.

5.3 Kritik an den Daten

Energieertragsraster

Das Energieertragsraster ist mit erheblichen Unsicherheiten behaftet und kann daher nur als grobe Orientierung herangezogen werden. Standortspezifische Windmessungen oder Ertragsprognosen können dadurch nicht ersetzt werden:

Die bereits in den Ursprungsdaten enthaltenen Unsicherheiten (Kapitel 5.1) werden durch die Extrapolation des A-Wertes (Kapitel 4.3) sowie durch die Unkenntnis über den wahren k-Wert weiter erhöht. Zudem ist die verwendete Leistungskennlinie auf eine Standardluftdichte von 1,225 kg/m^3 ausgelegt, was das Ergebnis der Energieertragsberechnung ebenfalls verfälscht. Letztlich kommt es bei der Überführung des Rasters von 200 m Pixelgröße auf 25 m Pixelgröße zu weiteren Verzerrungen.

OSM-Daten

Was die verwendeten und verarbeiteten OSM-Daten angeht, so sei auf die allgemeine Kritik verwiesen, die *Volunteered Geographic Information* (VGI) entgegengebracht wird: Zum einen kann die Vollständigkeit der Daten nicht gewährleistet werden. Zum anderen ist deren richtige Kategorisierung und Verortung nicht garantiert. Bergmann & Höfle (2013) weisen jedoch darauf hin, dass OSM-Daten eine ausreichende Qualität aufweisen, um potenzielle Standorte für die Windenergienutzung ausfindig zu machen. Unterschiede gebe es zwar hinsichtlich Grenzverlauf und Flächengröße, die räumliche Lage der Flächen sei aber größtenteils ident.

SRTM-Daten

Kritik an den SRTM-Daten betrifft vor allem deren grobe Auflösung sowie die Tatsache, dass das SRTM-Höhenmodell nicht die Geländeoberfläche wiederspiegelt, sondern von den darauf befindlichen Objekten (Wald, Bewuchs, Häuser etc.) beeinflusst wird und somit ein digitales Oberflächenmodell darstellt (Jacobsen 2010). Folglich ist eine Ableitung der Hangneigung aus diesen Daten mit entsprechenden Unsicherheiten verbunden.

Schutzgebiete

Da es sich bei den Daten der Schutzgebiete um amtliche Geodaten handelt, kann die räumliche Verortung der Gebiete als präzise angenommen werden. Selbiges gilt für die Vollständigkeit des Datensatzes. Es ist davon auszugehen, dass sich die Daten auf dem aktuellsten Stand befinden und sämtliche ausgewiesene Schutzgebiete beinhalten.

6 SDSS-Konzeption und Umsetzung

6.1 Anwendungsfall

Zur Definition der Benutzeranforderungen dient ein UML-Anwendungsfall-Diagramm, welches vereinfacht die Möglichkeiten des SDSS-Anwenders darstellt. Die darin ersichtlichen Anwendungsfälle werden im Folgenden spezifiziert.

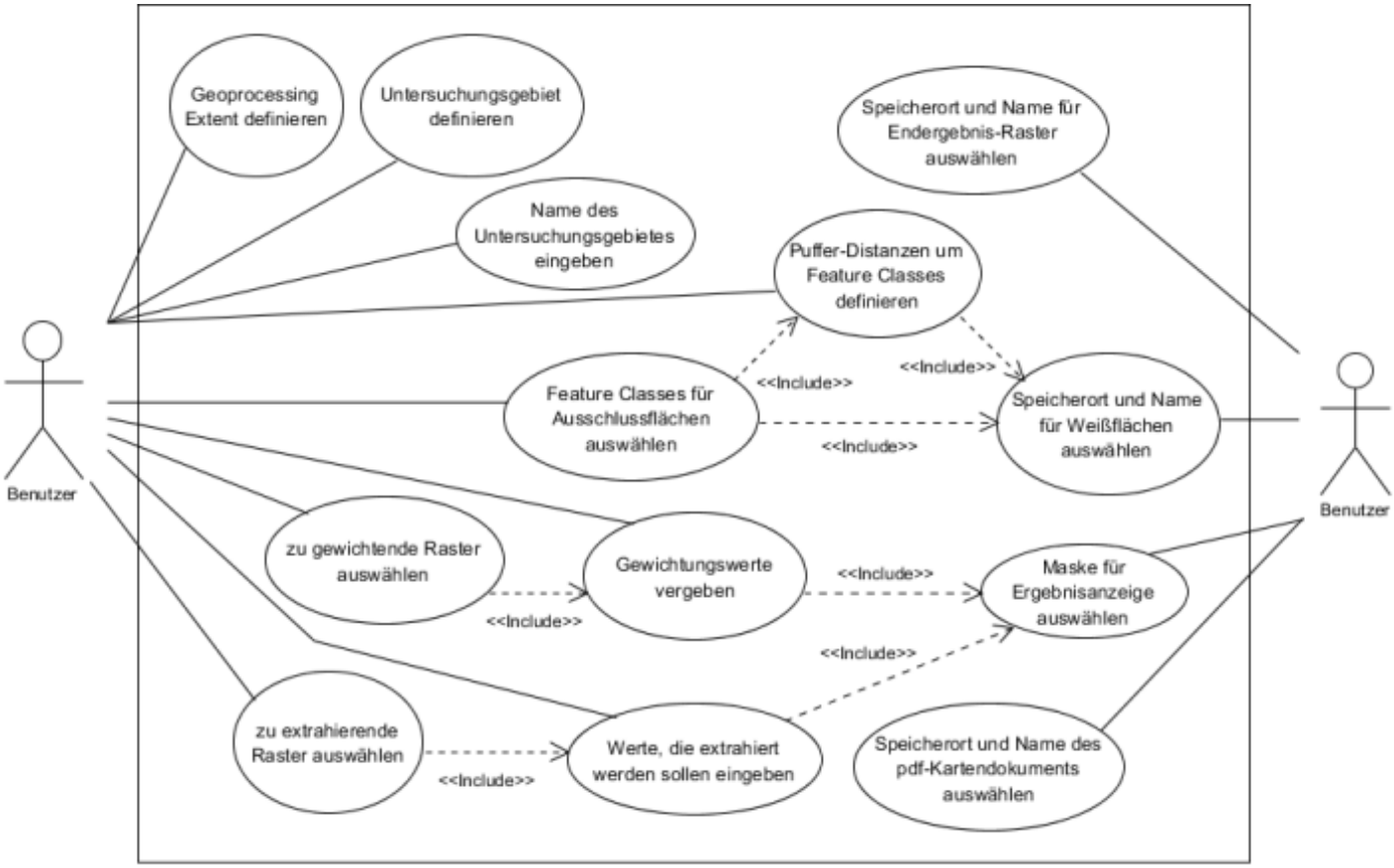


Abbildung 19: Anwendungsfall-Diagramm

Geoprocessing Extent definieren

Der *Geoprocessing Extent* wird über die Angabe eines Datensatzes definiert. Die Ausdehnung des Datensatzes entspricht somit der Größe des zu berechnenden Gebiets. Die Angabe ist obligatorisch. Bewusst wird darauf verzichtet, den *Geoprocessing Extent* über die Ausdehnung des Untersuchungsgebiets zu definieren, da das zu entwickelnde Tool auch die Generierung von Abstandsflächen ermöglichen soll: Würde der *Geoprocessing Extent* dem Untersuchungsgebiet entsprechen, könnten gerade an dessen inneren Rand Flächen als geeignet ausgewiesen werden, obwohl definierte Ab-

stände zu Objekten außerhalb des Untersuchungsraums nicht eingehalten werden. Um dies zu vermeiden, muss der *Geoprocessing Extent* grundsätzlich eine größere Ausdehnung aufweisen als das zu untersuchende Gebiet.

Untersuchungsgebiet

Der Untersuchungsraum wird über die Angabe einer *Polygon Feature Class* bestimmt. Es muss sich um eine *Polygon Feature Class* handeln, da das Ergebnis zum einen auf dieses Gebiet zugeschnitten wird. Zum anderen dient es zur Darstellung des Untersuchungsgebiets in der finalen Karte. Die Angabe des Untersuchungsgebiets ist somit obligatorisch.

Name des Untersuchungsgebiets

Über ein Textfeld wird der Name des Untersuchungsraums angegeben. Der angegebene Name dient in der Ergebniskarte als Titel. Die Eingabe ist ebenfalls obligatorisch.

Zu gewichtende Raster/Gewichtungswerte

Hier können die in Kapitel 5.2 standardisierten Rasterdatensätze für die WLC-Berechnung ausgewählt werden. Da das SDSS für die Flächenbewertung gedacht ist, ist die Eingabe von mindestens zwei Rasterdatensätzen obligatorisch. Für die ausgewählten Rasterdatensätze müssen anschließend noch die entsprechenden Gewichtungswerte vergeben werden.

Zu extrahierende Raster/zu extrahierende Werte

Optional können die nicht standardisierten bzw. die Raster der Absolutwerte aus Kapitel 5.2 ausgewählt werden. Anschließend müssen für jeden angegebenen Datensatz noch jene Werte definiert werden, welche in die Berechnung einfließen sollen. Im Endergebnis sind dann nur Gebiete enthalten, welche die angegebenen Werte aufweisen. Wird mehr als ein Datensatz extrahiert, erfolgt eine Verknüpfung über eine UND-Bedingung: zum Bespiel alle Flächen, bei denen die Hangneigung kleiner als 20% und der Ertrag größer als 8000 MWh/a ist.

Feature Classes für Ausschlussflächen/Puffer-Distanzen/Speicherort für Weißflächen

Diese optionale Eingabemöglichkeit dient zum Generieren von sog. Weißflächen. Hierbei handelt es sich um Flächen, welche formale Abstandskriterien, wie sie beispielsweise in den Winderlassen der Bundesländer definiert sind, erfüllen. Beliebig viele *Feature Classes* können mit Puffer-Distanzen versehen und somit Ausschlussflächen bestimmt werden. Die Ergebnisdarstellung beschränkt sich auf die ermittelten Weißflächen. Das Ergebnis dieser Analyse ist ein Rasterdatensatz, der abgespeichert werden muss.

Maske für Ergebnisanzeige

Wird keine Weißflächenanalyse durchgeführt, muss hier ein Datensatz angegeben werden, welcher als Maske für die Ergebnisdarstellung dient. Ist bereits eine Maske für die Ergebnisanzeige vorhanden, kann auf das berechnen von Ausschluss- und Abstandsflächen verzichtet werden. Ein schon generiertes Weißflächen- oder Ergebnisraster kann somit zudem einer weiteren Flächenbewertung oder Sensitivitätsanalyse hinsichtlich der Gewichtung von Kriterien unterzogen werden, ohne jedes Mal den kompletten Rechenprozess erneut durchführen zu müssen.

Speicherort Endergebnisraster/Speicherort PDF-Kartendokument

Für das Ergebnisraster der gesamten Analyse sowie für das PDF-Kartendokument, welches ebenfalls generiert wird, muss letztlich noch der Speicherort sowie der gewünschte Dateiname vergeben werden. Das PDF-Dokument soll neben der Karte ebenfalls Informationen zu den Flächeneigenschaften enthalten, welche aus der Benutzereingabe generiert werden: Die vergebenen Gewichtungswerte sowie die in die Berechnung eingeflossenen Werte der extrahierten Raster sollen dem Dokument entnommen werden können.

In Tabelle 7 sind nochmals alle für das SDSS vorgesehenen Eingabeparameter sowie deren geplante Eigenschaften dargestellt. Wie in Kapitel 6.4 noch zu zeigen sein wird, ließen sich bei der Tool-Gestaltung nicht alle Parameter wie geplant umsetzen.

Tabelle 7: Spezifikation der Eingabeparameter

Parameter Nr.	Parameter	*Data Type*	*Direction*	*Type*	*Multi Value*
1	*Geoprocessing Extent*	*Dataset*	*Input*	*Required*	*no*
2	Untersuchungs - gebiet	*Feature Class*	*Input*	*Required*	*no*
3	Name des Untersuchungsgebiets	*String*	*Input*	*Required*	*no*
4	zu gewichtende *Raster Datasets*	*Raster Dataset*	*Input*	*Required* (mind. 2)	*yes*
5	Gewichtungswerte	*Double*	*Input*	*Required*	*yes*
6	zu extrahierende Raster	*Raster Dataset*	*Input*	*Optional*	*yes*
7	Einschlusswerte der zu extrahierenden Raster	*String*	*Input*	*Required* falls Parameter Nr.7 ausgewählt wurde	*yes*
8	Zu puffernde *Features* für Weißflächenanalyse	*Feature Class*	*Input*	*Optional*	*yes*
9	Puffer-Distanzen	*Long*	*Input*	*Required* falls Parameter Nr.8 ausgewählt wurde	*yes*
10	Speicherort und Name Weißflächen	*Raster Dataset*	*Output*	*Required* falls Parameter Nr.8 ausgewählt wurde	*no*
11	Maske für Ergebnisanzeige	*Dataset*	*Input*	*Required* falls Parameter Nr.8 nicht ausgewählt wurde	*no*
12	Speicherort und Name Ergebnisraster	*Raster Dataset*	*Output*	*Required*	*no*
13	Speicherort und Name PDF-Kartendokument	PDF	*Output*	*Required*	*no*

6.2 Aktivitätsdiagramme

Die folgenden Aktivitätsdiagramme stellen den Programmablauf, wie er im *Python-Script* umgesetzt wurde, vereinfacht dar. Abhängig von der Anzahl der vom Nutzer ausgewählten optionalen Eingabeparameter ergeben sich unterschiedliche Prozessabläufe. Die Abbildungen 20-23 (Fall 1-4) zeigen jeweils den Prozess der Datenverarbeitung für die möglichen Kombinationen der Parameter auf. Abbildung 24 beschreibt den Prozess der Kartenerstellung. Für eine detaillierte Dokumentation sei hier auf das Skript im Anhang verwiesen.

Fall 1: Durchführung WLC

Fall 1 stellt den unkompliziertesten Programmablauf dar. Die vom Nutzer ausgewählten Kriterien-Raster werden per WLC kombiniert, anschließend auf die angegebene Eingabemaske und letztlich auf das Untersuchungsgebiet zugeschnitten. Bei diesem Ablauf wird von keinem optionalen Parameter Gebrauch gemacht.

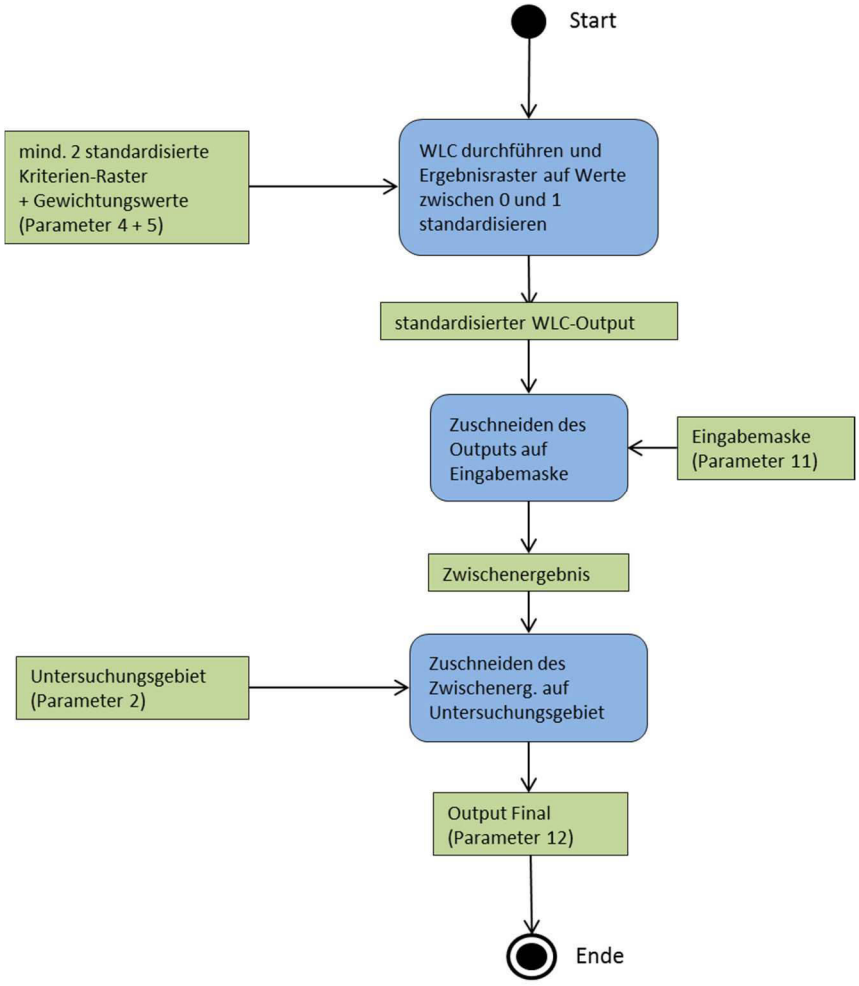


Abbildung 20: Aktivitätsdiagramm ‚Durchführung WLC'

Fall 2: Durchführung WLC und Generieren von Abstands- bzw. Weißflächen

In diesem Fall werden neben der Durchführung der WLC auch Abstandsflächen definiert. Wurde aus mehr als einer *Feature Class* ein Distanzraster erzeugt, kommt es zu einer *Boolean-Overlay*-Verknüpfung. Die erzeugten Abstandsflächen werden schließlich auf das Untersuchungsgebiet zugeschnitten. Diese Ausgabe dient als Maske für den WLC-Output.

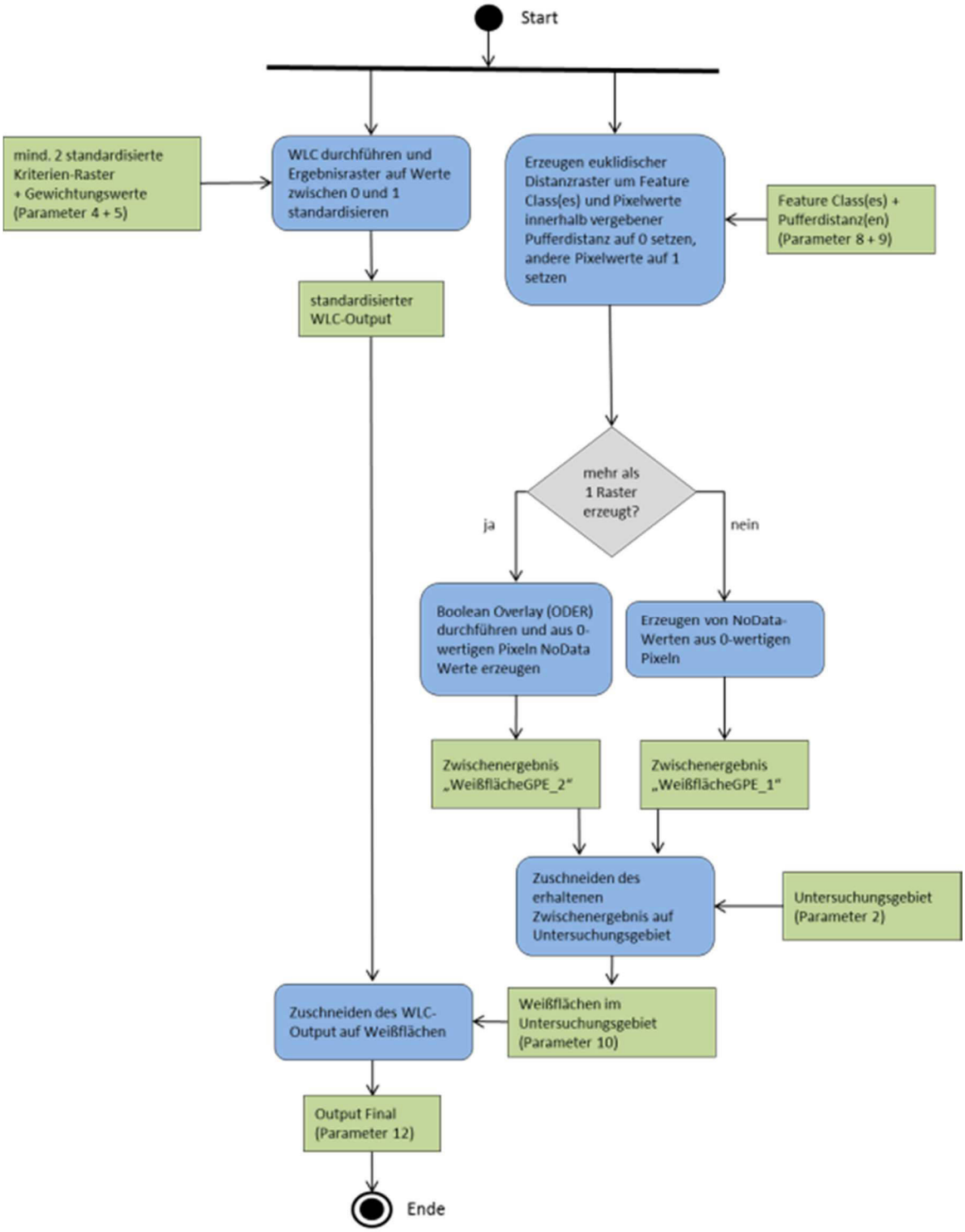


Abbildung 21: Aktivitätsdiagramm ‚WLC und Abstandsflächen'

Fall 3: Durchführung WLC und Extraktion von Rasterwerten

Im dritten Fall werden keine Abstandsflächen erzeugt, sondern Rasterwerte definiert, welche in die Berechnung einfließen sollen. Wurde mehr als ein Raster definiert, kommt es auch hier zu einer *Boolean-Overlay*-Verknüpfung. Das Ergebnis der WLC wird auf das Ergebnis der Rasterextraktion, die Eingabemaske und das Untersuchungsgebiet zugeschnitten.

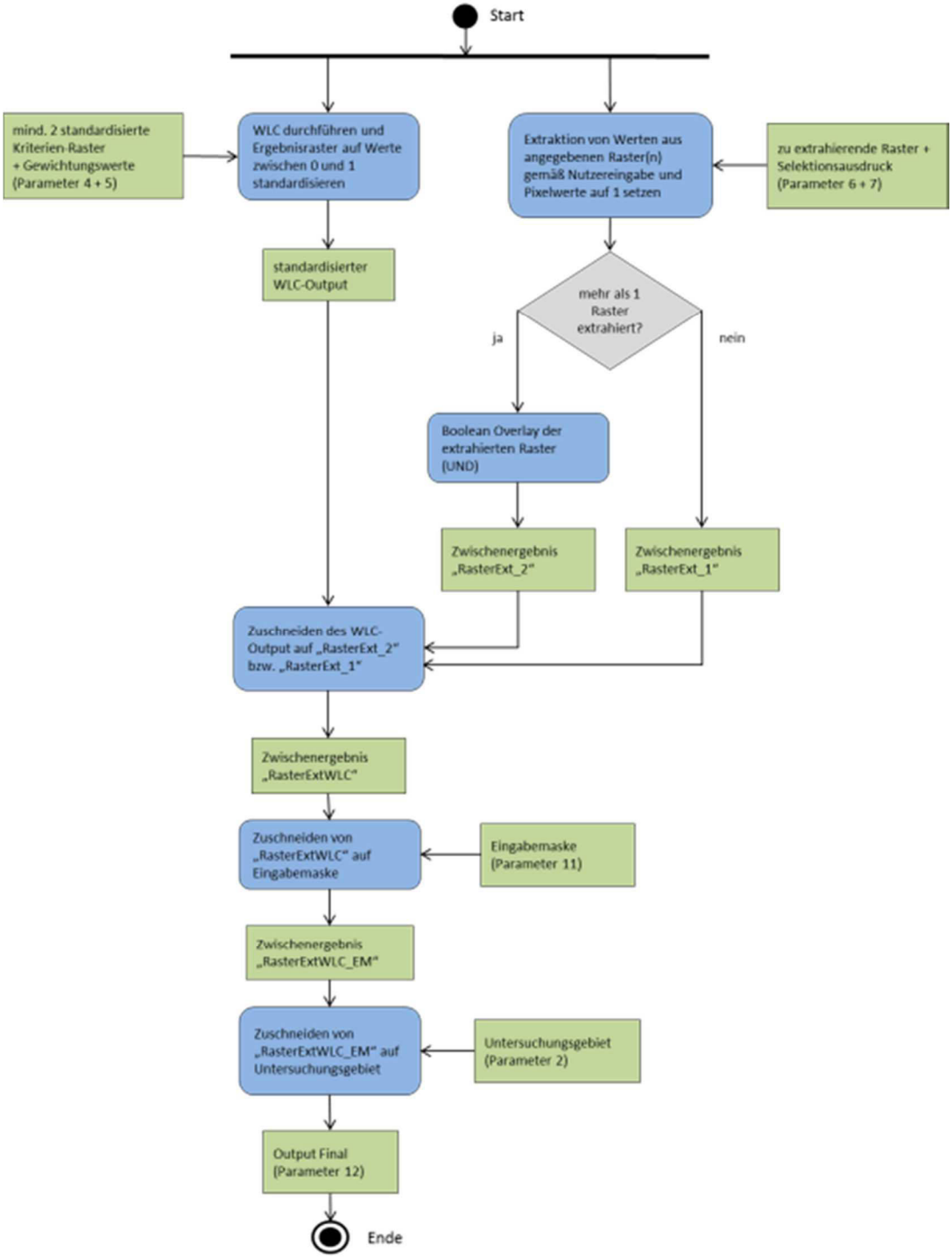


Abbildung 22: Aktivitätsdiagramm ‚WLC und Rasterextraktion'

Fall 4: Durchführung WLC, Rasterextraktion und Generieren von Abstandsflächen

Dieser Fall ist eine Kombination der ersten drei Fälle. Es werden sowohl Rasterwerte extrahiert, als auch Weißflächen erzeugt. Der Output der WLC wird anschließend wieder entsprechend zugeschnitten.

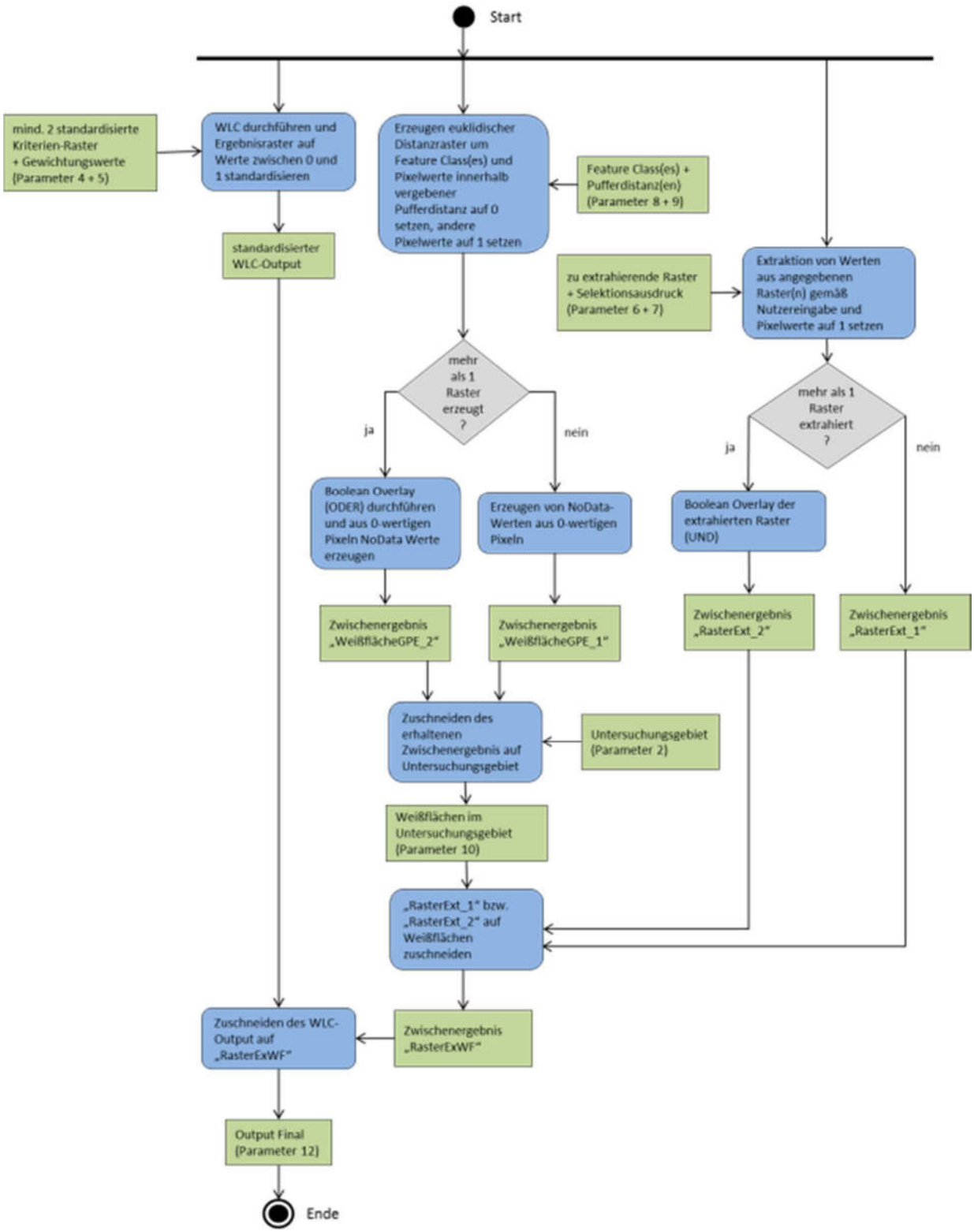


Abbildung 23: Aktivitätsdiagramm ‚WLC, Rasterextraktion und Abstandsflächen'

Fall 5: Prozess der Kartenerstellung

Zur Erstellung der Karte ist es notwendig eine mxd-Vorlage zu gestalten, in welche die Ergebnisse eingefügt werden. Die mxd-Vorlage kann Anhang B entnommen werden. Die verwendete Hintergrundkarte stammt von *Google Maps* und wurde mittels der Software *Ultimate Map Grabber* erstellt. Auf die Verwendung eines *Web Map Service* (WMS) als Hintergrundkarte wurde hier bewusst verzichtet. Diese werden häufig erst ab einer bestimmten Maßstabsebene angezeigt und weisen oft lange Ladezeiten auf, was dazu führen kann, dass die Hintergrundkarte im finalen PDF-Dokument nicht dargestellt wird. Neben der mxd-Vorlage müssen zudem *Symbology-Layer* erzeugt werden, die die Darstellung der Ergebnisse in der Karte definieren.

Was den *Script-Code* zur Kartenerstellung betrifft, so wird zu Beginn der *Layer* des Untersuchungsgebiets und das Ergebnisraster der Kartenvorlage hinzugefügt und diesen die entsprechende Symbologie zugewiesen. Anschließend wird der vergebene Titel aus Parameter 3 dem Titelelement übergeben und, falls nötig, an dessen Größe angepasst. Zudem wird der *Dataframe* auf die Ausdehnung des Ergebnisrasters gesetzt. Ebenso wird eine Legende der mxd-Vorlage zugewiesen. Des Weiteren werden die Namen der verwendeten Kriterien-Raster sowie die vergebenen Gewichtungswerte der Vorlage als Text hinzugefügt. Falls der Nutzer eine Rasterextraktion durchgeführt hat, werden zudem die Namen der extrahierten Raster und die extrahierten Werte in die Kartenvorlage eingefügt. Schließlich wird das mxd-Dokument als PDF exportiert.

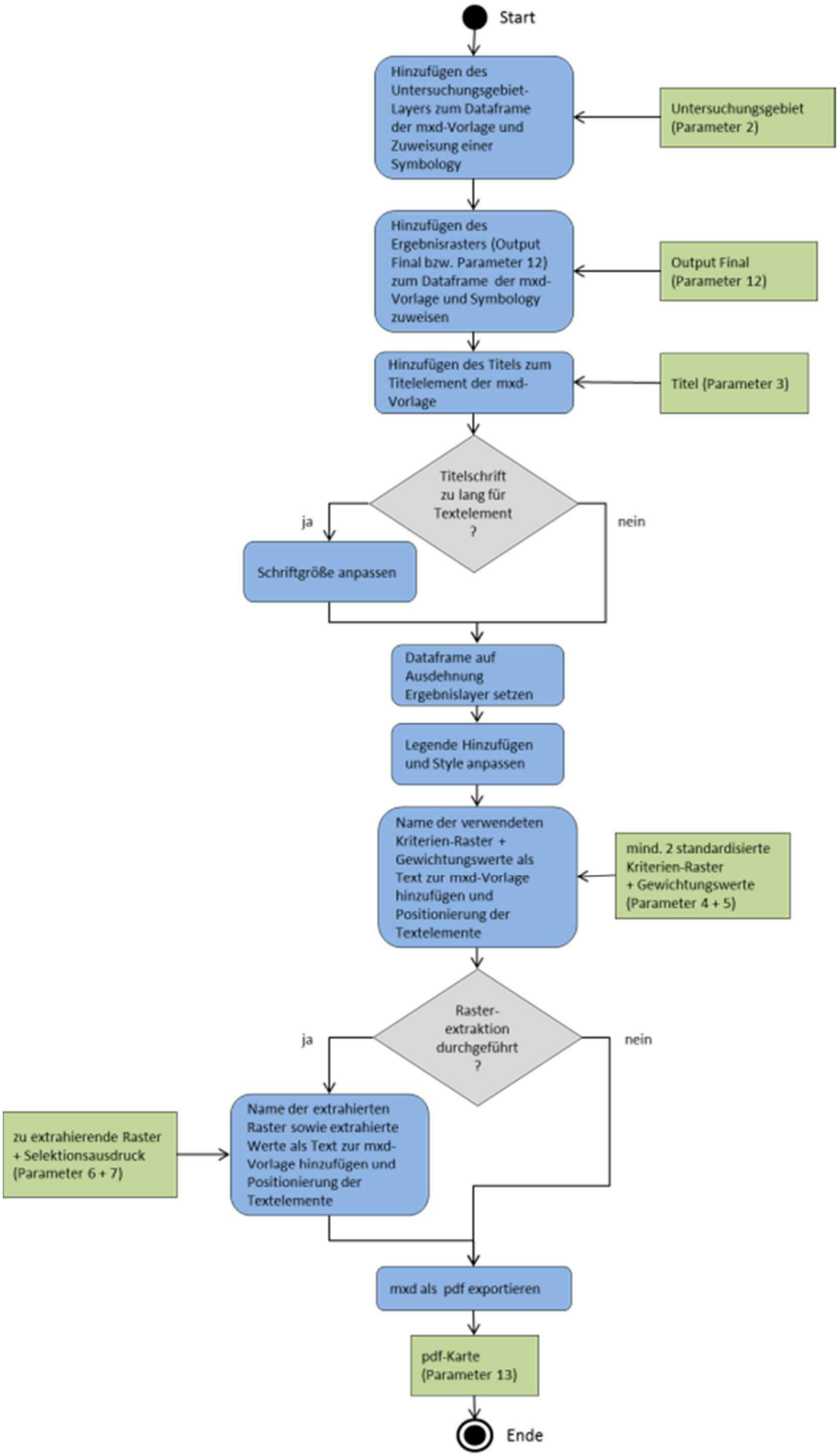


Abbildung 24: Aktivitätsdiagramm ‚Kartenerstellung'

6.3 Datenorganisation

Sämtliche Eingangsdaten des SDSS sind in zwei *Geodatabases* organisiert. Als weiterer Input sind die *Symbology-Layer* für das Ergebnisraster und das Untersuchungsgebiet sowie die mxd-Vorlage zu erwähnen. Die Verarbeitung des Inputs wird durch das *Python-Script* definiert (Kapitel 6.2). Ebenfalls ist ein Zwischenspeicher vorhanden, in welchen Daten, die zur Generierung des Outputs nötig sind, geschrieben werden. Als Zwischenspeicher wird der von *ArcGIS* bereitgestellte *In-Memory-Workspace* verwendet, da das Schreiben von Daten in diesen eine schnellere Datenverarbeitung ermöglicht als das Schreiben in andere Formate (ESRI 2013a). Der Ausgabeort der Ergebnisraster kann vom Benutzer frei ausgewählt werden, allerdings steht hierfür auch eine eigens angelegte *Geodatabase* zur Verfügung.

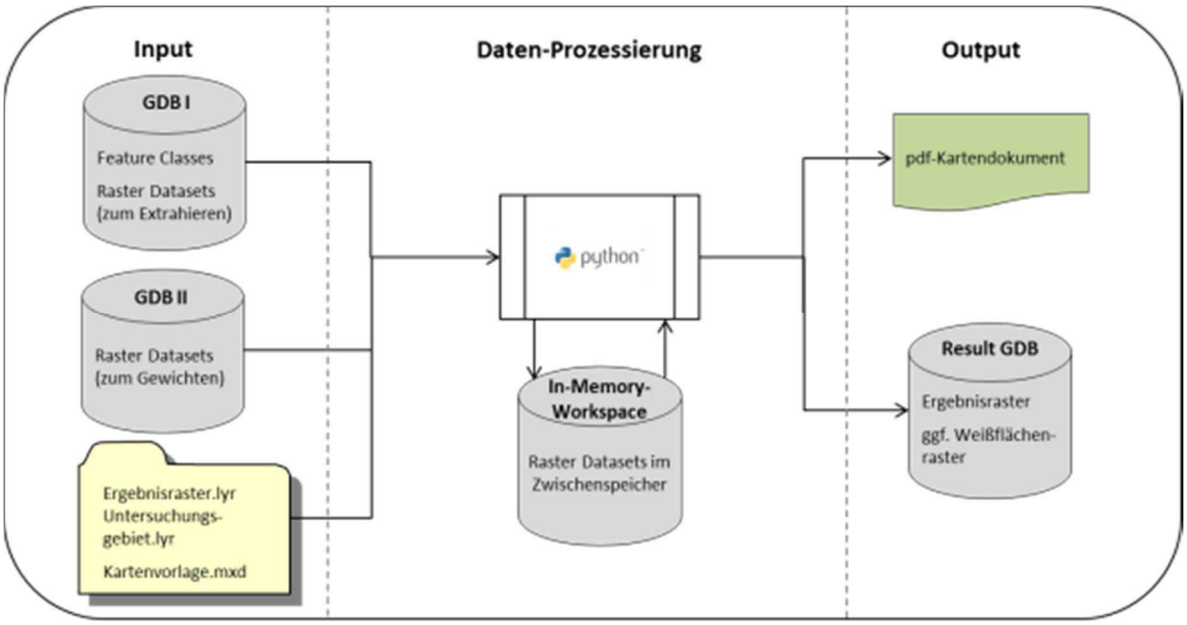


Abbildung 25: Aufbau SDSS

6.4 *Graphical User Interface*

Das entwickelte *Python-Script* wurde innerhalb von *ArcGIS 10.1* als *Script-Tool*, mit den in Kapitel 6.1 definierten Eingabeparametern umgesetzt. Das *Graphical User Interface* (GUI) ist dabei von der Software vorgegeben. Probleme bei der Umsetzung des GUI gab es bei Parameter 7. Bei der Ausweisung dieses Parameters mit dem Datentyp *String* bietet das GUI aus nicht geklärten Gründen für *Multi Value Parameter* keine Möglichkeit eine Anweisung zu übergeben. Das dazugehörige Feld ist nicht verfügbar (Abb. 26)[7]. Um das geplante Tool dennoch umsetzen zu können, wurde Parameter 7 mit dem Datentyp *Any Value* ausgewiesen. Dies ermöglicht die Eingabe eines oder mehrere Selektionsausdrücke, welche die Einschlusswerte definieren. Irreführend ist nun allerdings, dass neben der Eingabezeile auftauchende Ordnersymbol, welches suggeriert, man müsse nach einem entsprechenden Datensatz suchen (Abb. 28). Des Weiteren konnte für den Parameter 13 nicht der Datentyp PDF definiert werden, da dieser für die Tool-Umsetzung nicht zur Auswahl stand. Hier wurde ebenfalls der Datentyp *Any Value* vergeben.

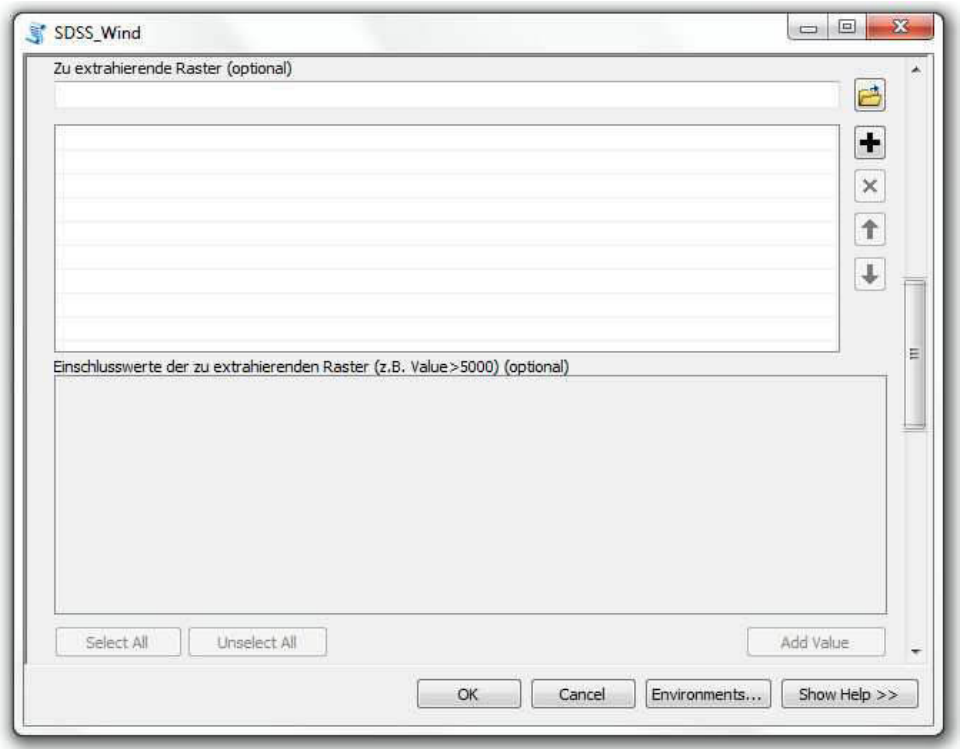


Abbildung 26: Fehler bei Umsetzung des GUI

[7]Zu dieser Problematik sei auch auf eine Forendiskussion hingewiesen: http://gis.stackexchange.com/questions/17821/unable-to-provide-input-for-a-multivalue-parameter-script-tool (Stand: 25.04.2014)

Das letztlich umgesetzte GUI ist den Abbildungen 27-29 zu entnehmen. Kritisch anzumerken ist hier, die mangelnde Möglichkeit das Tool-Verhalten zu steuern. So mussten die Parameter 7, 9, 10 und 11 als optional ausgewiesen werden, obwohl dies nur bedingt zutrifft: Diese Parameter werden in Abhängigkeit anderer Parameter erforderlich (Tab. 7). Da das GUI jedoch statisch ist, erscheinen diese immer als optional. Durch das Anpassen der Methode *updateParameters* innerhalb der *Tool Validator Class* war es möglich die genannten Parameter in Abhängigkeit anderer Eingaben zu aktivieren bzw. zu deaktivieren. Zu erkennen ist dies beispielsweise in Abbildung 29 an den grau eingefärbten Feld ‚Puffer-Distanzen'. Dieses Feld wird erst freigegeben, sobald ein Wert für das vorangehende Feld ‚Zu puffernde Features für Weißflächenanalyse' ausgewählt wurde.

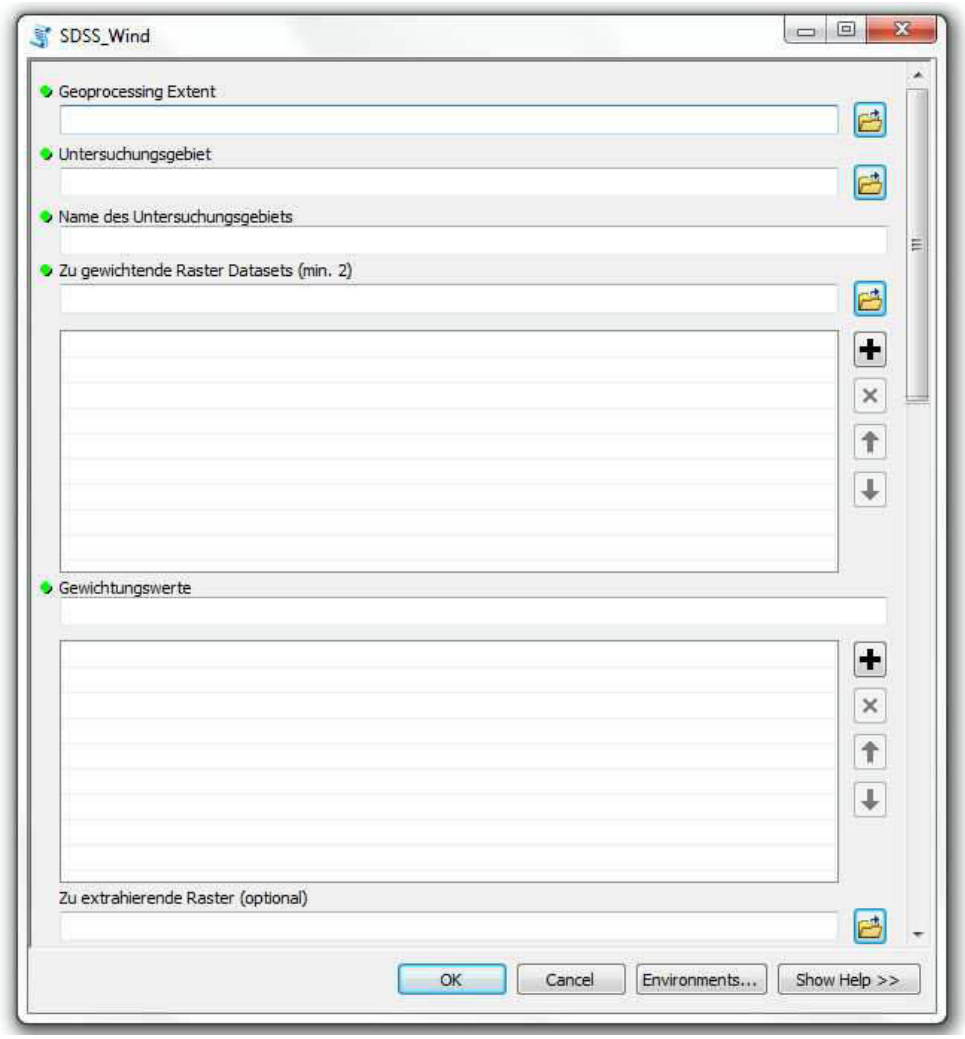


Abbildung 27: GUI Teil 1

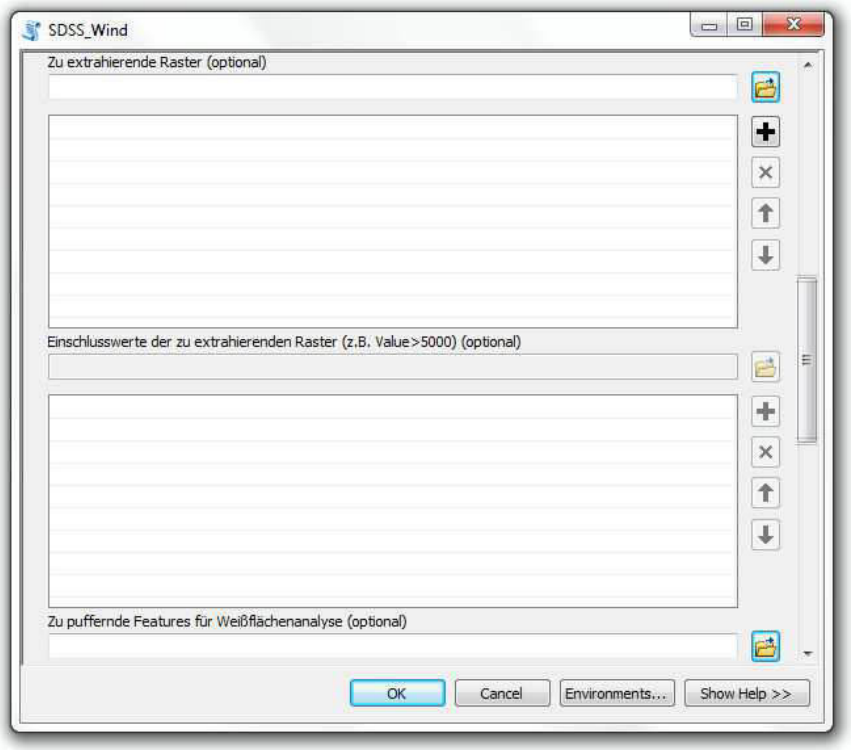


Abbildung 28: GUI Teil 2

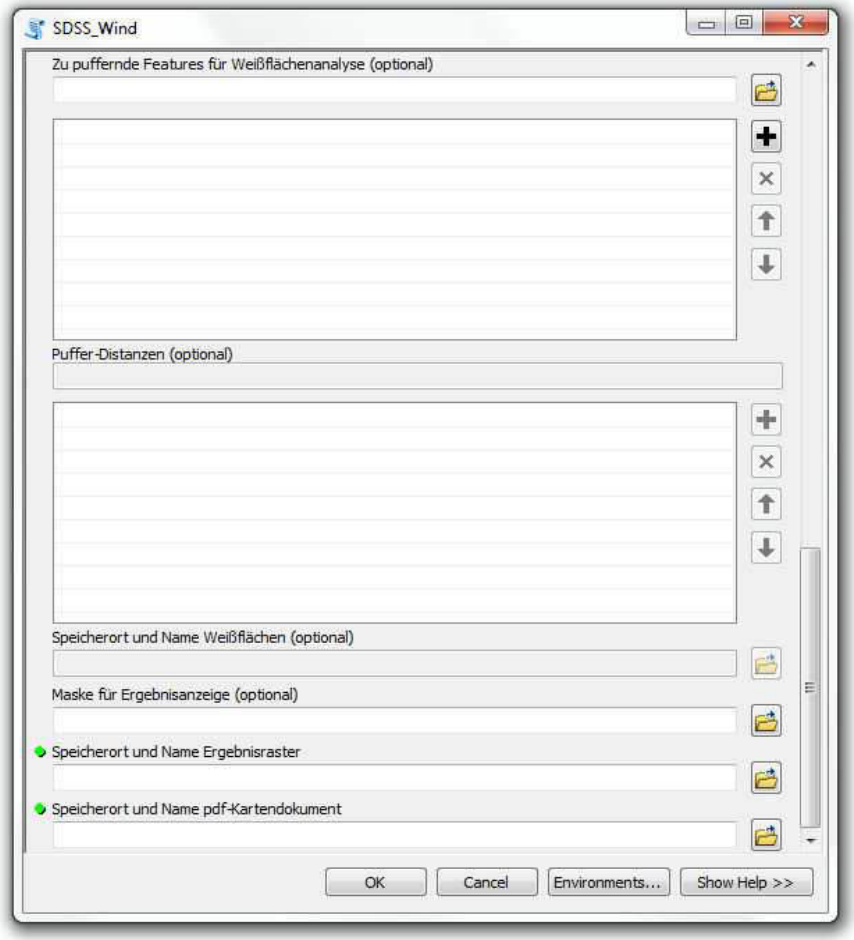


Abbildung 29: GUI Teil 3

7 Ergebnisse

7.1 Plausibilitätsprüfung

Im Folgenden wird der Output des SDSS auf Plausibilität geprüft. Hierzu werden verschiedene Szenarien durchgerechnet und das jeweilige Ergebnis mit den Eingabeparametern verglichen. In Tabelle 8 sind die vergebenen Parameterwerte der verschiedenen Szenarien aufgelistet. Die Abbildungen 30-34 stellen die dazugehörigen Ergebnisse dar. Die dazugehörigen PDF-Kartendokumente können Anhang C-G entnommen werden. Die Dokumente enthalten unterschiedliche Kartentitel, was auf die Variation bei Parameter 2 zurückzuführen.

Tabelle 8: Szenarien für Plausibilitätsprüfung

	Gewichtungswerte	**Abstandsflächen bzw. Maske für Ergebnisanzeige**	**extrahierte Rasterwerte**	**Untersuchungs-gebiet**
Szenario 1	Raster Ertrag: 50 Raster Umspannwerke: 50	**Ausschlussflächen:** Autobahn = 200 m Bundesstraße = 150 m Landschafts-schutzgebiete = 500 m		Untersuchungs-gebiet 1
Szenario 2	Raster Ertrag: 75 Raster Straßendistanz: 25	Ergebnis von Szenario 1 als Maske verwenden		Untersuchungs-gebiet 1
Szenario 3	Raster Ertrag: 90 Raster Hangneigung: 10	**Maske:** Untersuchungsgebiet 2	Ertrag: > 7000 MWh/a Distanz Um-spannwerke: < 7500 m	Untersuchungs-gebiet 2
Szenario 4	Raster Ertrag: 10 Raster Umspannwerke: 5 Raster Straßendistanz: 5 Raster Hangneigung: 80	**Ausschlussflächen:** Autobahn = 200 m Bundesstraße = 150 m Landschafts-schutzgebiete = 500 m	Ertrag: > 7000 MWh/a Distanz Um-spannwerke: < 7500 m	Untersuchungs-gebiet 2
Szenario 5	Raster Ertrag: 65 Raster Umspannwerke: 20 Raster Straßendistanz: 13 Raster Hangneigung: 2	**Ausschlussflächen:** Wohngebiete = 800 m Gewerbegebiete = 300 m Industriegebiete = 300 m FFH-Gebiete = 150 m Naturschutzgebiete = 200 m Naturpark = 200 m Vogelschutzgebiet = 500 m Landschaftsschutzgebiet = 0 m Autobahn = 100 m Bundesstraße = 100 m Landstraße = 40 m Kreisstraße = 40 m Bahnlinien = 50 m	Ertrag: > 7500MWh/a Distanz Um-spannwerke: < 8000 m Distanz Straße: < 1500 m Hangneigung: < 20%	Untersuchungs-gebiet 1

Szenario 1

Wie in Abbildung 30 zu sehen ist, werden die angegebenen Abstandsflächen im Ergebnisraster nicht dargestellt. Auch die Werte für den Grad der Eignung können mit den vergebenen Gewichtungswerten erklärt werden: So erhalten beispielsweise ertragsarmen Gebiete im Norden des Untersuchungsgebiets, welche sich zudem in größerer Distanz zum nächsten Umspannwerk befinden, niedrige Eignungswerte. Ertragsreiche Gebiete in der Nähe von Umspannwerken, zum Beispiel im Süden des Untersuchungsgebiets, erhalten hingegen hohe Eignungswerte. In Abbildung 30 ist der Vollständigkeit halber noch das Einzugsgebiet dargestellt. Dieses Einzugsgebiet wurde auch bei den restlichen Szenarien als *Geoprocessing Extent* gewählt.

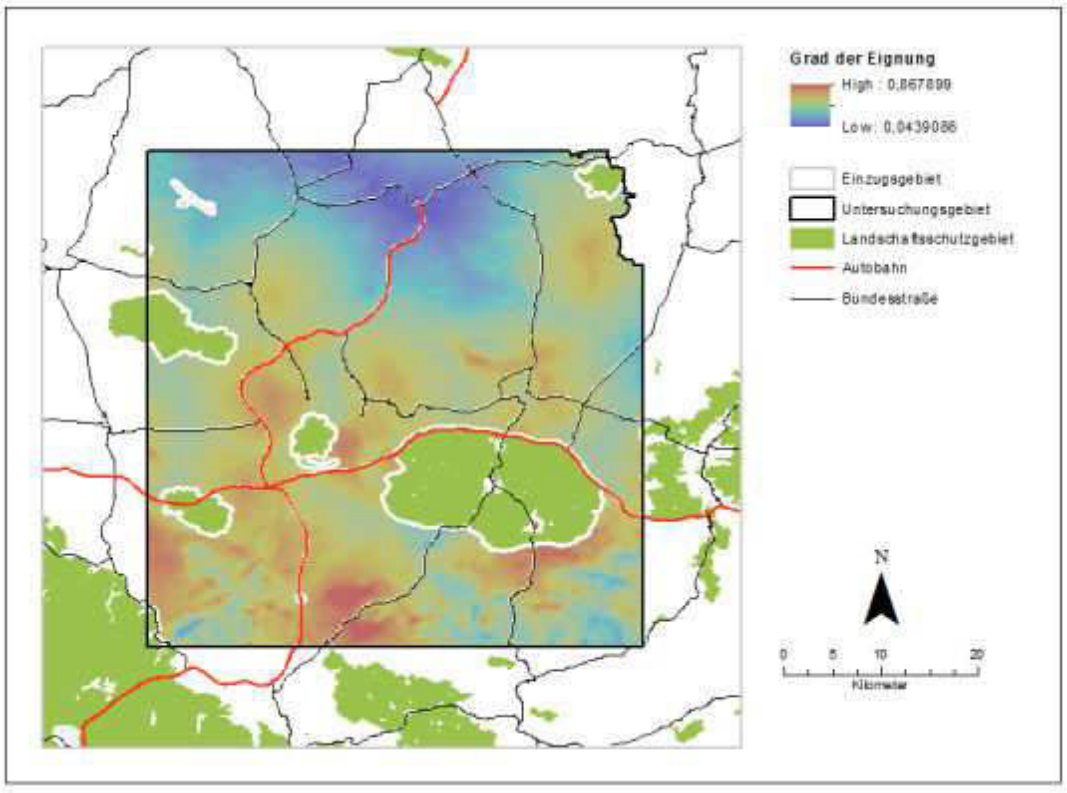


Abbildung 30: Ergebnis Plausibilitätsprüfung (Szenario 1)

Szenario 2

Im zweiten Szenario diente das Ergebnis des ersten Szenarios als Maske. Erwartungsgemäß ist das dargestellte Gebiet in beiden Fällen ident. Die hohe Gewichtung des Ertrags führt dazu, dass das Ergebnis stark den Energieertrags-Rastern aus Abbildung 14 ähnelt (Abb. 31).

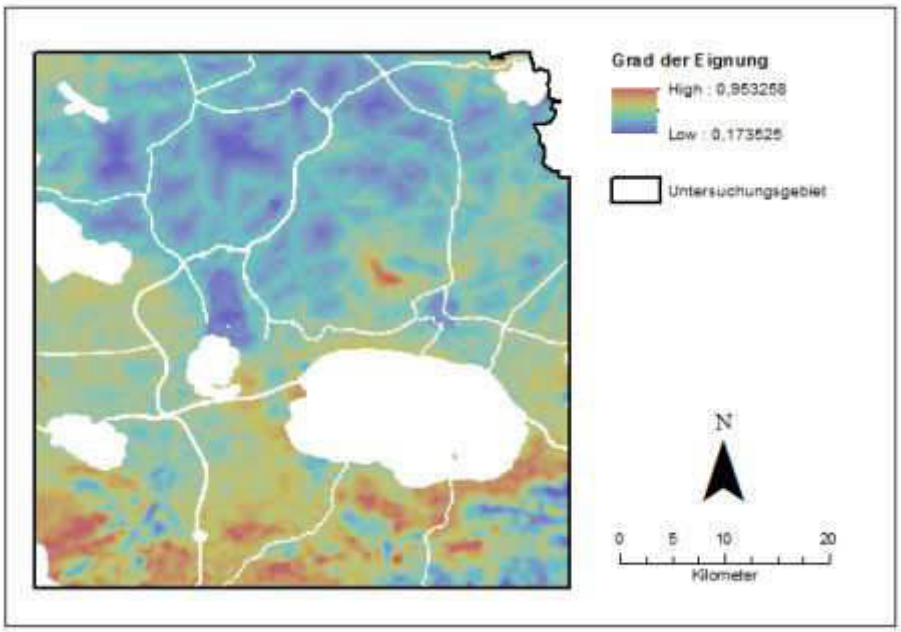


Abbildung 31: Ergebnis Plausibilitätsprüfung (Szenario 2)

Szenario 3

Beim dritten Szenario wurden lediglich Rasterwerte extrahiert. Auch hier ist das Resultat plausibel (Abb. 32): Das Ergebnis wurde auf das definierte Untersuchungsgebiet zugeschnitten. Die kreisförmigen Umrisse bestätigen, dass nur Gebiete innerhalb des angegebenen Radius zu den Umspannwerken berücksichtigt wurden. Ein Vergleich mit dem Ertragsraster zeigt zudem, dass Gebiete mit Erträgen geringer als 7000 MWh/a nicht dargestellt sind. Wie in Szenario 2 kommt auch hier die hohe Gewichtung des Ertrags zur Geltung. Die Hangneigung fällt hingegen kaum ins Gewicht.

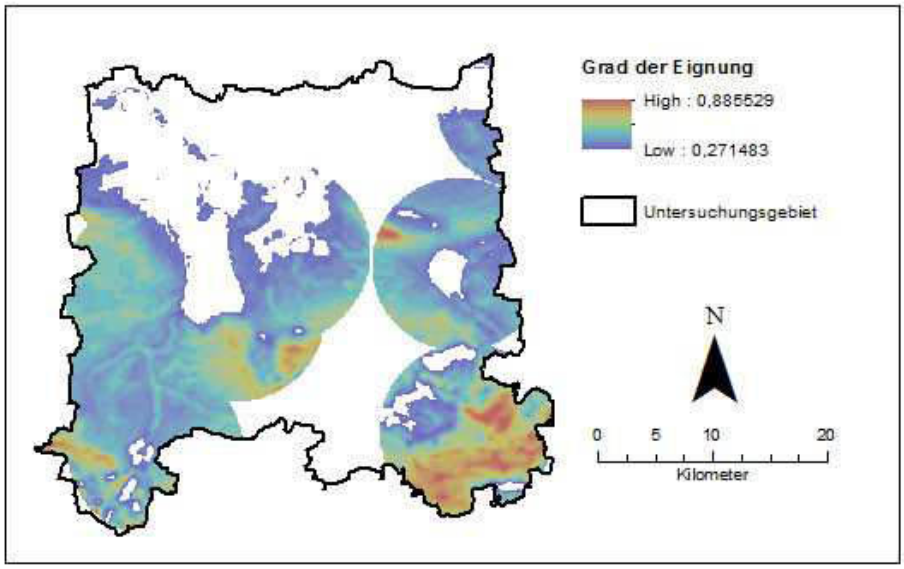


Abbildung 32: Ergebnis Plausibilitätsprüfung (Szenario 3)

Szenario 4

Auch das vierte Szenario liefert als Ergebnis Flächen, die den Eingabeparametern entsprechen (Abb. 33). Wie zu vermuten war, spiegelt sich in der Abbildung auch die hohe Gewichtung der Hangneigung wieder. Dies kommt besonders in der südwestlichen Ecke des Untersuchungsgebiets zur Geltung: Aufgrund der dort vorherrschenden hohen Hangneigungen erhalten diese Gebiete dementsprechend niedrige Eignungswerte.

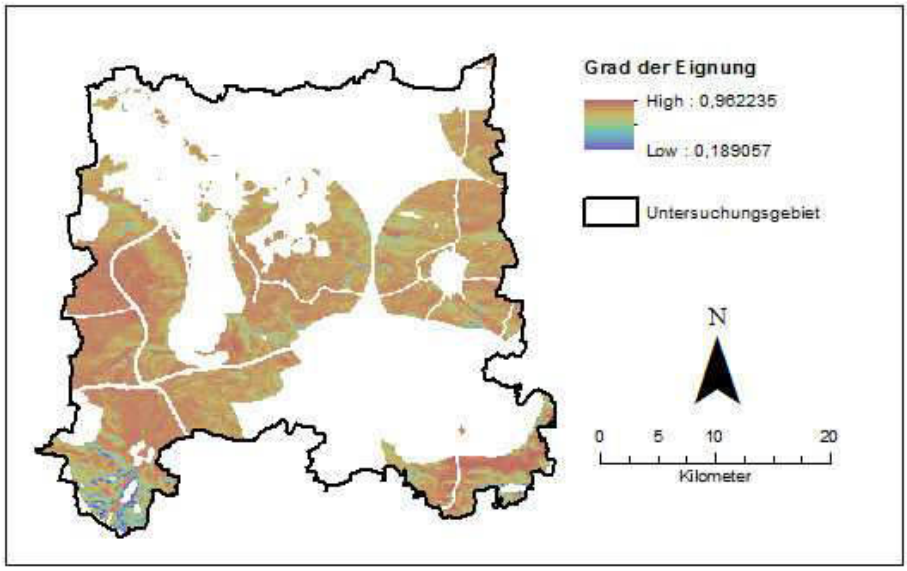


Abbildung 33: Ergebnis Plausibilitätsprüfung (Szenario 4)

Szenario 5

Das fünfte Szenario orientiert sich grob, was die politischen *Constraints* angeht, an den Empfehlungen des Thüringer Ministeriums für Bau, Landesentwicklung und Verkehr (2005) bzw. an den von der Bund-Länder-Initiative ‚Windenergie' herausgegebenen Abstandsempfehlungen (2013). Es sei nochmals darauf hingewiesen, dass aufgrund der Datenlage nicht alle dort erwähnten Abstandsflächen Berücksichtigung fanden. Dennoch kann dieses Szenario als eher realistische Flächenanalyse der Region angesehen werden. Ein Abgleich der Eingangsdaten mit dem Ergebnis bestätigt zudem dessen Plausibilität. Bei einem solchen Szenario wären nur sehr wenige hochwertige Flächen für die Windenergienutzung im Untersuchungsgebiet vorhanden. Das Szenario beweist zudem, dass das entwickelte SDSS in der Lage ist, eine hohe Anzahl von Datensätzen zu verarbeiten.

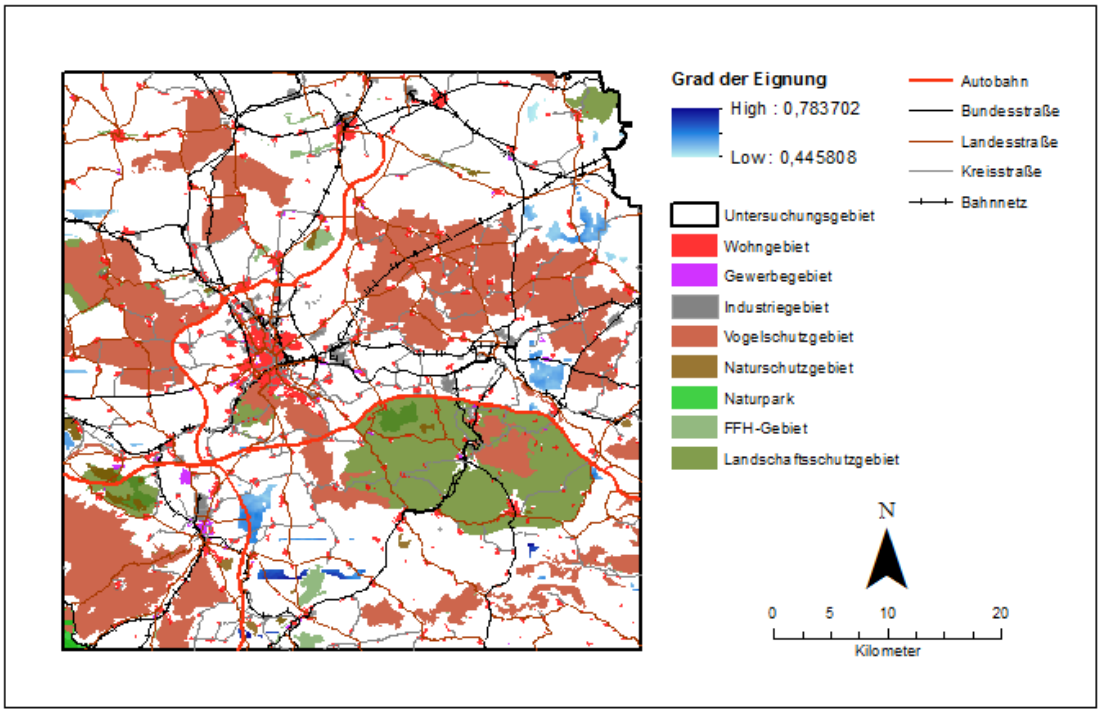


Abbildung 34: Ergebnis Plausibilitätsprüfung (Szenario 5)

7.2 Hypothesenprüfung

Im Folgenden werden die Hypothesen aus Kapitel 1.4 auf ihre Gültigkeit überprüft.

Hypothese A:

Das SDSS erfüllt alle Kriterien, die sich aus der Definition in Kapitel 2 ergeben.

Diese Hypothese kann größtenteils verifiziert werden. Das SDSS ist auf ein konkretes, semistrukturiertes und räumliches Entscheidungsproblem, das Auffinden und Bewerten geeigneter Flächen für WEAs, ausgerichtet. Zudem bietet die flexible Dateneingabe die Möglichkeit Szenario-Analysen durchzuführen. Damit leistet es einen Beitrag zur iterativen Problemlösung, da die Flächenanalyse für verschiedene Anwendungsfälle problemlos und schnell wiederholt werden kann. Hier sei auch auf den Punkt der Plausibilitätsprüfung in Kapitel 7.1 verwiesen, bei welchem mehrere Szenarien durchgerechnet wurden. Ebenfalls erzeugt das SDSS einen Bericht in Form einer Karte, welche einige benutzerdefinierte Parametereingaben beinhaltet. Der in der Definition aufgeführte Aspekt der leichten Bedienbarkeit des SDSS kann weder verifiziert noch falsifiziert werden. Hierzu wäre eigens ein Test mit mehreren Versuchspersonen notwendig, welcher die Bedienfreundlichkeit des SDSS untersucht. Es ist jedoch davon auszugehen, dass das SDSS in dieser Form zumindest nicht intuitiv zu bedienen ist.

Hypothese B:

Der Anwender kann das zu untersuchende Gebiet flexibel festlegen, Ausschluss- und Abstandsflächen selbst definieren sowie Eignungskriterien individuell auswählen und gewichten.

Hier sei ebenfalls auf den Punkt der Plausibilitätsprüfung verwiesen, bei welchem für die verschiedenen Szenarien unterschiedlichste Untersuchungsgebiete, Ausschluss- und Abstandsflächen, Eignungskriterien sowie Gewichtungswerte herangezogen wurden. Hypothese B kann somit verifiziert werden.

Hypothese C:

Das gesamte SDSS lässt sich als *Script-Tool* innerhalb von *ArcGIS 10.1* mittels der Skriptsprache *Python* über das *Arcpy-Site-Paket* umsetzen.

Hypothese D lässt sich ebenso bestätigen. Wie gezeigt wurde, konnte das gesamte SDSS mittels der Skriptsprache *Python* innerhalb von *ArcGIS 10.1* umgesetzt werden. Kritisch zu erwähnen bleibt die bereits erwähnte Bedienfreundlichkeit des SDSS.

8 Zusammenfassung und Ausblick

Mittels der Skriptsprache *Python* konnte innerhalb von *ArcGIS 10.1* ein SDSS entwickelt werden, das es ermöglicht, für beliebige Untersuchungsregionen, unter Berücksichtigung verschiedener Bewertungskriterien und beliebig vieler *Constraints*, Flächenanalysen für Windkraftanlagen durchzuführen. Für das Durchführen einer verlässlichen und möglichst sicheren Flächenanalyse müsste allerdings eine Erweiterung und Anpassung der Datenbasis erfolgen, aus welcher sich möglichst alle politischen Ausschlussflächen generieren lassen. Bei entsprechender Modifizierung der Datenbasis könnte das SDSS zudem für nicht-windenergiespezifische Standortfragen Verwendung finden.

Kritisch zu erwähnen bleibt allerdings, dass das GUI voraussichtlich selbst für GIS-affine Personen nicht auf Anhieb verständlich ist. Die Entwicklung einer Anwendung mit benutzerdefinierten *Interface* ließe sich mittels der Programmiersprachen *.NET, C++* oder *Java* unter Zugriff auf die *ArcObjects*-Bibliotheken erreichen (ESRI 2013b, Laudien et al. 2007).

Zudem sei nochmals auf die nicht näher erläuterte lineare Transformation der Eignungskriterien und die damit verbundene Kritik an der WLC verwiesen (Kapitel 2). Vor der Umsetzung eines SDSS sollte daher die Frage geklärt werden, welches Standardisierungskonzept für welches Kriterium am besten geeignet ist. Gegebenenfalls bestehen hier auch regionsspezifische Unterschiede. Eine Begründung für das angewendete Standardisierungsverfahren bleibt die vorliegende Arbeit jedenfalls schuldig. Auch in der einschlägigen Literatur erläutern die Autoren ihr Gründe hierzu nicht (Al-Yahyai et al. 2012, Baban & Parry 2011, Bennui et al. 2007, Gorsevski et al. 2013, Hansen 2005, Janke 2010, Rodman & Meentemeyer 2006, Tegou et al. 2010). Eine Analyse der für multikriterielle Bewertungsverfahren verwendeten Eignungskriterien kann somit als zukünftige Forschungsaufgabe angesehen werden. Ebenso gilt es die zur Anwendung kommende Methode der MKRE näher zu beleuchten. Diesbezüglich fehlt es an Studien, die untersuchen, ob die WLC oder eine andere Methode aus der Familie der MKREs sicherere Ergebnisse hinsichtlich der Eignung von Flächen für die Windkraftnutzung liefert.

Defizite heutiger SDSS sehen Gorsevski et al. (2013) zudem darin, dass diese größtenteils auf Desktop-GIS-Anwendungen basieren und der Öffentlichkeit nicht zugänglich

sind. Oftmals sind an Planungsprozessen jedoch verschiedene Gruppen mit kollidierenden Interessen zu beteiligen. Webbasierte Anwendungen könnten hier Abhilfe schaffen. Obschon solche Systeme keine Neuigkeit sind, stellt die Kombination multikriterieller Bewertungsverfahren mit diesen ein noch junges Feld dar, welches in Zukunft vermutlich noch an Bedeutung gewinnen wird (Greene et al. 2011).

Literatur

ALMOATAZ, A.Y., SAID, M.F. & MOHAMED, A.B. (2012): Geographic Information Systems (GIS) Application in Wind Farm Planning. – The Online Journal on Power and Energy Engineering 3, 2, 279-283.

AL-YAHYAI, S., CHARABI, Y., GASTLI, A. & AL-BADI, A. (2012): Wind farm land suitability indexing using multi-criteria analysis. – Renewable Energy 44, 80-87.

AYDIN, N.Y., KENTEL, E. & DUZGUN, S. (2010): GIS-based environmental assessment of wind energy systems for spatial planning: A case study from Western Turkey. – Renewable and Sustainable Energy Reviews 14, 364-373.

BABAN, S.M. & PARRY, T. (2001): Developing and applying a GIS-assisted approach to locating wind farms in the UK. – Renewable Energy 24, 59-71.

BENNUI, A., RATTANAMANEE, P., PUETPAIBOON, U., PHUKPATTARANONT, P. & CHETPATTANANONDH, K. (2007): Site selection for large wind turbine using GIS. In: Proceedings of the PSU-UNS International Conference on Engineering and Environment - ICEE. Hat Yai (Thailand): Prince of Songkla University.

BERGMANN, M. & HÖFLE, B. (2013): GIS-gestützte Standortplanung von Windenergieanlagen mit freien und amtlichen Geodaten. In: STROBL, J., BLASCHKE, T., GRIESEBNER, G. & ZAGEL, B. (Hrsg.). Angewandte Geoinformatik 2013: Beiträge zum 25. AGIT-Symposium Salzburg. Berlin: Wichmann, 480-489.

BOTTERO, M., COMINO, E., DURIAVIG, M., FERRETTI, V. & POMARICO, S. (2013): The application of a Multicriteria Spatial Decision Support System (MCSDSS) for the assessment of biodiversity conservation in the Province of Varese (Italy). – Land Use Policy 30, 730-738.

BOUDIA, S., BENMANSOUR, A., GHELLAI, N., BENMDJAHED, M. & HELLAL, M.T. (2012): Temporal assessment of wind energy resource in algerian highlands regions. – Revue des Energies Renouvelables 15, 1, 43-55.

BUND-LÄNDER-INITIATIVE WINDENERGIE (2013): Überblick zu den landesplanerischen Abstandsempfehlungen für die Regionalplanung zur Ausweisung von Windenergiegebieten.
Online abgerufen unter:
http://www.erneuerbare-energien.de/fileadmin/Daten_EE/Dokumente__PDFs_/abstandsempfehlungen_0512.pdf
(Stand: 25.11.2013)

CABELLO, M. & ORZA, J. (2010): Wind speed analysis in the province of Alicante, Spain. Potential for small-scale wind turbines. – Renewable and Sustainable Energy Reviews 14, 3185-3191.

DENSHAM, P.J. (1991): Spatial Decision Support Systems. In: MAGUIRE, D.J., GOOODCHILD, M.F. & DAVID, R.W. (Hrsg.). Geographical information systems: Principles and applications 1. Harlow (Essex): Longman Scientific & Technical, 403-412.

DORAN, J. & VERHOLEK, M. (1977): A Note on Vertical Extrapolation Formulas for Weibull Velocity Distribution Parameters. – Journal of Applied Meteorology 17, 410-412.

DROBNE, S. & LISEC, A. (2009): Multi-attribute Decision Analysis in GIS: Weighted Linear Combination and Ordered Weighted Averaging. – Informatica 33, 459-474.

EASTMAN, R.J. (1999): Multi-criteria evaluation and GIS. In: LONGLEY, P.A., GOODCHILD, M.F., RHIND, D.W. & MAGUIRE, D.J. (Hrsg.). Geographical information systems. New York: Wiley, 493-502.

EMEIS, S. (2001): Vertical variation of frequency distributions of wind speed in and above the surface layer observed by sodar. – Meteorologische Zeitschrift 10, 2, 141-149.

ENERCON (2013): Leistungskennlinie ENERCON E-115 3,0 MW (Vers. 1.0 / 21.06.2013 / D0266588-0).

ESRI (2013a): Verwenden des In-Memory-Workspaces.
Online abgerufen unter:
http://resources.arcgis.com/de/help/main/10.1/index.html#//002w0000005s000000
(Stand: 25.05.2014, letzte Änderung: 11.09.2013)

ESRI (2013b): ArcGIS 10.1 Engine Developer Kit und ArcGIS 10.1 Engine – Erste Schritte.
Online abgerufen unter:
http://resources.arcgis.com/de/help/quick-start-guides/10.1/index.html#//01q100000004000000
(Stand: 25.05.2014, letzte Änderung: 02.04.2013)

FEIZIZADEH, B. & BLASCHKE, T. (2014): An uncertainty and sensitivity analysis approach for GIS-based multicriteria landslide susceptibility mapping. – International Journal of Geographical Information Science 28, 3, 610-638.

GASS, V., SCHMIDT, J., STRAUSS, F. & SCHMID, E. (2013): Assessing the economic wind power potential in Austria. – Energy Policy 53, 323-330.

GERTH, W.P. & CHRISTOFFER, J. (1994): Windkarten von Deutschland. – Meteorologische Zeitschrift 3, 67-77.

GORSEVSKI, P.V., CATHCART, S.C., MIRZAEI, G., JAMALI, M.M., YE, X. & ENRIQUE, G. (2013): A group-based spatial decision support system for wind farm site selection in Northwest Ohio. – Energy Policy 55, 374-385.

GRASSI, S., CHOKANI, N. & ABHARI, R.S. (2012): Large scale technical and economical assessment of wind energy potential with a GIS tool: Case study Iowa. – Energy Policy 45, 73-85.

GREENE, R., DEVILLERS, R., LUTHER, J.E. & EDDY, B.G. (2011): GIS-Based Multiple-Criteria Decision Analysis. – Geography Compass 5, 6, 412-432.

HANSEN, H.S. (2005): GIS-based Multi-Criteria Analysis of Wind Farm Development. In: Proceedings of the 10th Scandinavian Research Conference on Geographical Information Science (ScanGIS). Stockholm: Department of Planning and Environment, 75-87.

HEIER, S. (2008[5]): Nutzung der Windenergie. BINE-Informationsdienst. Berlin: Solarpraxis.

HOCEVAR, A. & RIEDL, L. (2003): Vergleich verschiedener multikriterieller Bewertungsverfahren mit MapModels. In: SCHRENK, M. (Hrsg.). "CORP 2003" Computergestützte Raumplanung.: 8. Symposion zur Rolle der Informationstechnologie in der und für die Raumplanung. 2. Wien: Technische Universität Wien, 299-304.

JACOBSEN, K. (2010): Vergleich von ASTER GDEM- mit SRTM-Höhenmodellen. In: KOHLHOFER, G. & FRANZEN, M. (Hrsg.). Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V.: 30. Wissenschaftlich-Technische Jahrestagung der DGPF 19. Münster: Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation, 581-588.

JANKE, J.R. (2010): Multicriteria GIS modeling of wind and solar farms in Colorado. – Renewable Energy 35, 2228-2234.

JIANG, H. & EASTMAN, R.J. (2000): Application of fuzzy measures in multicriteria evaluation in GIS. – International Journal of Geographical Information Science 14, 2, 173-184.

JUSTUS, C.G. & MIKHAIL, A. (1976): Heigt variation of wind speed and wind distributions statistics. – Geophysical Research Letters 3, 5, 261-264.

KARNATAK, H.C., SARAN, S., BHATIA, K. & ROY, P. (2007): Multicriteria Spatial Decision Analysis in Web GIS Environment. – Geoinformatica 11, 407-429.

KEENAN, P.B. (2006): Spatial decision support systems: a coming of age. – Control and Cybernetics 35, 1, 9-27.

LAUDIEN, R., RÖHRIG, J., BARETH, G. & MENZ, G. (2007): Spatial Decision Support System zur Modellierung der agrarischen Marginalität in Benin (Westafrika). In: STROBL, J., BLASCHKE, T. & GRIESEBNER, G. (Hrsg.). Angewandte Geoinformatik 2007: Beiträge zum 19. AGIT-Symposium Salzburg. Heidelberg: Wichmann, 430-439.

LILBURNE, L., TARANTOLA, S. (2009): Sensitivity analysis of spatial models. – International Journal of Geographical Information Science 23, 2, 151-168.

MALCZEWSKI, J. (1999): GIS and multicriteria decision analysis. New York, Chichester, Weinheim, Brisbane, Singapore, Toronto: Wiley.

MALCZEWSKI, J. (2004): GIS-based land-use suitability analysis: a critical overview. – Progress in Planning 62, 1, 3-65.

MALCZEWSKI, J. (2006): GIS-based multicriteria decision analysis: a survey of the literature. – International Journal of Geographical Information Science 20, 7, 703-726.

MANN, D., LANT, C. & SCHOOF, J. (2012): Using map algebra to explain and project spatial patterns of wind energy development in Iowa. – Applied Geography 34, 219-229.

MARI, R., BOTTAI, L., BUSILLO, C., CALASTRINI, F., GOZZINI, B. & GUALTIERI, G. (2011): A GIS-based interactive web decision support system for planning wind farms in Tuscany (Italy). – Renewable Energy 36, 754-763.

MCWILLIAM, M., VAN KOOTEN, G. & CRAWFORD, C. (2012): A method for optimizing the location of windfarms. – Renewable Energy 48, 287-299.

OUAMMI, A., GHIGLIOTTI, V., ROBBA, M., MIMET, A. & ROBERTO, S. (2012): A decision support system for the optimal exploitation of wind energy on regional scale. – Renewable Energy 37, 299-309.

PETERSEN, E.L., MORTENSEN, N.G., LANDBERG, L., HØJSTRUP, J. & FRANK, H.P. (1997): Wind power meteorology. Roskilde (Denmark): Risø National Laboratory.
Online abgerufen unter:
http://www.windatlas.dk/home/download/wind%20power%20meteorology.pdf
(Stand: 25.01.2014)

RAMIREZ-ROSADO, I.J., GARCIA-GARRIDO, E., FÉRNANDEZ-JIMÉNEZ, A., ZORZANO-SANTAMARIA, P.J., MONTEIRO CLÁUDIO & MIRANDA VLADIMIRO (2008): Promotion of new wind farms based on a decision supportsystem. – Renewable Energy 33, 558-566.

RODMAN, L.C. & MEENTEMEYER, R.K. (2006): A geographic analysis of wind turbine placement in Northern California. – Energy Policy 34, 2137-2149.

ROZSAVOLGYI, K. (2009): Spatial Complex Model for Wind Farm Site Assessment. – The Open Atmospheric Science Journal 3, 204-211.

SCHIEMENZ, B. & SCHÖNERT, O. (2005[3]): Entscheidung und Produktion. Lehr- und Handbücher der Betriebswirtschaftslehre. München, Wien: Oldenbourg.

SIMÃO, A., DENSHAM, P.J. & HAKLAY, M. (2009): Web-based GIS for collaborative planning and public participation: An application to the strategic planning of wind farm sites. – Journal of Environmental Management 90, 2027-2040.

SIMON, H.A. (1960): The New Science of Management Decision. The Ford distinguished lectures 3. New York, Evanston: Harper & Row.

SLIZ-SZKLINIARZ, B. & VOGT, J. (2011): GIS-based approach for the evaluation of wind energy potential: A case study for the Kujawsko–Pomorskie Voivodeship. – Renewable and Sustainable Energy Reviews 15, 1696-1707.

SUGUMARAN, R. & DEGROOTE, J. (2011): Spatial Decision Support Systems: Principles and Practices. Boca Raton, London, New York: CRC Press.

TEGOU, L.-I., POLATIDIS, H. & HARALAMBOPOULOS, D.A. (2010): Environmental management framework for wind farm siting: Methodology and case study. – Journal of Environmental Management 91, 2134-2147.

THÜRINGER LANDESMINISTERIUM FÜR BAU, LANDESENTWICKLUNG UND VERKEHR (2005): Handlungsempfehlung für die Fortschreibung der Regionalpläne zur Ausweisung von Vorranggebieten „Windenergie“, die zugleich die Wirkung von Eignungsgebieten haben.
Online abgerufen unter:
https://www.thueringen.de/imperia/md/content/tmbv/landesplanung/grundlagen/regelungen/handlungsempfehlung-ausweisung-vorranggebiete-windenergie.pdf (Stand: 25.11.2013)

TONG, W. (Hrsg.) (2010): Wind Power Generation and Wind Turbine Design. Southampton, Boston: WIT Press.

VAN HAAREN, R. & FTHENAKIS, V. (2011): GIS-based wind farm site selection using spatial multi-criteria analysis (SMCA): Evaluating the case of New York State. – Renewable and Sustainable Energy Reviews 15, 3332-3340.

VOIVONTAS, D., ASSIMACOPOULOS, D. & MOURELATOS, A. (1998): Evaluation of renewable energy potential using a GIS decision support system. – Renewable Energy 13, 333-344.

VOOGD, J.H. (1983): Multicriteria evaluation for urban and regional planning. London: Pion.

YEH, T.-H. & WANG, L. (2008): A study on generator capacity for wind turbines under various tower heights and rated wind speeds using weibull distribution. – IEEE Transaction on Energy Conversion 23, 592-602.

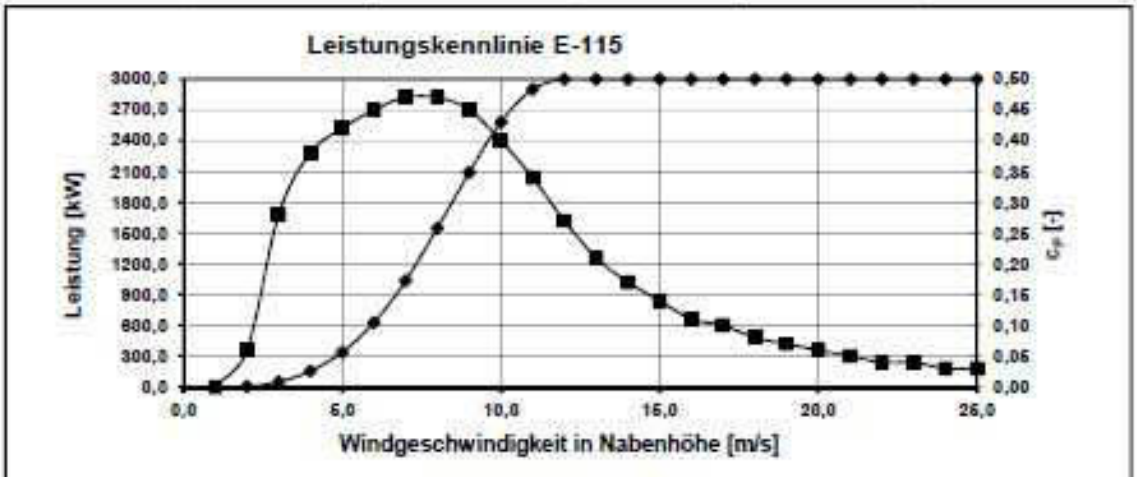
Anhang A: Leistungskennlinie Enercon E115 (3.0MW)

ENERCON
ENERGY FOR THE WORLD

Leistungskennlinie ENERCON E-115 3,0 MW

Nennleistung:	3.000 kW
Leistungskennlinie:	berechnet (Vers. 1.0 / 21.06.2013 / D0266588-0)
Standardluftdichte	1,225 kg/m³

Windgeschwindigkeit v [m/s]	Leistung P [kW]	cp [-]
1,0	0,0	0,00
2,0	3,0	0,06
3,0	48,5	0,28
4,0	155,0	0,38
5,0	339,0	0,42
6,0	627,5	0,45
7,0	1.035,5	0,47
8,0	1.549,0	0,47
9,0	2.090,0	0,45
10,0	2.580,0	0,40
11,0	2.900,0	0,34
12,0	3.000,0	0,27
13,0	3.000,0	0,21
14,0	3.000,0	0,17
15,0	3.000,0	0,14
16,0	3.000,0	0,11
17,0	3.000,0	0,10
18,0	3.000,0	0,08
19,0	3.000,0	0,07
20,0	3.000,0	0,06
21,0	3.000,0	0,05
22,0	3.000,0	0,04
23,0	3.000,0	0,04
24,0	3.000,0	0,03
25,0	3.000,0	0,03



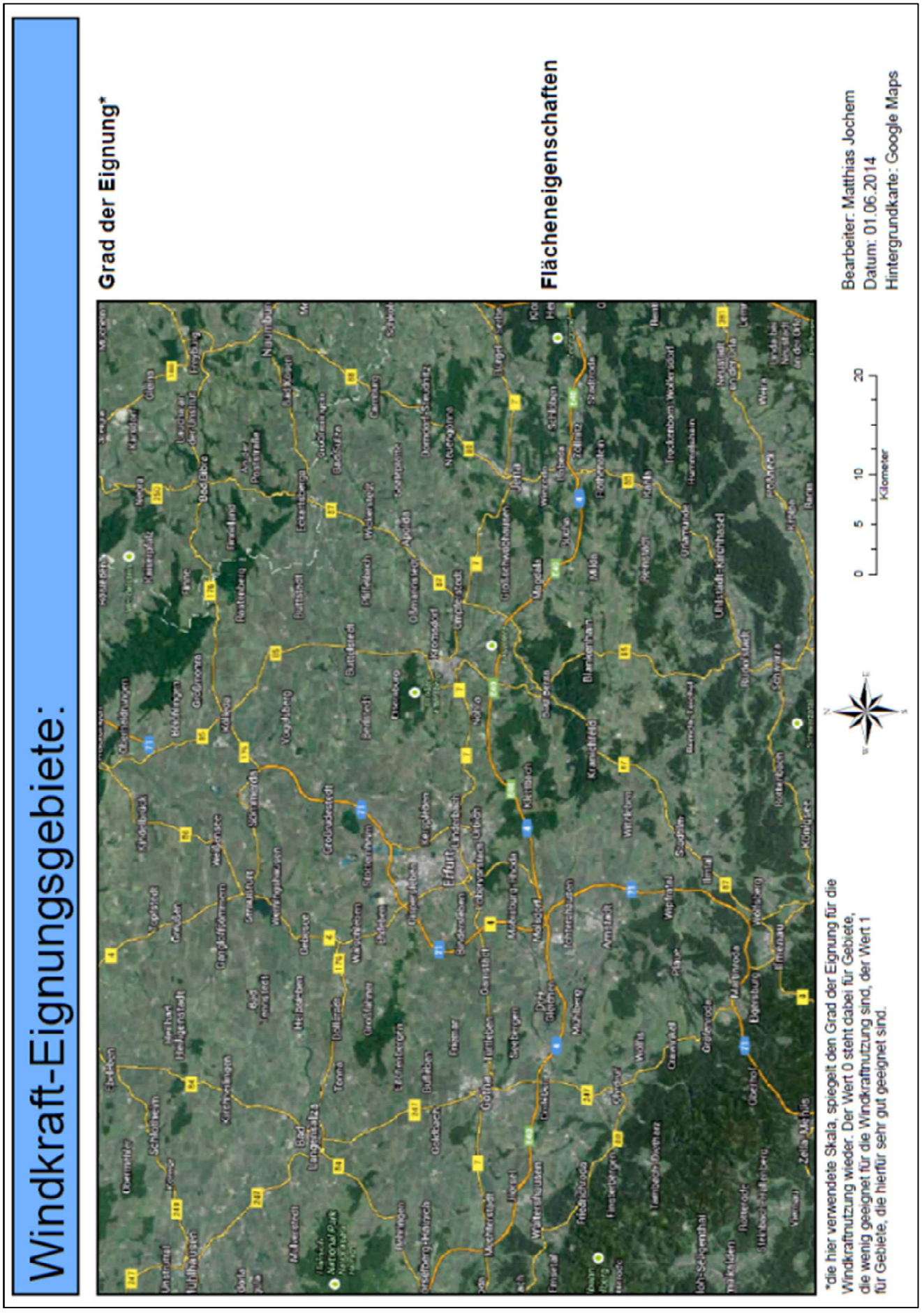
Dokument Informationen:

Autor/Datum:	CN/ 26.06.13
Abteilung:	SIAS
Geprüft/Datum:	KP/ 26.06.13

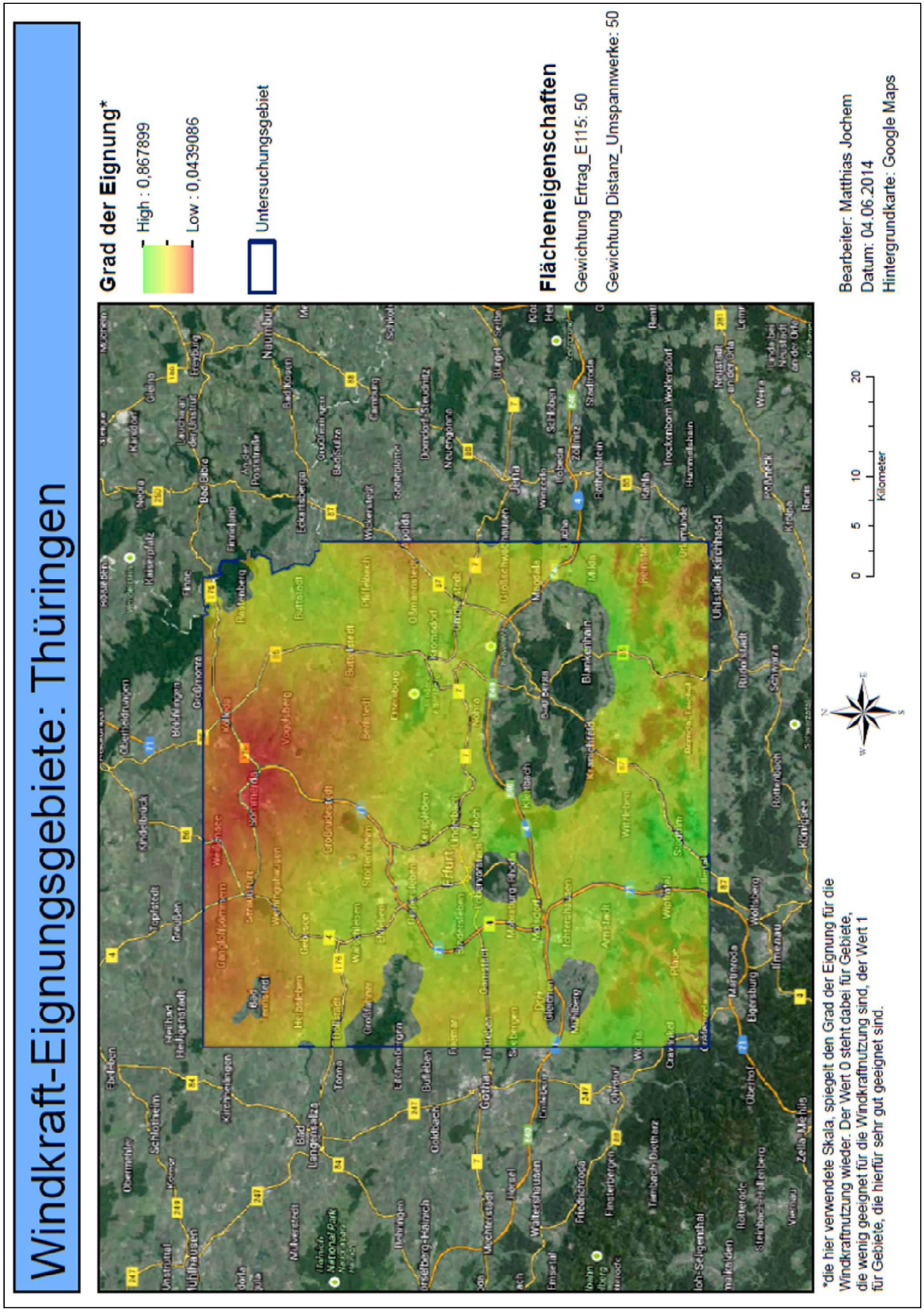
Translator/date:
Revisor/date:

Referenz: Power Curve E115 3.0MW calculated Vers 1_0

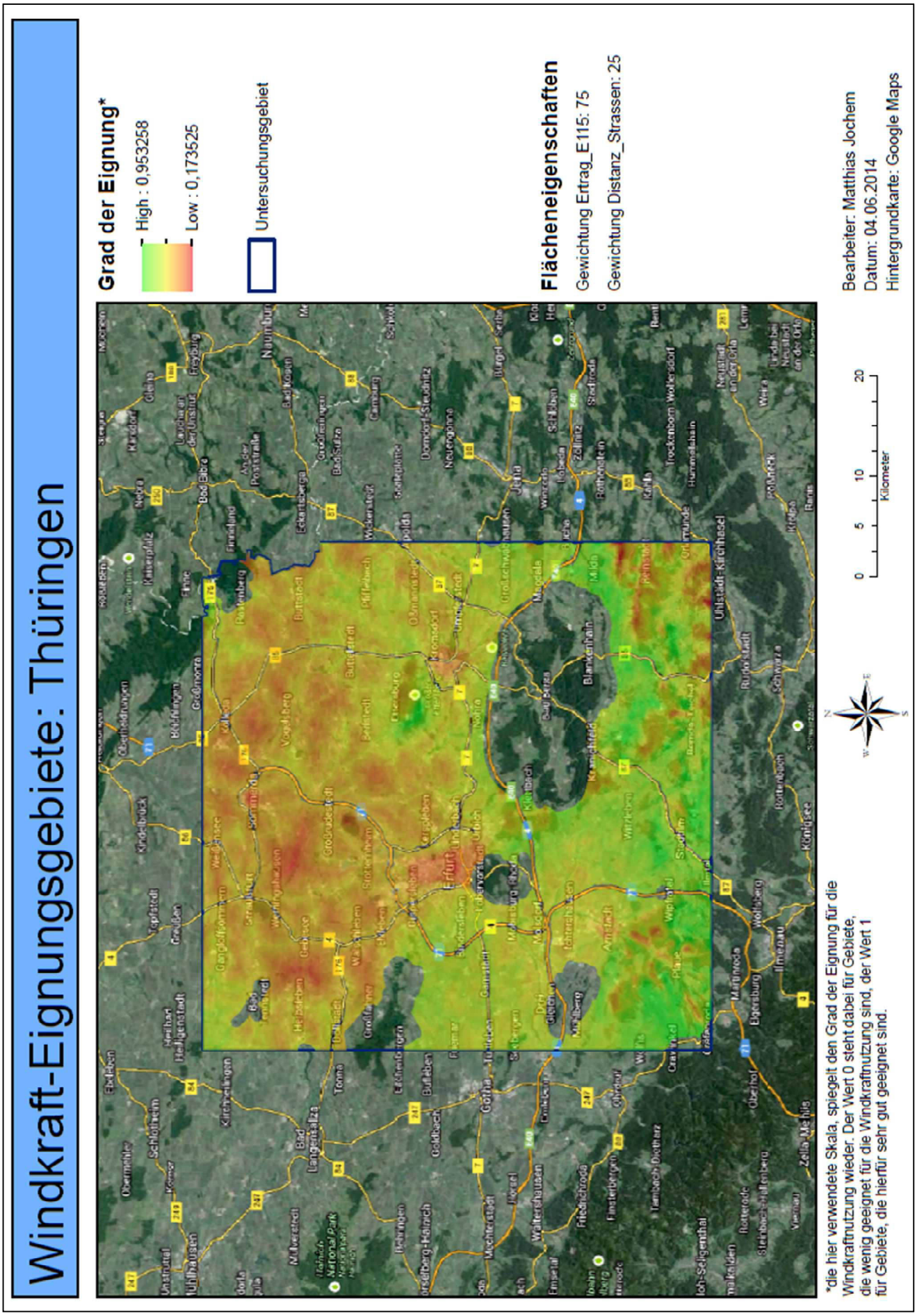
Anhang B: Kartenvorlage SDSS



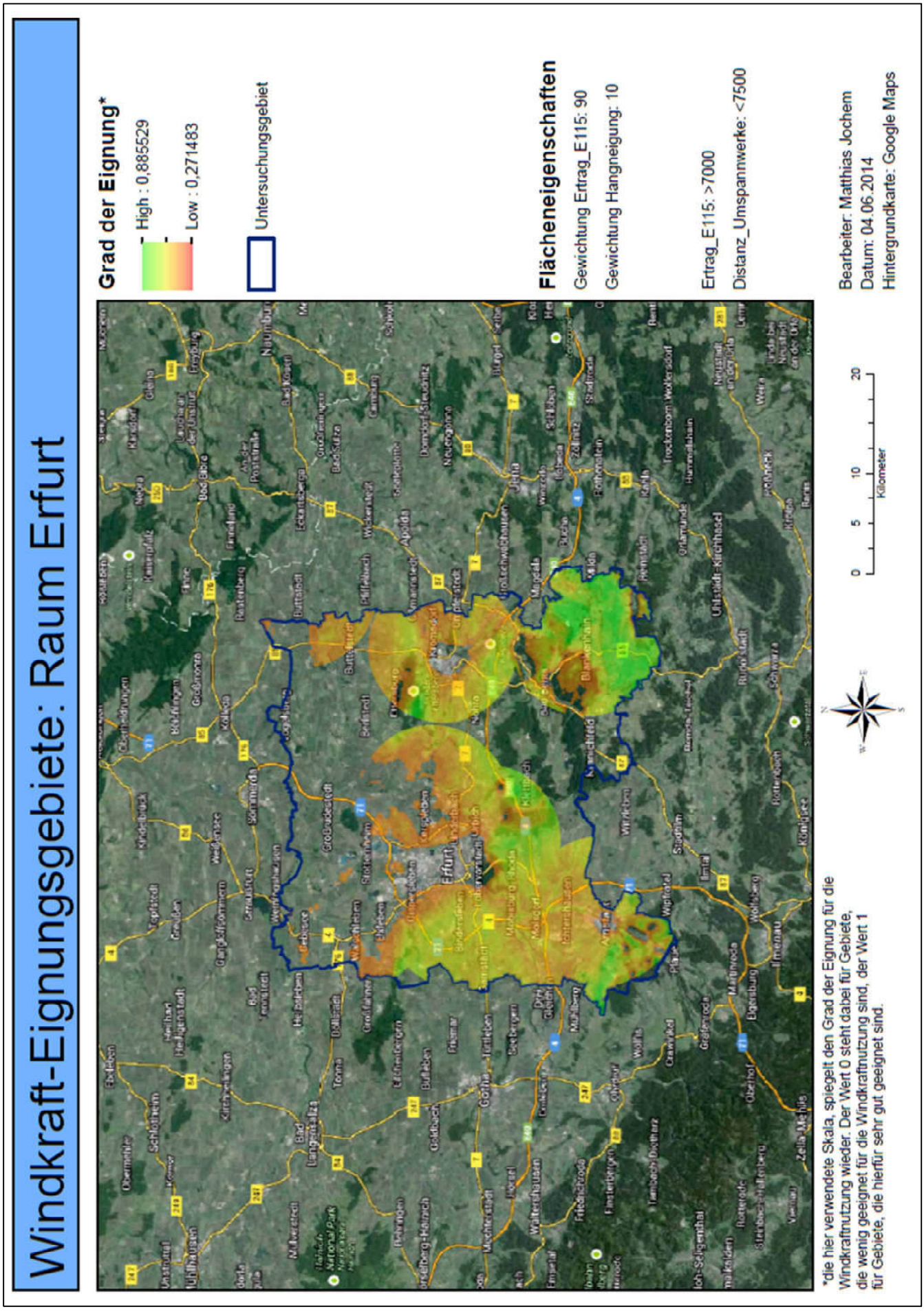
Anhang C: SDSS-Output der Plausibilitätsprüfung (Szenario 1)



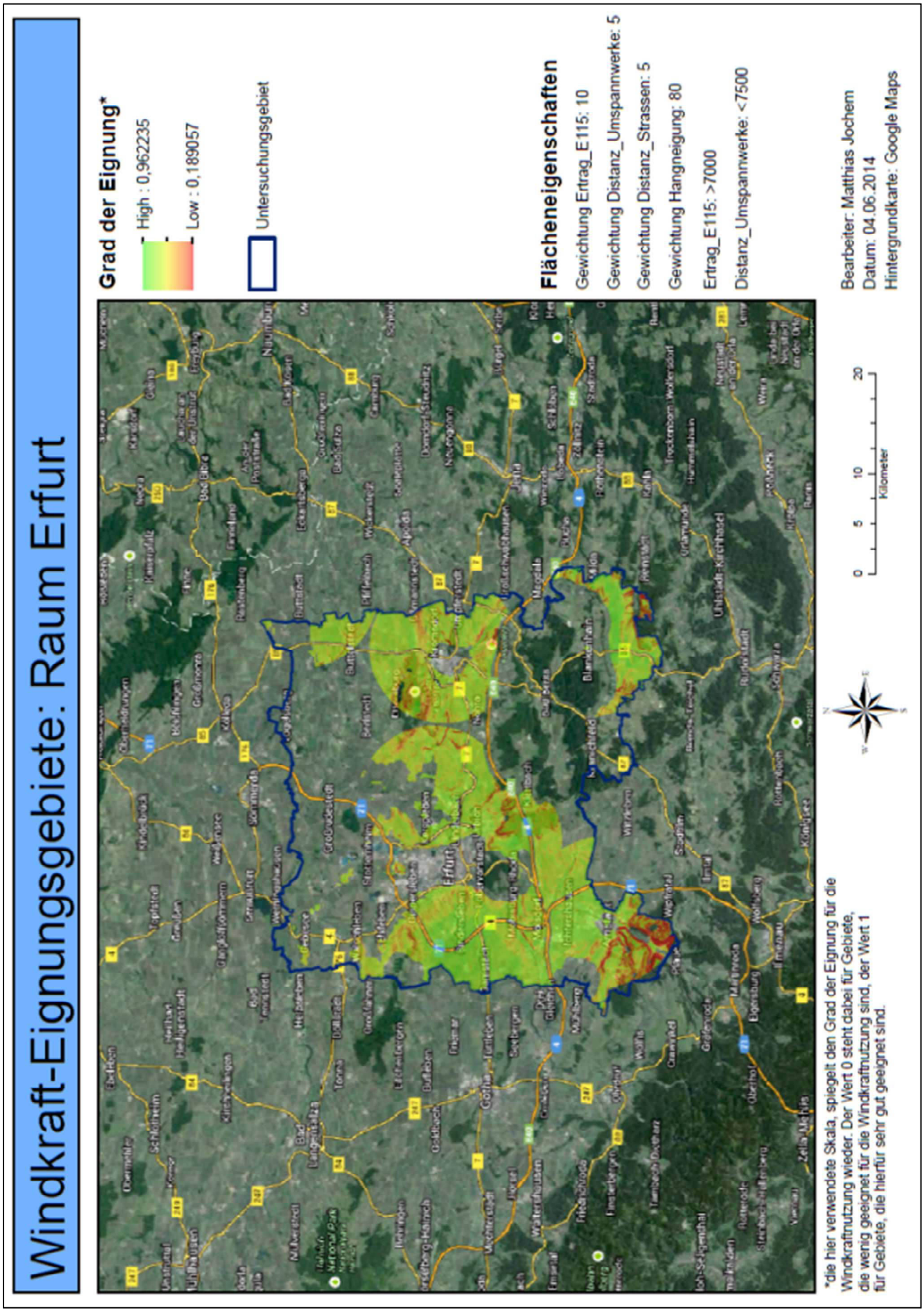
Anhang D: SDSS-Output der Plausibilitätsprüfung (Szenario 2)



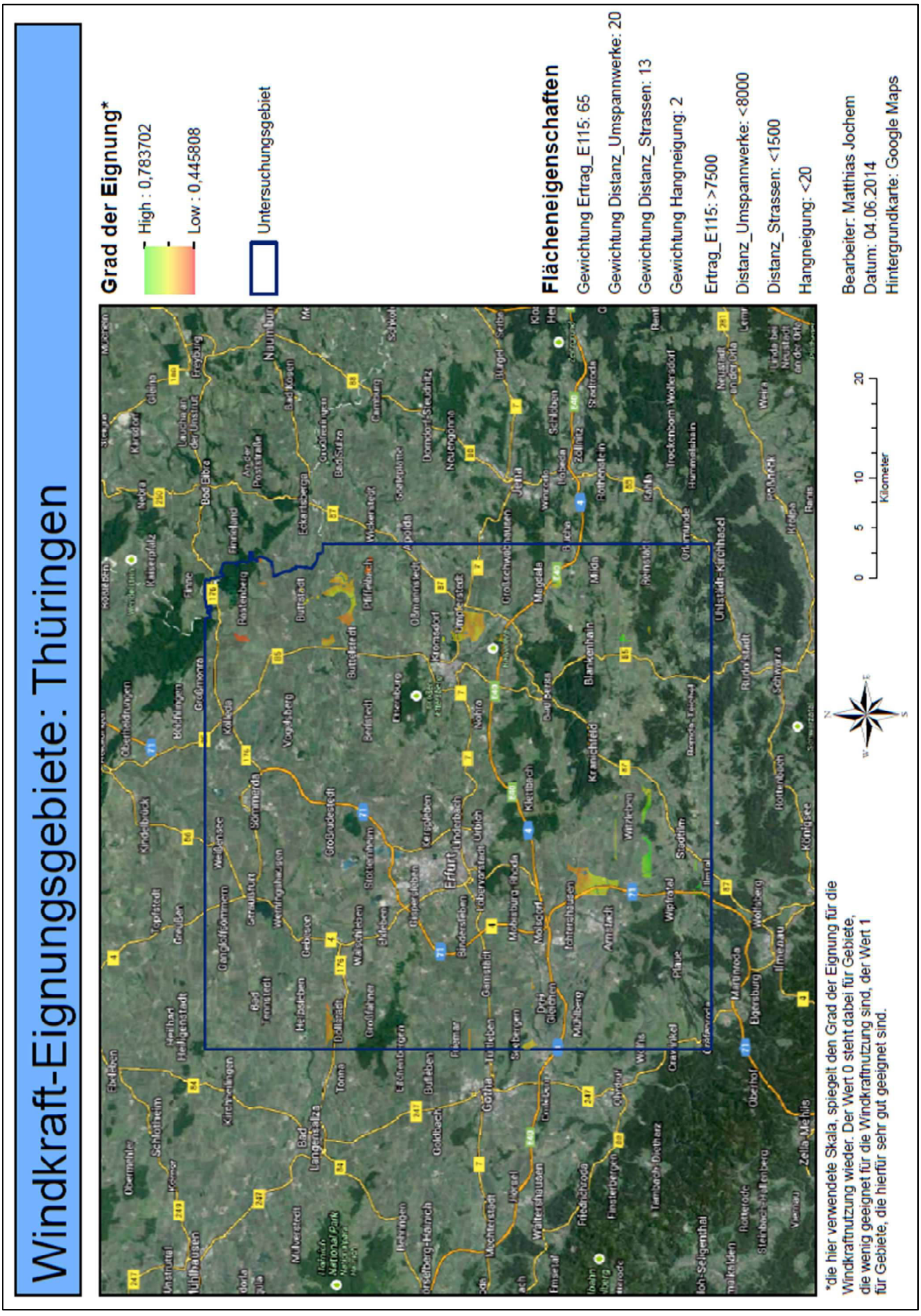
Anhang E: SDSS-Output der Plausibilitätsprüfung (Szenario 3)



Anhang F: SDSS-Output der Plausibilitätsprüfung (Szenario 4)



Anhang G: SDSS-Output der Plausibilitätsprüfung (Szenario 5)



Anhang H: Python Code

(Python Code: nächste Seite)

```
1  # Author: Matthias Jochem
2  # Date: 28.07.2014
3  # Zweck: Dieses Tool stellt ein SDSS dar, das die Flaechensuche und
   Flaechenbewertung für Standorte von Windkraftanlagen unterstuetzt.
4  # Neben einer multikriteriellen Bewertung(Weighted Overlay) besteht die
   Moeglichkeit Rasterwerte aus der Berechnung auszuschliessen bzw. einzuschliessen
5  # (Hangneigung, Energieertrag, Distanz zu Straßen, Distanz zu Umspannwerken).
   Zudem ist es möglich eine Weissflaechenanalyse anzuschliessen,
6  # sodass das Ergebnis der Flaechenbewertung auf "juristisch" legitime Flaechen
   beschraenkt wird.
7  # Das Tool ist sehr flexibel angelegt. Es bietet die Moeglichkeit bis zu 4
   Raster zu gewichten, sowie Werte aus der
8  # Analyse aus- bzw- einzuschliessen. Auch die integrierte Weissflaechenanalyse
   bietet die Moeglichkeit beliebig viele Feature Classes zu Puffern und
9  # diese Puffer aus der Bewertung auszuschliessen. So laesst sich das Tool auf
   verschiedenen Gebiete bzw. Bundeslaender mit unterschiedlichen
   Abstandsempfehlungen
10 # anwenden. Zudem wird das Ergebnis einer einmal durchgefuehrten
   Weissflaechenanalyse gespeichert, sodass dieser rechenintensive Prozess bei
   einem Gebiet nicht
11 # wiederholt werden muss. Bei einer erneuten Flaechenanlayse bzw. -bewertung
   koennen dann die einmal generierten Weissflaechen als Maske ueber die gewichteten
12 # und evtl. selektierten Raster gelegt werden. Das Tool generiert automatisch
   eine Karte als pdf-Dokument, welches neben der farblich dargestellten
13 # Flaechenbewertung auch die eingegebenen Gewichtungen und die in die
   Berechnung eingeflossenen Rasterwerte als Textelement beinhaltet.
14
15 # Sonstiges: Tool erstellt im Rahmen der Masterarbeit des UNIGIS (Msc)-Studiums
   (Jahrgang 2012).
16
17 # Beginn des Scripts:
18 # Import des arcpy- und sa-Moduls
19
20 import arcpy
21 from arcpy.sa import *
22
23 # Definition des Workspaces und der notwendigen Eingabe-Parameter
24 # Workspace dient nur als Zwischenspeicher (Ausgabe-Datein des Tools werden vom
   Benutzer festgelegt)
25 scratchPath = "in_memory"
26 arcpy.env.workspace = scratchPath
27
28 # Datein im Zwischenspeicher duerfen ueberschrieben werden
29 arcpy.env.overwriteOutput = True
30
31 # Geoprocessing Extent wird vom Benutzer festgelegt. Bei Verwendung der
   Weissflaechenanalyse sollte Geoprocessing Extent groeßer als
   Untersuchungsgebiet sein.
32 gpExtent = arcpy.GetParameterAsText(0)
33
34 # Untersuchungsgebiet wird ebenfalls vom Benutzer festgelegt. Bei Verwendung
   der Weissflaechenanalyse sollte Geoprocessing Extent groeßer als
   Untersuchungsgebiet sein.
35 ugArea = arcpy.GetParameterAsText(1)
36
37 # Name des Untersuchungsgebietes als
   Texteingabe
38 nameUG = arcpy.GetParameterAsText(2)
```

```
39
40  # Nur Objekte innerhalb des angegeben Geoprocessing Extent werden von dem Tool
    verarbeitet.
41  arcpy.env.extent = gpExtent
42
43  # Eingabe der reklassifizierten Raster, die gewichtet werden sollen.
44  rasterWeight = arcpy.GetParameterAsText(3)
45
46  # Eingabe der jeweiligen Rastergewichte.
47  weightsRaster = arcpy.GetParameterAsText(4)
48
49  # Eingabe der Raster mit absoluten Werten.
50  rasterInput = arcpy.GetParameterAsText(5)
51
52  # Selektionsausdruck, der festlegt, welche Werte der absoluten Raster in die
    Berechnung einfliessen sollen.
53  whereClause = arcpy.GetParameterAsText(6)
54
55  # Feature Classes, die in die Weissflaechenanalyse einfliessen sollen.
56  bufferFeature = arcpy.GetParameterAsText(7)
57
58  # Puffer-Distanzen um die Feature Classes der Weissflaechenanalyse. Diese
    Puffer werden aus dem Untersuchungsgebiet
59  # ausgeschnitten. Die übrigen Flaechen stellen die Weissflaechen dar. Sie
    werden als Maske ueber die gewichteten bzw.
60  # selektierten Raster gelegt. Das Ergebnis beschraenkt sich somit auf die
    "juristisch" legitimen Flaechen.
61  bufferDistance = arcpy.GetParameterAsText(8)
62
63  # Ausgabedatei der Weissflaechenanalyse. Diese Ergebniss kann bei erneuter
    Berechnung als Maske in Parameter(10) eingeben werden,
64  # sodass der rechenintensive Prozess der Weissflaechenanalyse bei einem Gebiet
    nicht wiederholt werden muss.
65  outWeissFlaeche = arcpy.GetParameterAsText(9)
66
67  # Eingabe der Maske, auf die das Ergebniss der Flaechenbewertung beschraenkt
    werden soll. Kann sich um einmal generierte Datei
68  # des Parameters(9) handeln.
69  clipFeature = arcpy.GetParameterAsText(10)
70
71  # Speicherort und Name für das Ausgaberaster
72  outputData = arcpy.GetParameterAsText(11)
73
74  # Speicherort und Name des Karten bzw. pdf-Dokuments
75  outputPDF = arcpy.GetParameterAsText(12)
76
77  # Feature Class, aus welcher die gepufferten Feature Classes fuer die
    Weissflaechenanalyse ausgeschnitten werden sollen. Entspricht dem Polygon des
78  # Geoprocessing Extent (Parameter(0)).
79  inputErase = gpExtent
80
81  # Aktivierung Spatial Analyst
82  arcpy.CheckOutExtension("Spatial")
83
84
85  # Beginn der Definition von Funktionen, die mehrmals im Skript auftauchen.
86
87  # Funktion "RasterGewicht" ist fuer die Weighted-Overlay Bewertung
```

```
    (multikriterielle Bewertung) verantwortlich:
88  # Als Eingabeparameter werden die zu gewichtenden reklassifizierten Raster und
    die Rastergewichte festgelegt.
89  def RasterGewicht(rasterWeight, weightsRaster):
90
91      # Da es sich um eine Multivalue-Eingabe handelt, wird der String gesplittet
        und somit eine Liste erzeugt.
92      listRasterWeight = rasterWeight.split(";")
93
94      # Auch bei den Rastergewichten handelt es sich um eine Multivalue-Eingabe,
        die gesplittet wird und eine Liste erzeugt wird.
95      listWeights = weightsRaster.split(";")
96
97      # Falls die Laenge der Liste der Raster und der Rastergewichte
        unterschiedlich ist, wird eine Fehlermeldung ausgegeben und das Programm
        beendet.
98      if len(listRasterWeight) != len(listWeights):
99          arcpy.AddError("Die Anzahl der zu gewichtenden Raster und der
            Rastergewichte muss gleich sein")
100         exit(1)
101
102     # Falls die Rasterliste aus nur einem Raster besteht wird ebenfalls eine
        Fehlermeldung ausgegeben und das Programm beendet.
103     elif len(listRasterWeight) == 1:
104         arcpy.AddError("Es muessen mindestens 2 Raster gewichtet werden")
105         exit(1)
106
107     # Ist die Listenlaenge gleich 2 werden die reklassifizierte Raster mit
        ihren gewichten multipliziert und addiert. Durch die Division wird das
108     # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
109     elif len(listRasterWeight) == 2:
110         outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
            Raster(listRasterWeight[1]) * int(listWeights[1]))/(int(listWeights[0])
            + int(listWeights[1]))
111
112     # Ist die Listenlaenge gleich 3 werden die reklassifizierte Raster mit
        ihren gewichten multipliziert und addiert. Durch die Division wird das
113     # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
114     elif len(listRasterWeight) == 3:
115         outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
            Raster(listRasterWeight[1]) * int(listWeights[1]) + Raster(
            listRasterWeight[2]) * int(listWeights[2]))/(int(listWeights[0]) + int(
            listWeights[1]) + int(listWeights[2]))
116
117     # Ist die Listenlaenge gleich 4 werden die reklassifizierte Raster mit
        ihren gewichten multipliziert und addiert. Durch die Division wird das
118     # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
119     elif len(listRasterWeight) == 4:
120         outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
            Raster(listRasterWeight[1]) * int(listWeights[1]) + Raster(
            listRasterWeight[2]) * int(listWeights[2]) + Raster(listRasterWeight[3])
             * int(listWeights[3]))/(int(listWeights[0]) + int(listWeights[1]) + int
            (listWeights[2]) + int(listWeights[3]))
121
122     # Funktion gibt als Ergebnis das auf Werte zwischen 0 und 1 standardisierte
        Raster zurück.
123     return outputWeight
124
```

```
125
126 # Funktion "RasterSelect" bietet die Moeglichkeit nur bestimmte Werte in der Ergebnis-Darstellung anzuzeigen:
127 # Als Eingabeparameter werden die zu selektierenden Raster und der Selektionsausdruck festgelegt.
128 def RasterSelect(rasterInput, whereClause):
129
130     # Da es sich um eine Multivalue-Eingabe handelt, wird der String gesplittet und somit eine Liste erzeugt.
131     listRasterSelect = rasterInput.split(";")
132
133     # Auch bei den Selektionsausdruecken handelt es sich um eine Multivalue-Eingabe, die gesplittet wird und eine Liste erzeugt wird.
134     listWhere = whereClause.split(";")
135
136     # Falls die Laenge der Liste der Raster und der Selektionsausdruecke unterschiedlich ist, wird eine Fehlermeldung ausgegeben und das Programm beendet.
137     if len(listRasterSelect) != len(listWhere):
138         arcpy.AddError("Die Anzahl der zu selektierenden Raster und der Selektionsausdruecke muss gleich sein")
139         exit(1)
140
141     # Ist ein Raster in der Liste vorhanden, werden aus diesem Werte, die dem Selektionsausdruck entsprechen, extrahiert und ein Raster
142     # mit den Zellwerten 1 erzeugt.
143     elif len(listRasterSelect) == 1:
144         outputExtract_1 = arcpy.sa.ExtractByAttributes(listRasterSelect[0], listWhere[0])
145         outCon = Con(outputExtract_1 >= 0, 1)
146
147     # Sind zwei Raster in der Liste vorhanden, werden aus diesen Werte, die den Selektionsausdruecken entsprechen (UND-Verknuepfung),
148     # extrahiert und ein Raster mit den Zellwerten 1 erzeugt.
149     elif len(listRasterSelect) == 2:
150         outputExtract_1 = arcpy.sa.ExtractByAttributes(listRasterSelect[0], listWhere[0])
151         outputExtract_2 = arcpy.sa.ExtractByAttributes(listRasterSelect[1], listWhere[1])
152         outCon = Con((outputExtract_1 >= 0) & (outputExtract_2 >= 0), 1)
153
154     # Sind drei Raster in der Liste vorhanden, werden aus diesen Werte, die den Selektionsausdruecken entsprechen (UND-Verknuepfung),
155     # extrahiert und ein Raster mit den Zellwerten 1 erzeugt.
156     elif len(listRasterSelect) == 3:
157         outputExtract_1 = arcpy.sa.ExtractByAttributes(listRasterSelect[0], listWhere[0])
158         outputExtract_2 = arcpy.sa.ExtractByAttributes(listRasterSelect[1], listWhere[1])
159         outputExtract_3 = arcpy.sa.ExtractByAttributes(listRasterSelect[2], listWhere[2])
160         outCon = Con((outputExtract_1 >= 0) & (outputExtract_2 >= 0) & (outputExtract_3 >= 0), 1)
161
162     # Sind vier Raster in der Liste vorhanden, werden aus diesen Werte, die den Selektionsausdruecken entsprechen (UND-Verknuepfung),
163     # extrahiert und ein Raster mit den Zellwerten 1 erzeugt.
164     elif len(listRasterSelect) == 4:
```

```
165             outputExtract_1 = arcpy.sa.ExtractByAttributes(listRasterSelect[0],
                listWhere[0])
166             outputExtract_2 = arcpy.sa.ExtractByAttributes(listRasterSelect[1],
                listWhere[1])
167             outputExtract_3 = arcpy.sa.ExtractByAttributes(listRasterSelect[2],
                listWhere[2])
168             outputExtract_4 = arcpy.sa.ExtractByAttributes(listRasterSelect[3],
                listWhere[3])
169             outCon = Con((outputExtract_1 >= 0) & (outputExtract_2 >= 0) & (
                outputExtract_3 >= 0) & (outputExtract_4 >= 0), 1)
170
171         # Funktion gibt als Ergebnis ein Raster mit den Zellwerten 1 zurueck,
            welches nur die Flaechen darstellt, die den Selektionsausdruecken
            entsprechen (UND-Verknuepfung).
172         return outCon
173
174
175     # Funktion "Buffer_1" wird benoetigt, wenn neben der Weissflaechenanalyse keine
        Rasterselektion stattfindet,
176     # ansonsten wird Funktion "Buffer_2" benoetigt (siehe Zeile 284 - 400).
177     # Als Eingabeparameter werden die zu puffernden Feature Classes und die
        Puffer-Distanzen festgelegt.
178     def Buffer_1(bufferFeature, bufferDistance):
179
180         # Da es sich um eine Multivalue-Eingabe handelt, wird der String gesplittet
            und somit eine Liste erzeugt.
181         listFeature = bufferFeature.split(";")
182
183         # Auch bei den Distanzen handelt es sich um eine Multivalue-Eingabe, die
            gesplittet wird und eine Liste erzeugt wird.
184         listDistance = bufferDistance.split(";")
185
186         # Falls die Laenge der Liste der Feature Classes und der Puffer-Distanzen
            unterschiedlich ist, wird eine Fehlermeldung ausgegeben und das Programm
            beendet.
187         if len(listFeature) != len(listDistance):
188             arcpy.AddError("Die Anzahl der Puffer-Distanzen muss jener der zu
                puffernden Features entsprechen und umgekehrt")
189             exit(1)
190
191         # Ansonsten beginnt der Puffer-Abschnitt.
192         else:
193             a = 0 # festlegen eines Zaehlers fuer die folgende for-Schleife.
194
195             # Beginn des Durchlaufens der Liste der Feature Classes.
196             for data in listFeature:
197
198                 # Erzeugen der euklidischen Distanzraster
199                 outEucDistance = EucDistance(listFeature[a],"", 25)
200                 outEucDistance.save(scratchPath+"\\"+"t7878raster"+str(a))
201                 eucRaster = Raster(scratchPath+"\\"+"t7878raster"+str(a))
202
203                 # Pixel innerhalb der Puffer-Distanz erhalten den Wert 0, alle
                    anderen den Wert 1.
204                 outConBuffer = Con(eucRaster <= int(listDistance[a]), 0, 1)
205
206                 # abspeichern der Raster.
207                 outConBuffer.save(scratchPath+"\\"+"traster"+str(a))
```

```
208
209             # erhoehen des Zaehlers um 1.
210             a = a+1
211
212
213         # erzeugen der Rasterliste aus den zweiwertigen Rastern (0 und 1).
214         rasterList = arcpy.ListRasters("traster*")
215
216         # Falls Liste nur aus einem Raster besteht, erzeuge NoData-Werte für
            die 0-wertigen Pixel, schneide Weissflaechen auf Untersuchungsgebiet zu
            und speicher das Ergebnis gemaeß Nutzereingabe ab.
217         if len(rasterList) == 1:
218
219             outConRaster = Raster(rasterList[0])
220             outConRasterFinal = Con(outConRaster == 1, outConRaster)
221             outConRasterFinal.save(scratchPath+"\\"+"Weiss_2")
222             weissRaster2 = Raster(scratchPath+"\\"+"Weiss_2")
223             extractMask = arcpy.sa.ExtractByMask(weissRaster2, ugAreaDissolve)
                # ugAreaDissolve siehe Zeile 665
224             extractMask.save(outWeissFlaeche)
225             arcpy.AddMessage("Weissflaechen gespeichert")
226
227
228         # ansonsten multipliziere die zweiwertigen Raster und wandle die
            0-Werte des Ergebnissrasters in NoData-Werte um, schneide Weissflaechen
            auf Untersuchungsgebiet zu und speichere das Ergebnis gemaeß
            Nutzereingabe ab.
229         else:
230             b = 0
231             for raster in rasterList:
232
233                 if b == 0:
234                     outRaster = Raster(raster)
235                     b += 1
236                 else:
237                     outRaster = outRaster * raster
238                     b += 1
239
240             outRaster.save(scratchPath+"\\"+"rasterWeiss")
241             weissRaster = Raster(scratchPath+"\\"+"rasterWeiss")
242             outConRasterFinal = Con(weissRaster == 1, weissRaster)
243             outConRasterFinal.save(scratchPath+"\\"+"Weiss_2")
244             weissRaster2 = Raster(scratchPath+"\\"+"Weiss_2")
245             extractMask = arcpy.sa.ExtractByMask(weissRaster2, ugAreaDissolve)
                # ugAreaDissolve siehe Zeile 665
246             extractMask.save(outWeissFlaeche)
247
248             # Benachrichtigungen und loeschen des Rasters aus dem
                Zwischenspeicher
249             arcpy.AddMessage("Weissflaechen gespeichert")
250
251
252     # Ausgabe der Funktion sind die Weissflaechen, die innerhalb des
        Untersuchungsgebietes liegen. Untersuchungsgebiet wird in Parameter(1)
        angegeben.
253     return extractMask
254
255
```

```
256  # Funktion "Buffer_2" ist fast ident zu Funktion "Buffer_1", wird jedoch
     benoetigt, wenn zusaetzlich zur Rasterselektion eine Weissflaechenanalyse
257  # stattfindet. Unterschiede zu Funktion "Buffer_1" bestehen dadurch, dass bei
     der Funktion "Buffer_2" am Ende noch die Maskenextraktion integriert ist.
258  def Buffer_2():
259
260      # Da es sich um eine Multivalue-Eingabe handelt, wird der String gesplittet
         und somit eine Liste erzeugt.
261      listFeature = bufferFeature.split(";")
262
263      # Auch bei den Distanzen handelt es sich um eine Multivalue-Eingabe, die
         gesplittet wird und eine Liste erzeugt wird.
264      listDistance = bufferDistance.split(";")
265
266      # Falls die Laenge der Liste der Feature Classes und der Puffer-Distanzen
         unterschiedlich ist, wird eine Fehlermeldung ausgegeben und das Programm
         beendet.
267      if len(listFeature) != len(listDistance):
268          arcpy.AddError("Die Anzahl der Puffer-Distanzen muss jener der zu
             puffernden Features entsprechen und umgekehrt")
269          exit(1)
270
271      # Ansonsten beginnt der Puffer-Abschnitt.
272      else:
273          a = 0 # festlegen eines Zaehlers fuer die folgende for-Schleife.
274
275          # Beginn des Durchlaufens der Liste der Feature Classes.
276          for data in listFeature:
277
278              # Erzeugen der euklidischen Distanzraster
279              outEucDistance = EucDistance(listFeature[a],"", 25)
280              outEucDistance.save(scratchPath+"\\"+"t7878raster"+str(a))
281              eucRaster = Raster(scratchPath+"\\"+"t7878raster"+str(a))
282
283              # Pixel innerhalb der Puffer-Distanz erhalten den Wert 0, alle
                 anderen den Wert 1.
284              outConBuffer = Con(eucRaster <= int(listDistance[a]), 0, 1)
285
286              # abspeichern der Raster.
287              outConBuffer.save(scratchPath+"\\"+"traster"+str(a))
288
289              # erhoehen des Zaehlers um 1.
290              a = a+1
291
292          # erzeugen der Rasterliste aus den zweiwertigen Rastern (0 und 1).
293          rasterList = arcpy.ListRasters("traster*")
294
295          # Falls Liste nur aus einem Raster besteht, erzeuge NoData-Werte für
             die 0-wertigen Pixel, schneide Weissflaechen auf Untersuchungsgebiet zu
             und speicher das Ergebnis gemaeß Nutzereingabe ab.
296          if len(rasterList) == 1:
297
298              outConRaster = Raster(rasterList[0])
299              outConRasterFinal = Con(outConRaster == 1, outConRaster)
300              outConRasterFinal.save(scratchPath+"\\"+"Weiss_2")
301              weissRaster2 = Raster(scratchPath+"\\"+"Weiss_2")
302              extractMask = arcpy.sa.ExtractByMask(weissRaster2, ugAreaDissolve)
                 # ugAreaDissolve siehe Zeile 665
```

```
303                 extractMask.save(outWeissFlaeche)
304                 arcpy.AddMessage("Weissflaechen gespeichert")
305 
306             # ansonsten multipliziere die zweiwertigen Raster und wandle die
                0-Werte des Ergebnissrasters in NoData-Werte um, schneide Weissflaechen
                auf Untersuchungsgebiet zu und speichere das Ergebnis gemaeß
                Nutzereingabe ab.
307             else:
308                 b = 0
309                 for raster in rasterList:
310 
311                     if b == 0:
312                         outRaster = Raster(raster)
313                         b += 1
314                     else:
315                         outRaster = outRaster * raster
316                         b += 1
317 
318                 outRaster.save(scratchPath+"\\"+"rasterWeiss")
319                 weissRaster = Raster(scratchPath+"\\"+"rasterWeiss")
320                 outConRasterFinal = Con(weissRaster == 1, weissRaster)
321                 outConRasterFinal.save(scratchPath+"\\"+"Weiss_2")
322                 weissRaster2 = Raster(scratchPath+"\\"+"Weiss_2")
323                 extractMask = arcpy.sa.ExtractByMask(weissRaster2, ugAreaDissolve)
                    # ugAreaDissolve siehe Zeile 665
324                 extractMask.save(outWeissFlaeche)
325 
326                 # Benachrichtigungen und loeschen des Rasters aus dem
                    Zwischenspeicher
327                 arcpy.AddMessage("Weissflaechen gespeichert")
328 
329 
330             # Durch die folgende ExtractByMask-Funktion wird das Ergebniss auf die
                Ausdehnung des Untersuchungsgebietes beschraenkt. "ugAreaDissolve" wird
                in Zeile 665 definiert.
331             extractMask = arcpy.sa.ExtractByMask(outWeissFlaeche, ugAreaDissolve)
332 
333             # Durch die erzeugte Weissflaechenmaske (extractMask), werden nur die
                selektierten Raster beruecksichtigt,
334             # die inerhalb der erzeugten Weissflaechen liegen.
335             outExtractByMask = arcpy.sa.ExtractByMask(scratchPath+"\\"+"outcon",
                extractMask)
336 
337             # Das Ergebnis dient als neue Maske und dient dazu nur diese Gebiete
                aus der Rastergewichtung zu extrahieren.
338             outExtractByMask_2 = arcpy.sa.ExtractByMask(outputWeight,
                outExtractByMask)
339 
340             # Das Ergebniss wird unter dem vom Nutzer definierten Namen und Zielort
                gespeichet.
341             outExtractByMask_2.save(outputData)
342 
343             # Nachricht, dass Funktion zu Ende.
344             arcpy.AddMessage("Ende der Funktion Buffer_2.")
345 
346 
347     # Funktion "ExtractMask_1" wird wiederholt verwendet wenn neben der
        Rastergewichtung auch Raster selektiert und eine Maske erzeugt wird.
```

```
348 def ExtractMask_1():
349
350             # Aufruf der Funktion "RasterSelect".
351             outCon = RasterSelect(rasterInput, whereClause)
352
353             # Durch die erzeugte Maske der selektierten Raster (outCon), werden
                nur die Rasterzellen des Gewichtungsergebnisses beruecksichtigt,
354             # bei welchen die Bedingungen der Rasterselektion erfuellt
                sind.
355             outExtractByMask = arcpy.sa.ExtractByMask(outputWeight, outCon)
356
357             # Das Ergebnis wird wiederum ueber die vom Benutzer definierte
                Eingabemaske (Parameter(10)) extrahiert.
358             outExtractByMask_2 = arcpy.sa.ExtractByMask(outExtractByMask,
                clipFeature)
359
360             # erneute Extraktion mit Polygon des Untersuchungsgebiets.
                "ugAreaDissolve" wird in Zeile 665 definiert.
361             outExtractByMask_3 = arcpy.sa.ExtractByMask(outExtractByMask_2,
                ugAreaDissolve)
362
363             # Abspeichern des Ergebnisses entsprechend der Nutzereingabe
                (Parameter(11)).
364             outExtractByMask_3.save(outputData)
365
366             # Aufruf der Funktion fuer die Kartenproduktion.
367             mapLayout()
368
369
370 # Funktion "ExtractMask_2" wird wiederholt verwendet wenn neben der
    Rastergewichtung auch eine Weissflaechenanalyse durchgeführt wird..
371 def ExtractMask_2():
372
373             # Aufruf der Funktion "Buffer_1".
374             extractMask = Buffer_1(bufferFeature, bufferDistance)
375
376             # Durch die erzeugte Weissflaechenmaske (extractMask), werden nur
                die selektierten Rasterwerte beruecksichtigt,
377             # die innerhalb der erzeugten Weissflaechen liegen.
378             outExtractByMask = arcpy.sa.ExtractByMask(outputWeight, extractMask)
379
380             # erneute Extraktion mit Polygon des Untersuchungsgebiets.
                "ugAreaDissolve" wird in Zeile 665 definiert.
381             outExtractByMask_2 = arcpy.sa.ExtractByMask(outExtractByMask,
                ugAreaDissolve)
382
383             # Abspeichern des Ergebnisses entsprechend der Nutzereingabe
                (Parameter(11)).
384             outExtractByMask_2.save(outputData)
385
386             # Aufruf der Funktion fuer die Kartenproduktion.
387             mapLayout()
388
389
390 # Funktion "ExtractMask_3" wird wiederholt verwendet wenn neben der
    Rastergewichtung auch Raster selektiert werden und zusaetzlich eine
    Weissflaechenanalyse durchgeführt wird.
391 def ExtractMask_3():
```

```
392
393             # Ist ein Raster in der Liste vorhanden, werden aus diesem Werte,
                die dem Selektionsausdruck entsprechen, extrahiert und ein Raster
394             # mit den Zellwerten 1 erzeugt.
395             if len(listRasterSelect) == 1:
396                 outputExtract_1 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    0], listWhere[0])
397                 outCon = Con(outputExtract_1 >= 0, 1)
398                 outCon.save(scratchPath+"\\"+"outcon")
399
400             # Sind zwei Raster in der Liste vorhanden, werden aus diesen Werte,
                die den Selektionsausdruecken entsprechen (UND-Verknuepfung),
401             # extrahiert und ein Raster mit den Zellwerten 1 erzeugt.
402             elif len(listRasterSelect) == 2:
403                 outputExtract_1 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    0], listWhere[0])
404                 outputExtract_2 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    1], listWhere[1])
405                 outCon = Con((outputExtract_1 >= 0) & (outputExtract_2 >= 0) , 1)
406                 outCon.save(scratchPath+"\\"+"outcon")
407
408             # Sind drei Raster in der Liste vorhanden, werden aus diesen Werte,
                die den Selektionsausdruecken entsprechen (UND-Verknuepfung),
409             # extrahiert und ein Raster mit den Zellwerten 1 erzeugt.
410             elif len(listRasterSelect) == 3:
411                 outputExtract_1 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    0], listWhere[0])
412                 outputExtract_2 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    1], listWhere[1])
413                 outputExtract_3 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    2], listWhere[2])
414                 outCon = Con((outputExtract_1 >= 0) & (outputExtract_2 >= 0) & (
                    outputExtract_3 >= 0), 1)
415                 outCon.save(scratchPath+"\\"+"outcon")
416
417             # Sind vier Raster in der Liste vorhanden, werden aus diesen Werte,
                die den Selektionsausdruecken entsprechen (UND-Verknuepfung),
418             # extrahiert und ein Raster mit den Zellwerten 1 erzeugt.
419             elif len(listRasterSelect) == 4:
420                 outputExtract_1 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    0], listWhere[0])
421                 outputExtract_2 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    1], listWhere[1])
422                 outputExtract_3 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    2], listWhere[2])
423                 outputExtract_4 = arcpy.sa.ExtractByAttributes(listRasterSelect[
                    3], listWhere[3])
424                 outCon = Con((outputExtract_1 >= 0) & (outputExtract_2 >= 0) & (
                    outputExtract_3 >= 0) & (outputExtract_4 >= 0), 1)
425                 outCon.save(scratchPath+"\\"+"outcon")
426
427             # Aufruf der beiden Funktion "Buffer_2" und "mapLayout"
428             Buffer_2()
429             mapLayout()
430
431 # Funktion "mapLayout" ist fuer die Gestaltung der Karte, einschliesslich
    pdf-Produktion verantwortlich.
432 # Eingabeparameter nicht erforderlich.
```

```
433   def mapLayout ():
434
435               # Bezugnahme auf geoeffnetes mxd.
436               mxd = arcpy.mapping.MapDocument("CURRENT")
437
438               # Auflisten der Dataframe-Elemente.
439               df = arcpy.mapping.ListDataFrames(mxd)[0]
440
441               # Erzeugung eines Layers  aus dem Eingabeparameter "ugAreaDissolve"
                  (siehe auch Zeile 665) und abspeichern dieses Layers.
442               layerUG = arcpy.MakeFeatureLayer_management(ugAreaDissolve,
                  "Untersuchungsgebiet")
443               outLayerUG =
                  "C:\\Users\\Matsche\\Documents\\Matsche\\Master\\Masterarbeit\\Daten_
                  SDSS\\Untersuchungsgebiet"
444               layerUGSave = arcpy.SaveToLayerFile_management(layerUG, outLayerUG)
445
446               # Bezugnahme auf erzeugten Layer und Hinzufuegen des Layers an
                  oberster Position des Dataframe.
447               addLayerUG = arcpy.mapping.Layer(outLayerUG+".lyr")
448               arcpy.mapping.AddLayer(df, addLayerUG, "TOP")
449
450               # Auflisten des hinzugefuegten Layers und Uebernahme der Symbologie
                  aus einem vorher manuell erzeugten Layer
                  ("untersuchungsgebietSymbology.lyr).
451               lyrUG = arcpy.mapping.ListLayers(mxd)[0]
452               lyrFileUG = arcpy.mapping.Layer(
                  "C:\\Users\\Matsche\\Documents\\Matsche\\Master\\Masterarbeit\\Daten_
                  SDSS\\untersuchungsgebietSymbology.lyr")
453               arcpy.ApplySymbologyFromLayer_management(lyrUG, lyrFileUG)
454
455               # Erzeugung eines Layers  aus dem Ausgabeparameter
                  (Ergebniss-Datei) "outputData" und abspeichern dieses Layers.
456               layerRaster = arcpy.MakeRasterLayer_management(outputData,
                  "RasterLayer")
457               outLayer =
                  "C:\\Users\\Matsche\\Documents\\Matsche\\Master\\Masterarbeit\\Daten_
                  SDSS\\RasterLayer"
458               layerSave = arcpy.SaveToLayerFile_management(layerRaster, outLayer)
459
460               # Bezugnahme auf neu erzeugten Layer und Hinzufuegen des Layers an
                  oberster Position des Dataframe.
461               # Rasterlayer wird ueber zuvor erzeugten Layer "addLayerUG"
                  positioniert.
462               addLayer = arcpy.mapping.Layer(outLayer+".lyr")
463               arcpy.mapping.AddLayer(df, addLayer, "TOP")
464
465               # Auflisten des hinzugefuegten Layers und Uebernahme der Symbologie
                  aus einem vorher manuell erzeugten Layer ("rasterSymbology.lyr).
466               lyrRaster = arcpy.mapping.ListLayers(mxd)[0]
467               lyrFile = arcpy.mapping.Layer(
                  "C:\\Users\\Matsche\\Documents\\Matsche\\Master\\Masterarbeit\\Daten_
                  SDSS\\rasterSymbology.lyr")
468               arcpy.ApplySymbologyFromLayer_management(lyrRaster, lyrFile)
469
470               # Auflisten des Titelelements.
471               elm = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT", "TITEL"
                  )[0]
```

```
472
473             # Definieren des Titels durch Eingabeparameter (Parameter(2)).
474             mxd.title = "Windkraft-Eignungsgebiete: "+nameUG
475
476             # Festlegen der Breite des Titelelements.
477             elmWidth = 28.1
478
479             # Festlegen der Start-Schriftgroesse des Textes.
480             x = 35
481             elm.text = '<dyn type="document" property="title"/>'
482             elm.fontSize = x
483
484             # Solange Breite des Textes groesser als jene des Titelelemenets,
                wird Schriftgroeesse schrittweise verkleinert.
485             while elm.elementWidth > float(elmWidth):
486                 elm.fontSize = x
487                 x = x - 0.5
488
489             # Festlegen der Hoehe des Titelelements.
490             elmHeight = 1.2
491
492             # Festlegen der Start-Schriftgroeße des Textes.
493             y = 35
494             elm.text = '<dyn type="document" property="title"/>'
495             elm.fontSize = x
496
497             # Solange Hoehe des Textes groesser als jene des Titelelemenets,
                wird Schriftgroesse schrittweise verkleinert.
498             while elm.elementHeight > float(elmHeight):
499                 elm.fontSize = y
500                 y = y - 0.5
501
502             # Positionierung des Titelelements.
503             elm.elementPositionX = 0.75
504             elm.elementPositionY = 19.85
505
506             # Auflisten des Legendenelements.
507             leg = arcpy.mapping.ListLayoutElements(mxd, "LEGEND_ELEMENT",
                "Legend")[0]
508
509             # Festlegen der Schriftgroeße und der Schriftart.
510             leg.title = '<FNT name="Arial" size="14"><BOL>Grad der
                Eignung*</BOL></FNT>'
511
512             # Positionierung des Legendenelements.
513             leg.elementPositionX = 23.1
514             leg.elementPositionY = 18.8
515
516             # Veraendern des Legenden Styles. Funktioniert nur, wenn der Layer
                nochmals extra aufgelistet wird. Mit der Variable "lyrFile"
                funktioniert es nicht
517             styleItem = arcpy.mapping.ListStyleItems(
                "C:\\Users\\Matsche\\AppData\\Roaming\\ESRI\\Desktop10.1\\ArcMap\\Mat
                sche.style", "Legend Items", "LegendStyle")[0]
518             lyr_2 = arcpy.mapping.ListLayers(mxd, 'RasterLayer', df)[0]
519             leg.updateItem(lyr_2, styleItem)
520
521             # Abfragen der Ausdehnung des Rasterlayers (Ergebnislayer).
```

```
522             ext = lyrRaster.getExtent()
523
524             # Dataframe auf Ausdehnung des Rasterlayers setzen.
525             df.extent = ext
526
527             # Veraenderung der Transparenz des Rasterlayers.
528             lyrRaster.transparency = 50
529
530             # Erzeugung einer Liste mit den gewichteten Rastern (Parameter(3)).
531             listRasterWeight = rasterWeight.split(";")
532
533             # Erzeugung einer Liste mit den vergebenen Gewichten (Parameter(4)).
534             listWeights = weightsRaster.split(";")
535
536             # Auflisten der Textelemente mit Namensbeginn "Text1+". Es handelt
                sich hier um die Textelemente,
537             # die die gewichteten Raster bzw. die vergebenen Gewichte
                wiederspiegeln.
538             textelm = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT",
                "Text1*")
539
540             x = 0 # Festlegen eines Zaehlers für die folgende for-Schleife.
541             position = 7.6 # Festlegen einer festen Positionierung
542             for weights in listWeights: # Initialisierung der for Schleife,
                welche die Liste der Gewichte durchlaeuft.
543
544                 # Namen fuer Listenelemente der zu gewichteten Raster festlegen.
545                 rasterdaten = str(listRasterWeight[x])
546
547                 # Festlegen des Textelement-Textes. Text setzt sich aus
                    Dateinamen des jeweils gewichteten Rasters
548                 # und dem dazugehoerigen Gewicht zusammen
549                 textelm[x].text = "Gewichtung "+rasterdaten[92:]+": "+str(
                    weights)
550
551                 # Positionierung der Textelemente.
552                 textelm[x].elementPositionX = 23.1
553                 textelm[x].elementPositionY = position
554                 x = x+1 # Erhoehung des Zaehlers.
555
556                 position = position-0.7 # naechstes Textelement wird 0.7
                    Einheiten weiter unten positioniert.
557
558             # Falls nur zwei Raster gewichtet wurden, bleiben die anderen
                beiden Textelemente leer, werden aber positioniert.
559             if x == 2:
560                 textelm[2].text = " "
561                 textelm[2].elementPositionX = 23.1
562                 textelm[2].elementPositionY = 6.2
563                 textelm[3].text = " "
564                 textelm[3].elementPositionX = 23.1
565                 textelm[3].elementPositionY = 5.5
566
567             # Falls drei Raster gewichtet wurden, bleiben die anderen beiden
                Textelemente leer, werden aber positioniert.
568             if x == 3:
569                 textelm[3].text = " "
570                 textelm[3].elementPositionX = 23.1
```

```
571                     textelm[3].elementPositionY = 5.5
572
573                 # Erzeugung einer Liste mit den selktierten Rastern (Parameter(5)).
574                 listRasterSelect = rasterInput.split(";")
575
576                 # Erzeugung einer Liste mit den Selektionsausdruecken (Parameter(6)).
577                 listWhere = whereClause.split(";")
578
579                 y = 0 # Festlegen eines Zaehlers für die folgende for-Schleife.
580
581                 # Falls mind. ein Raster selektiert wurde.
582                 if rasterInput is not "":
583
584                     # Liste die Textelemente auf, die mit "Text2*" beginnen
585                     textelm_2 = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT"
                        , "Text2*")
586
587                     position_2 = 4.7 # Festlegen einer festen Positionierung
588
589                     for where in listWhere: # Initialisierung der for Schleife,
                        welche die Liste der Selektionsausdruecke durchlaeuft.
590
591                         # Namen fuer Listenelemente der zu selektierenden Raster
                            festlegen.
592                         whereData = str(listRasterSelect[y])
593
594                         # Festlegen des Textelement-Textes. Text setzt sich aus
                            Dateinamen des jeweils selektierten Rasters
595                         # und dem dazugehoerigen Selektionsausdruck zusammen
596                         textelm_2[y].text = whereData[90:]+": "+str(where)[5:]
597
598                         # Positionierung der Textelemente.
599                         textelm_2[y].elementPositionX = 23.1
600                         textelm_2[y].elementPositionY = position_2
601                         y = y+1 # Erhoehung des Zaehlers.
602
603
604                         position_2 = position_2-0.7 # naechstes Textelement wird
                            0.7 Einheiten weiter unten positioniert.
605
606                     # Falls ein Raster selektiert wurde, bleiben die anderen
                        Textelemente leer, werden aber positioniert.
607                     if y == 1:
608                         textelm_2[1].text = " "
609                         textelm_2[1].elementPositionX = 23.1
610                         textelm_2[1].elementPositionY = 4.0
611                         textelm_2[2].text = " "
612                         textelm_2[2].elementPositionX = 23.1
613                         textelm_2[2].elementPositionY = 3.3
614                         textelm_2[3].text = " "
615                         textelm_2[3].elementPositionX = 23
616                         textelm_2[3].elementPositionY = 2.6
617
618                     # Falls zwei Raster selektiert wurden, bleiben die anderen
                        Textelemente leer, werden aber positioniert.
619                     if y == 2:
620                         textelm_2[2].text = " "
621                         textelm_2[2].elementPositionX = 23.1
```

```
622                     textelm_2[2].elementPositionY = 3.3
623                     textelm_2[3].text = " "
624                     textelm_2[3].elementPositionX = 23.1
625                     textelm_2[3].elementPositionY = 2.6
626
627                 # Falls drei Raster selektiert wurden, bleibt das letzte
                    Textelement leer, wird aber positioniert.
628                 if y == 3:
629                     textelm_2[3].text = " "
630                     textelm_2[3].elementPositionX = 23.1
631                     textelm_2[3].elementPositionY = 2.6
632
633             # Falls kein Raster selektiert wurde, bleiben alle weiteren
                Textelement leer, werden aber positioniert.
634             if y == 0:
635                 textelm_2 = arcpy.mapping.ListLayoutElements(mxd, "TEXT_ELEMENT"
                    , "Text2*")
636                 textelm_2[0].text = " "
637                 textelm_2[0].elementPositionX = 23.1
638                 textelm_2[0].elementPositionY = 4.7
639                 textelm_2[1].text = " "
640                 textelm_2[1].elementPositionX = 23
641                 textelm_2[1].elementPositionY = 4.0
642                 textelm_2[2].text = " "
643                 textelm_2[2].elementPositionX = 23.1
644                 textelm_2[2].elementPositionY = 3.3
645                 textelm_2[3].text = " "
646                 textelm_2[3].elementPositionX = 23.1
647                 textelm_2[3].elementPositionY = 2.6
648
649             # Update Active-View und Table of Contents
650             arcpy.RefreshTOC()
651             arcpy.RefreshActiveView()
652
653             # Export mxd als pdf. Zielordner und Name wird von Nutzer
                festgelegt (Parameter(12))
654             arcpy.mapping.ExportToPDF(mxd, outputPDF)
655
656             # Entfernung der Objekt-Referenzierung
657             del mxd
658
659
660
661 # Beginn des try-Blocks:
662 try:
663
664     # Zu Beginn wird die Untersuchungsgebiets Feature Class in ein einzelnes
        Polygon aufgeloest: wichtig fuer Ergebnisdarstellung in der Karte
665     ugAreaDissolve =
        "C:\\Users\\Matsche\\Documents\\Matsche\\Master\\Masterarbeit\\Daten_SDSS\\Re
        sult.gdb\\UGebiet"
666     arcpy.Dissolve_management(ugArea, ugAreaDissolve)
667
668
669     # Definition des Prozessablaus, falls lediglich Raster gewichtet
        werden.
670     if rasterWeight is not "" and rasterInput is "" and bufferFeature is "":
671
```

```
672             # Aufruf der Funktion "RasterGewicht".
673             outputWeight = RasterGewicht(rasterWeight, weightsRaster)
674
675             # Ausgabe der Funktion "RasterGewicht" und Extraktion mit Maske aus
                Eingabeparameter (Parameter(10)).
676             outWeight_2 = arcpy.sa.ExtractByMask(outputWeight, clipFeature)
677
678             # erneute Extraktion mit Polygon des Untersuchungsgebiets.
679             outWeight_3 = arcpy.sa.ExtractByMask(outWeight_2, ugAreaDissolve)
680
681             # Abspeichern des Ergebnisses entsprechend der Nutzereingabe
                (Parameter(11)).
682             outWeight_3.save(outputData)
683
684             # Aufruf der Funktion fuer die Kartenproduktion.
685             mapLayout()
686
687
688         # Definition des Prozessablaus, falls Raster gewichtet und selektiert werden.
689         elif rasterWeight is not "" and rasterInput is not "" and bufferFeature is
            "":
690
691             # Da es sich bei den zu selektierenden Rastern eine Multivalue-Eingabe
                handelt, wird der String gesplittet und somit eine Liste erzeugt.
692             listRasterSelect = rasterInput.split(";")
693
694             # Auch bei den Selektionsausdruecken handelt es sich um eine
                Multivalue-Eingabe, die gesplittet wird und eine Liste erzeugt wird.
695             listWhere = whereClause.split(";")
696
697             # Da es sich bei den zu gewichtenden Rastern eine Multivalue-Eingabe
                handelt, wird der String gesplittet und somit eine Liste erzeugt.
698             listRasterWeight = rasterWeight.split(";")
699
700             # Auch bei den Gewichten handelt es sich um eine Multivalue-Eingabe,
                die gesplittet wird und eine Liste erzeugt wird.
701             listWeights = weightsRaster.split(";")
702
703             # Falls die Laenge der Liste der Raster und der Rastergewichte
                unterschiedlich ist, wird eine Fehlermeldung ausgegeben und das
                Programm beendet.
704             if len(listRasterWeight) != len(listWeights):
705                 arcpy.AddError("Die Anzahl der zu gewichtenden Raster und der
                    Rastergewichte muss gleich sein")
706                 exit(1)
707
708             # Falls die Rasterliste aus nur einem Raster besteht, wird ebenfalls
                eine Fehlermeldung ausgegeben und das Programm beendet.
709             elif len(listRasterWeight) == 1:
710                 arcpy.AddError("Es muessen mindestens 2 Raster gewichtet werden")
711                 exit(1)
712
713             # Ist die Listenlaenge gleich 2 werden die reklassifizierte Raster mit
                ihren gewichten multipliziert und addiert. Durch die Division wird das
714             # Ergebnis auf Werte zwischen 0 und 11 standardisiert.
715             elif len(listRasterWeight) == 2:
716                 outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                    Raster(listRasterWeight[1]) * int(listWeights[1]))/(int(listWeights[
```

```
                0]) + int(listWeights[1]))
717
718         # Ist die Listenlaenge gleich 3 werden die reklassifizierte Raster mit
            ihren gewichten multipliziert und addiert. Durch die Division wird das
719         # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
720         elif len(listRasterWeight) == 3:
721             outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                Raster(listRasterWeight[1]) * int(listWeights[1]) + Raster(
                listRasterWeight[2]) * int(listWeights[2]))/(int(listWeights[0]) +
                int(listWeights[1]) + int(listWeights[2]))
722
723         # Ist die Listenlaenge gleich 4 werden die reklassifizierte Raster mit
            ihren gewichten multipliziert und addiert. Durch die Division wird das
724         # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
725         elif len(listRasterWeight) == 4:
726             outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                Raster(listRasterWeight[1]) * int(listWeights[1]) + Raster(
                listRasterWeight[2]) * int(listWeights[2]) + Raster(listRasterWeight
                [3]) * int(listWeights[3]))/(int(listWeights[0]) + int(listWeights[1
                ]) + int(listWeights[2]) + int(listWeights[3]))
727
728         # Aufruf der Funktion "ExtractMask_1"
729         ExtractMask_1()
730
731
732     # Definition des Prozessablaus, falls Raster gewichtet und
        Weissflaechenanalyse durchgefuehrt wird.
733     elif rasterWeight is not "" and rasterInput is "" and bufferFeature is not
        "":
734
735         # Da es sich bei den zu gewichtenden Rastern eine Multivalue-Eingabe
            handelt, wird der String gesplittet und somit eine Liste erzeugt.
736         listRasterWeight = rasterWeight.split(";")
737
738         # Auch bei den Gewichten handelt es sich um eine Multivalue-Eingabe,
            die gesplittet wird und eine Liste erzeugt wird.
739         listWeights = weightsRaster.split(";")
740
741         # Falls die Laenge der Liste der Raster und der Rastergewichte
            unterschiedlich ist, wird eine Fehlermeldung ausgegeben und das
            Programm beendet.
742         if len(listRasterWeight) != len(listWeights):
743             arcpy.AddError("Die Anzahl der zu gewichtenden Raster und der
                Rastergewichte muss gleich sein")
744             exit(1)
745
746         # Falls die Rasterliste aus nur einem Raster besteht, wird ebenfalls
            eine Fehlermeldung ausgegeben und das Programm beendet.
747         elif len(listRasterWeight) == 1:
748             arcpy.AddError("Es muessen mindestens 2 Raster gewichtet werden")
749             exit(1)
750
751         # Ist die Listenlaenge gleich 2 werden die reklassifizierte Raster mit
            ihren gewichten multipliziert und addiert. Durch die Division wird das
752         # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
753         elif len(listRasterWeight) == 2:
754             outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                Raster(listRasterWeight[1]) * int(listWeights[1]))/(int(listWeights[
```

```
                0]) + int(listWeights[1]))
755
756         # Ist die Listenlaenge gleich 3 werden die reklassifizierte Raster mit
            ihren gewichten multipliziert und addiert. Durch die Division wird das
757         # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
758         elif len(listRasterWeight) == 3:
759             outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                Raster(listRasterWeight[1]) * int(listWeights[1]) + Raster(
                listRasterWeight[2]) * int(listWeights[2]))/(int(listWeights[0]) +
                int(listWeights[1]) + int(listWeights[2]))
760
761         # Ist die Listenlaenge gleich 4 werden die reklassifizierte Raster mit
            ihren gewichten multipliziert und addiert. Durch die Division wird das
762         # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
763         elif len(listRasterWeight) == 4:
764             outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                Raster(listRasterWeight[1]) * int(listWeights[1]) + Raster(
                listRasterWeight[2]) * int(listWeights[2]) + Raster(listRasterWeight
                [3]) * int(listWeights[3]))/(int(listWeights[0]) + int(listWeights[1
                ]) + int(listWeights[2]) + int(listWeights[3]))
765
766         # Aufruf der Funktion "ExtractMask_2"
767         ExtractMask_2()
768
769     # Definition des Prozessablaus, falls Raster gewichtet und selektiert
        werden und Weissflaechenanalyse durchgefuehrt wird.
770     elif rasterWeight is not "" and rasterInput is not "" and bufferFeature is
        not "":
771
772         # Da es sich bei den zu selektierenden Rastern eine Multivalue-Eingabe
            handelt, wird der String gesplittet und somit eine Liste erzeugt.
773         listRasterSelect = rasterInput.split(";")
774
775         # Auch bei den Selektionsausdruecken handelt es sich um eine
            Multivalue-Eingabe, die gesplittet wird und eine Liste erzeugt wird.
776         listWhere = whereClause.split(";")
777
778         # Da es sich bei den zu gewichtenden Rastern eine Multivalue-Eingabe
            handelt, wird der String gesplittet und somit eine Liste erzeugt.
779         listRasterWeight = rasterWeight.split(";")
780
781         # Auch bei den Gewichten handelt es sich um eine Multivalue-Eingabe,
            die gesplittet wird und eine Liste erzeugt wird.
782         listWeights = weightsRaster.split(";")
783
784         # Da es sich bei den zu puffernden Feature Classes um eine
            Multivalue-Eingabe handelt, wird der String gesplittet und somit eine
            Liste erzeugt.
785         listFeature = bufferFeature.split(";")
786
787         # Auch bei den Distanzen handelt es sich um eine Multivalue-Eingabe,
            die gesplittet wird und eine Liste erzeugt wird.
788         listDistance = bufferDistance.split(";")
789
790         # Falls die Laenge der Liste der Raster und der Rastergewichte
            unterschiedlich ist, wird eine Fehlermeldung ausgegeben und das
            Programm beendet.
791         if len(listRasterWeight) != len(listWeights):
```

```
792             arcpy.AddError("Die Anzahl der zu gewichtenden Raster und der
                Rastergewichte muss gleich sein")
793             exit(1)
794
795         # Falls die Laenge der Liste der zu puffernden Feature Classes und der
            Distanzen unterschiedlich ist, wird eine Fehlermeldung ausgegeben und
            das Programm beendet.
796         elif len(listFeature) != len(listDistance):
797             arcpy.AddError("Die Anzahl der Puffer-Distanzen muss jener der zu
                puffernden Features entsprechen und umgekehrt")
798             exit(1)
799
800         #  Falls die Laenge der Liste der Raster und der Selektionsausdruecke
            unterschiedlich ist, wird eine Fehlermeldung ausgegeben und das
            Programm beendet.
801         elif len(listRasterSelect) != len(listWhere):
802             arcpy.AddError("Die Anzahl der zu selektierenden Raster und der
                Selektionsausdrücke muss gleich sein")
803             exit(1)
804
805         # Falls die Rasterliste aus nur einem Raster besteht, wird ebenfalls
            eine Fehlermeldung ausgegeben und das Programm beendet.
806         elif len(listRasterWeight) == 1:
807             arcpy.AddError("Es muessen mindestens 2 Raster gewichtet werden")
808             exit(1)
809
810         # Ist die Listenlaenge gleich 2 werden die reklassifizierte Raster mit
            ihren gewichten multipliziert und addiert. Durch die Division wird das
811         # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
812         elif len(listRasterWeight) == 2:
813             outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                Raster(listRasterWeight[1]) * int(listWeights[1]))/(int(listWeights[
                0]) + int(listWeights[1]))
814
815         # Ist die Listenlaenge gleich 2 werden die reklassifizierte Raster mit
            ihren gewichten multipliziert und addiert. Durch die Division wird das
816         # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
817         elif len(listRasterWeight) == 3:
818             outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                Raster(listRasterWeight[1]) * int(listWeights[1]) + Raster(
                listRasterWeight[2]) * int(listWeights[2]))/(int(listWeights[0]) +
                int(listWeights[1]) + int(listWeights[2]))
819
820         # Ist die Listenlaenge gleich 2 werden die reklassifizierte Raster mit
            ihren gewichten multipliziert und addiert. Durch die Division wird das
821         # Ergebnis auf Werte zwischen 0 und 1 standardisiert.
822         elif len(listRasterWeight) == 4:
823             outputWeight = (Raster(listRasterWeight[0]) * int(listWeights[0]) +
                Raster(listRasterWeight[1]) * int(listWeights[1]) + Raster(
                listRasterWeight[2]) * int(listWeights[2]) + Raster(listRasterWeight
                [3]) * int(listWeights[3]))/(int(listWeights[0]) + int(listWeights[1
                ]) + int(listWeights[2]) + int(listWeights[3]))
824
825         # Aufruf der Funktion "ExtractMask_3"
826         ExtractMask_3()
827
828     arcpy.Delete_management(scratchPath)
829     arcpy.AddMessage("in_memory gelöscht")
```

```
830
831   # Ende des try-Blocks
832
833   # Beginn des except-Blocks
834   except Exception as e:
835       print e.message
836
837       # Ausgabe Fehlermeldung
838       arcpy.AddError(e.message)
839
840   # Ende des Skripts
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
```