Jade Hochschule

Fachbereich Bauwesen Geoinformation Gesundheitstechnologie

Studiengang: B. Sc. Angewandte Geodäsie

Sommersemester 2022/23

Bachelorarbeit zum Thema:

Automatisierung des Aktualisierungsprozesses von umfangreichen Rasterdaten-Sammlungen - Ein Vergleich der Umsetzung mit ArcGIS Pro (Mosaic Datasets) und QGIS (GDAL Virtual Raster)

Automatisation of the update process of large raster data collections - A comparison of the implementation using ArcGIS Pro (Mosaic Datasets) and QGIS (GDAL Virtual Raster)

Vorgelegt von:

Moritz Hackenberg
Nordkampen 54
29664 Walsrode

Moritz.hackenberg@student.jade-hs.de
Tel.: +49 15733736453
Matrikelnummer: 6033436
Fachsemester: 8

Abgabedatum: 16.08.2023

Erstprüfer:

Andreas Gollenstede
Jade Hochschule Oldenburg

Zweitprüfer:

Prof. Dr. Thomas Brinkhoff
Jade Hochschule Oldenburg

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Listing-Verzeichnis

Abkürzungsverzeichnis

AdV	Arbeitsgemeinschaft der Vermessungsverwaltungen
API	Application Programming Interface
ATKIS	Amtliches Topographisch-Kartographisches Informationssystem
BKG	Bundesamt für Geodäsie und Kartografie
COM	Component Object Model (Microsoft)
cURL	Client for Uniform Ressource Locator
DGM	Digitales Geländemodell
DLM	Digitales Landschaftsmodell
Esri	Environmental Systems Research Institute
fGDB	File Geodatabase
DLM	Digitales Landschaftsmodell
DOP	Digitales Orthophoto
DTK	Digitale Topografische Karte
GDAL	Geospatial Data Abstraction Library

GIS	Geoinformationssystem
gpkg	Geo Package (Format)
GRASS	Geographic Resources Analysis Support System
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IBM	International Business Machines
LAMP	Linux, Apache, MySQL, PHP/Perl/Python
LZW	Lempel-Ziv-Welch
LZ77	Lempel-Ziv 77
MD	Mosaic Dataset
MDCS	Mosaic Dataset Configuration Script
MIT	Massachusetts Institute of Technology
NASA	National Aeronautics and Space Administration
PaaS	Platform as a Service
OGC	Open Geospatial Consortium
OGR	OpenGIS Simple Features Reference Implementation
OSGeo	Open Source Geospatial Foundation
RDBMS	Relational Database Management System
SaaS	Software as a Service
SDE	Spatial Database Engine
shp	Shape
SQL	Structured Query Language
SRS	Spatial Reference System
TIFF	Tagged Image File Format
UTM	Universal Transverse Mercator
VRT	Virtual Format / Virtual Raster Dataset
WCS	Web Coverage Service
WMS	Web Mapping Service
WPS	Web Processing Service
XML	Extensible Markup Language

1 Einführung

Wenn bestimmte Bereiche großer Rasterdatensammlungen mit einer regelmäßigen Frequenz aktualisiert werden müssen, stellt sich die Frage, wie dieser Prozess auf effiziente Art automatisiert werden kann. Für die Effizienz ist vor allem wichtig, dass die Rasterdaten beim Aktualisierungsprozess nicht mehrfach kopiert werden, sondern ein einziger physischer Speicherort referenziert wird. Bei ArcGIS Pro kann dies mit sogenannten *Mosaic Datasets*, beim quelloffenem QGIS mit dem *Virtual Format* der Geospatial Data Abstraction Library (GDAL), realisiert werden. Die Automatisierung des Workflows mit diesen Referenzierungsmethoden wird durch Python-Skripte und Nutzung der Bibliotheken *ArcPy* und *PyQGIS* implementiert.

Die Umsetzung des Verfahrens wird am Beispiel der Aktualisierung der Digitalen Topografischen Karte (DTK), wie sie bei der Datenintegration des Bundesamtes für Geodäsie und Kartografie (BKG) durchgeführt wird, im Detail untersucht. Außerdem wird das Skript für die Aktualisierung der integrierten Datenbestände von Digitalen Orthophotos (DOP) erweitert. Dabei sind verschiedene technische Randbedingungen durch die spezifischen Gegebenheiten des Arbeitsablaufs beim BKG zu berücksichtigen. Solche und ähnliche Rahmenbedingungen sind aber bei vielen Aktualisierungsprozessen von Rasterdatensammlungen zu erwarten.

Die Implementierungen mit ArcGIS und QGIS sollen hinsichtlich der Handhabung und Komplexität, Laufzeit, Performanz sowie allgemeiner wirtschaftlicher Aspekte verglichen werden. Im Ausblick werden die Möglichkeiten, diese Workflows serverseitig als Geodienste auszuführen, skizziert. Dies hätte den Vorteil der vollständigen Automatisierung durch die automatische Ausführung der Skripte beim Dateneingang.

2 Grundlagen Geoinformationssysteme (GIS)

Die Aktualisierung der Rasterdatensammlungen wird mit den Programmen ArcGIS und QGIS realisiert. Ein GIS ist ein computergestütztes System, mit welchem georeferenzierte Daten erfasst und vorbereitet, verwaltet, gespeichert und gewartet, editiert, analysiert und repräsentiert werden können (HUISMAN u. DE BY 2009: 32). Der Begriff geht ursprünglich auf das 1963 vom Geografen Roger Tomlinson entwickelte Canada Geographic Information System zurück, welches auf IMB Mainframe-Rechnern betrieben wurde. In Kanada kam aufgrund der großen Fläche, ökologischen Schätze und natürlichen Ressourcen des Landes Anfang der 60er-Jahre der Bedarf für eine schnelle und effiziente Handhabung großer Geodatensammlungen auf (ZEILE et al 2020: 493). Für Entscheidungen über eine zweckmäßige Landnutzung wurden detaillierte räumliche Informationen mit Karten der Maßstäbe 1:250.000 bis 1:20.000 gebraucht, deren manuelle Analyse einen immensen Zeit- und Arbeitsaufwand bedeutete (BRINKHOFF 2022: 15).

GIS lassen sich heute in die drei Hauptkategorien Desktop-, verteilte (Server-) und Cloud-basierte GIS einteilen (BILL, 2016: 144 ff.). Letztere wiederum lassen sich, gemäß der allgemeinen Unterscheidung von Cloud-Diensten, in *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) und *Software as a Service* (SaaS) einteilen. Bei IaaS stellt der Dienstleister Rechenressourcen in Form von Speicher, Netzwerk, Servern und Virtualisierung zur Verfügung. Die Cloud kann je nach Form der Hardware-Infrastruktur öffentlich oder privat sein. Bei PaaS wird eine Plattform für die textuelle programmatische (z.B. Google Earth Engine und ArcGIS Online mit Javascript) oder visuelle Entwicklung mit grafischer Oberfläche (ArcGIS Online Web Builder) von Software-Paketen und Applikationen, bei SaaS dagegen eine Anwendungssoftware selbst zur Verfügung gestellt (z.B. Google Mail, ArcGIS Online).

2.1 Proprietäre und freie Software

Ein Vorteil proprietärer Software, bei welcher der Hersteller ein Wissensmonopol über seine Produkte hat, sind ein stabiles Service-Management mit auf die Software spezialisiertem Personal. Sicherheit, Support und Skalierbarkeit werden durch den Software-Hersteller abgewickelt. Diese zentrale Organisation von Entwicklung, Wartung, Vertrieb und Verwaltung ist bei den meisten quelloffenen Software-Projekten nicht gegeben. Bei diesen Gemeinschaftsprojekten leisten verschiedene voneinander unabhängige Parteien und Einzelpersonen Beiträge in den genannten Aufgabenbereichen. Nachteile proprietärer Software sind die Lizenzkosten, der nicht offene Quellcode und ein längerer Weg zwischen etablierten Standards und implementierter Software (HOLTAN 2023: 33 f.). Dies wird in Abschnitt 2.3 näher erläutert.

Die OSGeo (vgl. Abschnitt 2.3) definiert Open Source als Software mit einer Lizenz für die kostenfreie (Wieder-) Nutzung. Außerdem müsse der Zugang zum Quellcode für die Prüfung, Veränderungen und die Möglichkeit, die Software ohne zusätzliche Kosten weiter zu verteilen, gegeben sein (OSGEO 2023: o.S.). Durch die Zugänglichkeit des Quellcodes ergeben sich für die Nutzer offener Software Vorteile wie die Möglichkeiten für schnelle Problembehebungen und Verbesserungen und eine auf die individuelle Nutzung optimierte Funktionalität. Ein Nachteil der Nutzung und Entwicklung von Open Source Anwendungen ist die Übernahme von Verantwortung für eigene Entwicklungen und Lösungen sowie für die Weitergabe des Wissens über diese. Die Leitung einer Open Source Gemeinschaft kann einen erheblichen Arbeitsaufwand verursachen (IGN 2023: 11).

2.2 ArcGIS Pro und proprietäre GIS

ArcGIS Pro ist das aktuelle Desktop-GIS des Unternehmens Esri (Environmental Systems Research Institute), welches aus einer 1969 gegründeten Beratungsfirma für Landnutzung hervorging (HOWELL 2019: o.S.). Es entwickelte sich direkt aus den Arbeiten des Landschaftsarchitekten Jack Dangermond am Labor für Computergrafik und Raumanalyse an der Universität Harvard (WIKIPEDIA (b, c) 2023: o.S.). Die erste GIS-Software des Unternehmens war das 1982 veröffentlichte Kommandozeilenprogramm Arc/INFO. Der Name steht für die Aufteilung in Werkzeuge für die Eingabe, Verarbeitung und Ausgabe geografischer Daten (Architecture) und eine komplementäre, aber separate Datenbank. Diese Art der Architektur prägt auch noch das aktuelle ArcGIS, und auch QGIS, welches an ArcGIS angelehnt ist und diesem in seiner Grundstruktur sowie vielen Begrifflichkeiten und Funktionalitäten ähnelt (vgl. Abschnitt 5.4). Parallel zu Arc/INFO bot Esri verschiedene GUI-basierte Produkte wie ArcView und MapObjects, ArcSDE (RDMS) und eine Programmierbibliothek (vgl. Abschnitt 2.4.1) an. Diese wurden mit der Veröffentlichung von ArcGIS 8.0 (1999) zusammengefasst (WIKIPEDIA (a, b) 2023: o.S.).
2015 hat Esri einen Anteil von etwa 40% am weltweiten GIS-Markt (ESRI (a) 2015: o.S.). Neben ArcGIS Pro (Desktop) gehören ArcGIS Server (verteiltes GIS) und ArcGIS Online (SaaS und PaaS) zur Produktfamilie der ArcGIS Enterprise-Suite.
Andere weit verbreitete proprietäre GIS sind AutoCAD Map 3D (Autodesk Inc., USA), MapInfo Pro (Pitney Bowes Inc., USA), das PaaS Google Earth Engine (Alphabet Inc., USA), Geomedia (Hexagon AB, Schweden) und SuperMap GIS (SuperMap Software Co., China) (TECHNAVIO 2023: o.S.).

2.3 OSGeo und QGIS

Die meisten Open Source Produkte im GIS-Bereich, wie z.B. QGIS, GRASS GIS (Geographic Resources Analysis Support System) und MapGuide, sind Projekte der Open Source Geospatial Foundation (OSGeo) (vgl. Abb. 1).

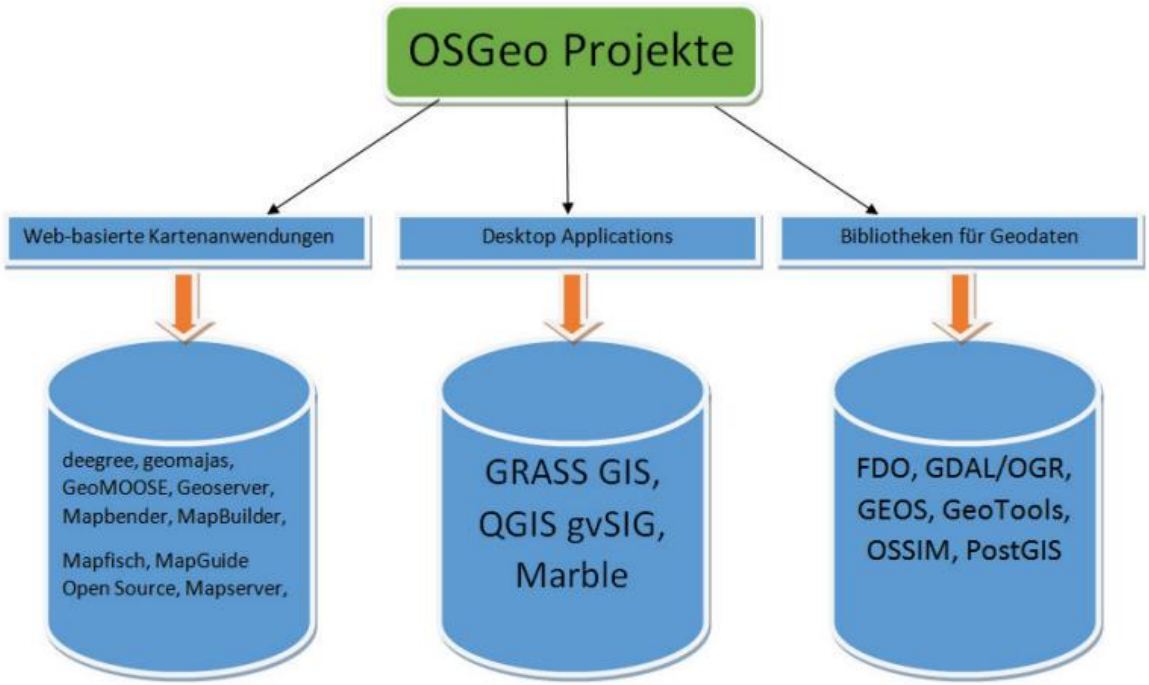


Abbildung 1 Projekte der OSGeo (DIEDERICH 2016: 4)

Die 2006 gegründete OSGeo ist „*eine gemeinnützige Organisation zur Entwicklung und Nutzung von freien und quelloffenen GIS*“ (LANGE 2013: 2). Diese hat sich selbst u.a. Ziele wie die Förderung der Nutzung von Open Source Geo-Software, die Bildung von OSGeo-Gemeinschaften und die Kommunikation und Kooperation zwischen ihnen gesetzt. (OSGEO 2023: o.S.). Außerdem sollen Ressourcen für Projektgründungen zur Verfügung gestellt werden und ein hoher Qualitäts- und Innovationsstandard und die Interoperabilität zwischen den Software-Projekten sowie mit offenen Standards wie denen des Open Geospatial Consortiums (OGC) sichergestellt werden.

Das OGC, selbst Mitglied des World Wide Web Consortium (W3C), nimmt eine zentrale Rolle im Bereich der Geoinformation ein. Es wurde 1994 mit dem Ziel der Entwicklung von Standards „*zur Beseitigung mangelnder Interoperabilität zwischen verschiedenen GIS und zu deren Öffnung gegenüber Standardanwendungen*“ gegründet (BRINKHOFF 2022: 17). Seine Mitglieder sind öffentliche und private Institutionen, GIS-Anbieter, -Nutzer und Forschungseinrichtungen wie z. B. die European Space Agency ESA, die US National Aeronautics and Space Administration NASA, die Unternehmen Google (Alphabet Inc.),

Microsoft, Esri, Hexagon, Trimble, Airbus, die Feng Chia University, Wuhan University und die Fraunhofer-Gesellschaft) (OGC (a) 2023: o.S.).

Das ursprünglich 2002 vom Geologen Gary Sherman entwickelte Quantum GIS (heute QGIS), wurde 2007 als Projekt von der OSGeo aufgenommen und erstmals 2009 veröffentlicht. Das Ziel war zunächst lediglich die Entwicklung eines Viewers für PostGIS-Daten auf Linux-Betriebssystemen. PostGIS (vgl. Abb. 1) ist eine externe Erweiterung der objektrelationalen Datenbank PostgreSQL für die Speicherung und Abfrage von Geodaten auf Grundlage etablierter Standards (BRINKHOFF 2022: 127). Heute umfasst QGIS ein volles Desktop- und Server-GIS, und ist für alle gängigen Unix- und Linux-Distributionen, Windows und MacOS verfügbar (SHERMAN 2009: o.S.). Es wird von freiwilligen Entwicklern gewartet und umfasst eine Vielzahl von in Python und C++ geschriebenen Plugins mit erweiternden Funktionalitäten. Es bestehen Schnittstellen zu PostGIS, MapServer und GeoServer (vgl. Abschnitt 7.1). Die Funktionalitäten der GDAL und GRASS GIS sind in QGIS integriert (CAVALLINI u. LAMI 2010: o.S.). Ein Vorteil des Programms im Vergleich zu anderen GIS ist der geringe Bedarf an Festplatten- und Arbeitsspeicher (WIKIPEDIA (d) 2023: o.S.).

GDAL ist eine für die Konvertierung von Rasterdatenformaten mit Raumbezug entwickelte Programmier-Bibliothek der OSGeo. Sie wurde ursprünglich von Frank Warmerdam entwickelt und erstmals im Jahr 2000 veröffentlicht. Zu ihren Sponsoren gehören u.a. Microsoft, Esri, Google und Amazon Web Services. Die OGR-Bibliothek (*OpenGIS Simple Features Reference Implementation*) für Vektorgeometrien wurde später in die GDAL integriert. Ihre Funktionen können in QGIS sowohl über die Benutzeroberfläche als auch mit der Programmierschnittstelle PyQGIS genutzt werden.

Die Förderung der Interoperabilität zu offenen Standards durch die OSGeo zeigt sich bei QGIS z. B. durch die Nutzung und Bereitstellung offener Dateiformate wie *Virtual Raster* (vgl. Abschnitt 3.4.4) und das Vektordatenformat *GeoPackage* oder dadurch, dass die Veröffentlichung von Webdiensten ausschließlich mit OGC-Standards wie WMS und WPS realisierbar ist (vgl. Abschnitt 7.1). In ArcGIS existieren zwar Schnittstellen zu diesen, allerdings parallel dazu oder ausschließlich auch eigene, nicht offene Lösungen wie z. B. im Falle der Geodienste der *Map*- und *Geoprocessing Service* oder die sogenannten *File Geodata Bases* (fGDB) für die Speicherung von Geodaten. Bei QGIS existiert kein solcher Dateicontainer. Dessen Bedienoberfläche ermöglicht lediglich den Zugriff auf Dateien des Betriebssystems durch einen eigenen Dateibrowser, welcher sich von dem des Betriebssystems nur dadurch unterscheidet, dass Sidecar-Dateien (z. B. *World Files*, vgl. Abschnitt 3.4.1) von Raster- und Vektordateien nicht sichtbar sind. Die *GDB* bietet zwar Funktionalitäten wie z. B. die Dateiabfrage und -änderung (ESRI (d) 2023: o.S.), hat aber den Nachteil, dass die in ihr enthaltenen Dateien vom Betriebssystem aus nicht zugänglich sind.

Dadurch müssen in vielen Fällen zusätzliche Arbeitsschritte wie der Im- und Export von Dateien in und aus der *GDB* durchgeführt werden.

Ansonsten ähnelt der Aufbau von QGIS der bereits durch Arc/INFO geprägten Aufteilung in Dateisystem (*Feature Classes* und *GDB* in ArcGIS) und Karten-Layer, bei dem die Dateien durch das Einladen als Karten-Layer mit ihren einzelnen Features für die Geoverarbeitungsfunktionen zugänglich gemacht werden und mit dem Projekt der Karte verknüpft werden. Umgekehrt kann der Inhalt eines Karten-Layers auch in das Betriebsdateisystem exportiert werden. Sowohl bei Arc- als auch bei QGIS können jedoch bei vielen Funktionen auch die Dateien im Betriebsdateisystem oder der *GDB* verarbeitet werden. Eine der Grundfunktionen, die nur mit Karten-Layern genutzt werden kann, ist die Selektion einzelner Features einer *Feature Class*, wie z. B. einer Vektordatei (z.B. Shape- oder GeoPackage-Dateien). Am BKG werden die Esri-Produkte wie ArcGIS Enterprise in großem Umfang genutzt, aber auch QGIS und andere Open-Source-Software wie z.B. GeoServer oder PostgreSQL sind im Einsatz (vgl. Abschnitt 3.3).

2.4 Python

Python ist eine interpretierte und objektorientierte, höhere Programmiersprache (PYTHON (a) 2023: o.S.). Ihre Syntax zeichnet sich durch eine gute Lesbarkeit und eine flache Lernkurve aus (KAMINSKI 2016: Vorwort) und wird häufig als Skriptsprache verwendet. Sie unterstützt Module und Pakete, was die Modularität und Wiederverwendung von Programmcode fördert. Alle Veröffentlichungen von Python sind quelloffen. Die Sprache wurde Anfang der 1990er Jahre vom Mathematiker Guido van Rossum am Centrum Wiskunde & Informatica in Amsterdam als Nachfolger für die Programmierlehrsprache ABC entwickelt (WIKIPEDIA (e) 2023: o.S.). 1995 entwickelt dieser die Sprache an der Corporation for National Research Initiatives in Virginia weiter, wo sie auch erstmals veröffentlicht wird. 2001 wurde die gemeinnützige Python Software Foundation gegründet (PYTHON 2023: o.S.). Für die Skripte dieser Arbeit wird die Python-Version 3.9 verwendet.

2.4.1 ArcPy

Esri entwickelte für Arc/INFO die AML (*ARC Macro Language*), welche auf einer Skriptsprache für Minicomputersyteme basiert. In ArcGIS 8.1 wird die anwendungsspezifische AML zunächst durch VBA (*Visual Basic for Applications*), und später in ArcGIS 9 durch Python und andere den COM-Standard von Microsoft unterstützende Skriptsprachen wie Perl und VBScript abgelöst. (WIKIPEDIA (a, b) 2023: o.S.).
Abbildung 2 zeigt die Struktur der ArcPy-Bibliothek in vereinfachter Form für eine beispielhafte Auswahl von Modulen, Klassen und Funktionen. Im Workflow angewendete

Elemente sind farbig gekennzeichnet. Die Funktionen sind in viele weitere Untergruppen (*Toolsets*) gegliedert, die hier nicht aufgeführt sind. So wird z. B. die Funktion *clip()* im Skript mit *arcpy.management.Clip()* aufgerufen, da diese der Untergruppe *Data Management* angehört. Sie ist aber ebenso dem Klassenschema entsprechend über das Modul *Image Analysis* mit *arcpy.ia.Clip()* abrufbar. Ein Karten-Layer wird in ArcPy sowohl im Falle von Raster- als auch Vektordaten mit

`arcpy.MakeFeatureLayer_management([Dateipfad], "Layername"),`

oder alternativ mit

`arcpy.management.MakeFeatureLayer([Dateipfad], "Layername")` geladen.

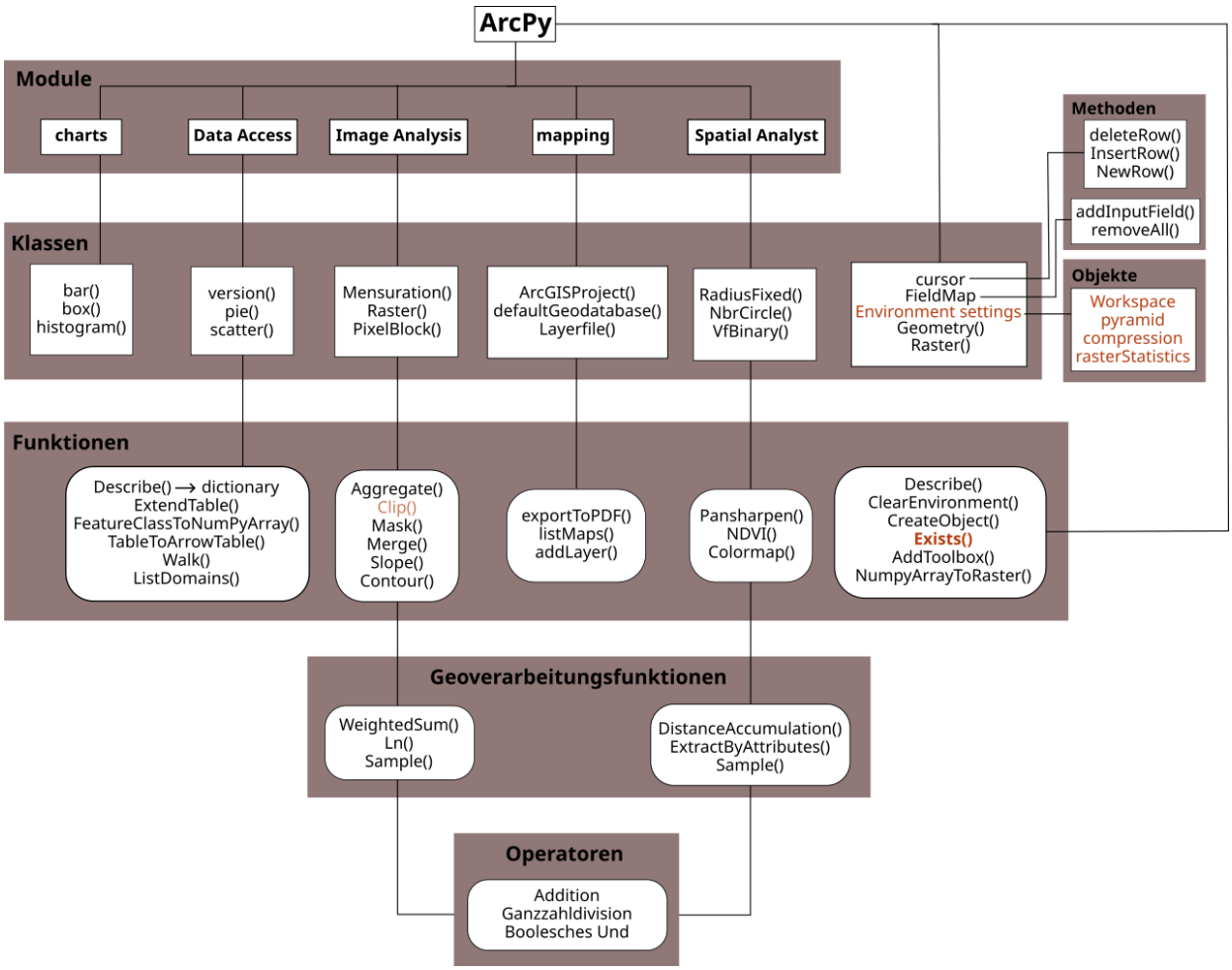


Abbildung 2: Klassendiagramm Arcpy (Quelle: eigene / ESRI (b) 2023: o.S.)

2.4.2 PyQGIS

In QGIS wird Python ab der Version 0.9 als Programmierschnittstelle unterstützt (QGIS (b) 2023: o.S.). Ab 2007 beginnt mit Version 0.11 die Entwicklung einer separaten Bibliothek (GHISLA, WARMERDAM 2008: o.S.), welche erstmals mit QGIS 1.0 als stabile Version veröffentlicht wird (SHERMAN 2009: o.S.). Abbildung 3 zeigt eine Auswahl einiger ihrer

Klassen und ihre Beziehungen untereinander. Sie sind den Modulen *core, gui, analysis, server, processing* und *3D* untergeordnet (QGIS (c) 2023: o.S.). Qt ist ein von QGIS genutztes GUI-Toolkit.

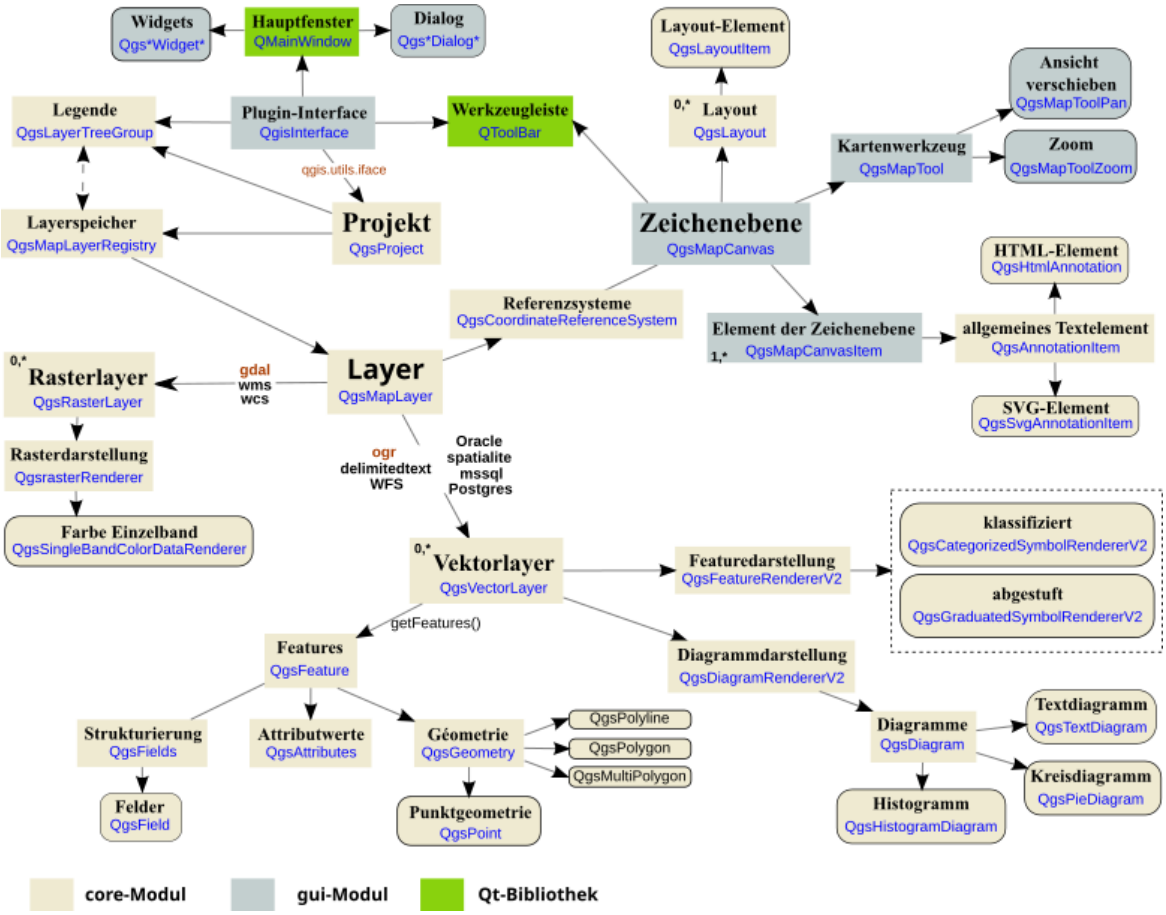


Abbildung 3: Klassendiagramm PyQGIS (eigene Darstellung nach GRATIER 2015: o.S.)

Ein Vektorlayer wird mit `variable = QgsVectorLayer([Pfad], "LayerName", "ogr")` als Variablenwert geladen und mit

`QgsProject.instance().addMapLayer(variable)`

der Karte des Projekts hinzugefügt. Alternativ können diese beiden Schritte auch mit der Methode *addVectorLayer()* der Klasse *QgisInterface* aus dem gui-Modul zusammengefasst werden. Dabei ist *iface()* aus dem Modul *utils* eine Instanz der Klasse *QgisInterface* :

`iface.addVectorLayer([Pfad], "LayerName", "ogr")`

Für Rasterlayer wird analog `variable = QgsRasterLayer([Pfad], "LayerName")` und `iface.addRasterLayer([Pfad], "layerName")`

angewendet. Eine Funktion kann durch *processing.run()* mit dem Namen der Anbieterbibliothek, dem Funktionsnamen und einem Dictionary mit den Parametern aufgerufen werden: `processing.run('gdal:translate', {'INPUT': [Pfad],'OUTPUT': r"[Pfad]/Name.vrt", 'EXTRA': '-b 1', 'NODATA': 0})`

3 Datenintegration am Beispiel der DTK

3.1 Problemstellung

Eine Rasterdatensammlung besteht aus einer Vielzahl von Einzelrastern, die ein bestimmtes Größenformat aufweisen. Der einfachste denkbare Aktualisierungsprozess von Rasterdatensammlungen wäre der Austausch bestimmter Kacheln, für den Fall, dass die neuen Daten im gleichen Format wie die Bestandsraster vorliegen. Dabei wäre keine Prozessierung und folglich auch keine Methode für die Datenreferenzierung notwendig. Bei der gestellten Aufgabe ist dies aber nicht der Fall, weil sich die variablen Kachelgrößen der übergebenen Daten der Vermessungsverwaltungen der Bundesländer von den sogenannten Vertriebseinheiten des BKG unterscheiden. (Erstere werden als Erfassungseinheiten bezeichnet, auch wenn es sich bei der DTK vor allem bei den kleineren Maßstäben eigentlich um ein Ableitungsprodukt, und nicht um originär erfasste Geodaten handelt.) Der zweite Grund für die Notwendigkeit der Datenprozessierung beim Austausch ist ihre geografische Aufteilung nach den Bundeslandgrenzen. Abbildung 4 zeigt die grundlegende Problemstellung am Beispiel von fünf Kartenblättern im Maßstab 1:100 des Bundeslandes Rheinland-Pfalz: eine variable Anzahl zusammenhängender oder einzelner Eingangsraster, deren Inhalt im Bereich des betreffenden Bundeslandes in den Bestandsdatensatz der DTK eingefügt werden soll.

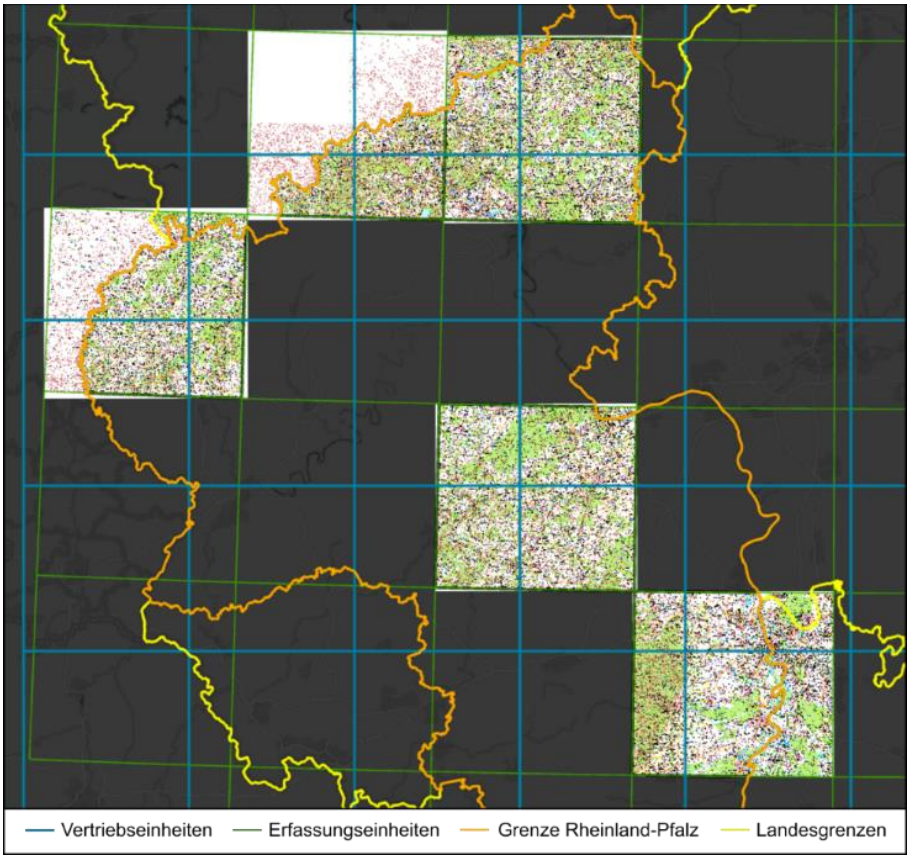


Abbildung 4: prinzipieller Ablauf bei der Datenintegration der DTK (Quelle: eigene)

Der gesetzliche Auftrag für die Erfassung und nach einheitlichen Grundsätzen erfolgende Darstellung der Topografie des Landesgebietes geht in Deutschland grundsätzlich an die Vermessungsverwaltungen der Länder (BKG 2023: o.S.). Trotzdem ist auch das BKG als Institution des Bundes in den Prozess der kartografischen Informationsgewinnung für die Produktion und Aktualisierung der DTK eingebunden (vgl. Abschnitt 3.2). In Bezug auf die Bereitstellung und Verwaltung der Daten beschränkt sich die Aufgabe für das BKG aber auf die Integration der Daten. Die Vermessungsämter können weitgehend autonom entscheiden, wie sie den Auftrag der Kartenproduktion und -pflege erfüllen. Daraus resultiert eine gewisse Heterogenität in Bezug auf Häufigkeit, Datenmenge und -qualität der Übergaben an das BKG. Einige Länder liefern die Daten im Standardblattschnitt (Kartenblätter), andere in Form von am UTM-Gitter ausgerichteten Kacheln verschiedener Größen (vgl. Abb. 5). Die Kartenblattkacheln der drei Maßstäbe haben Seitenlängen von etwa 11 km, 22 km und 45 km, die Vertriebseinheiten 10 km, 20 km und 40 km.

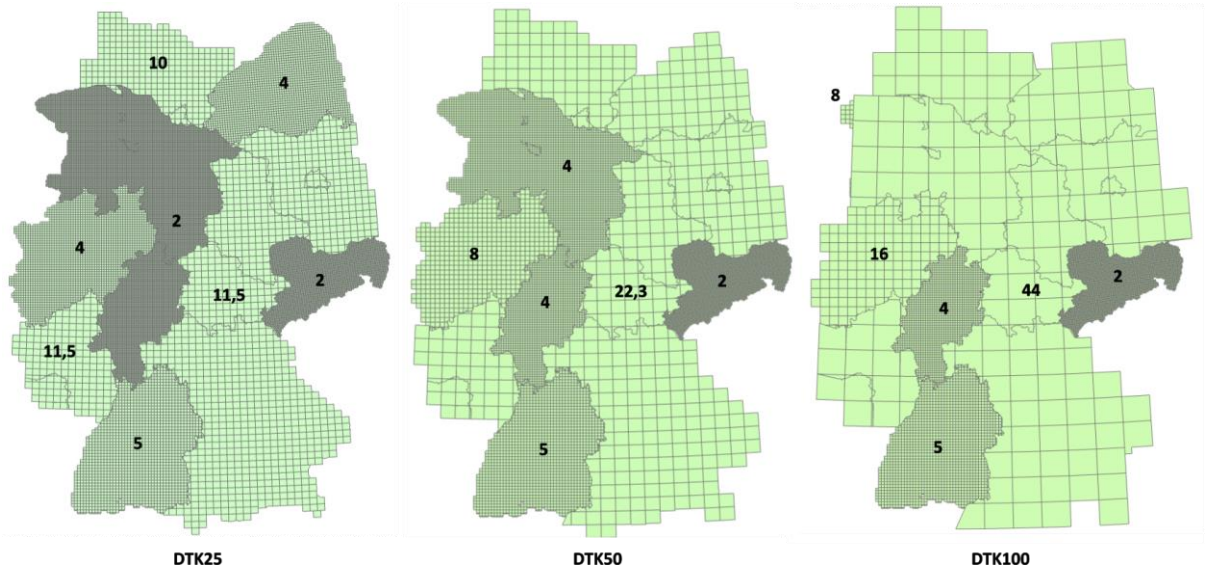


Abbildung 5: Seitenlängen der Erfassungseinheiten in Kilometer (Quelle: eigene / BKG)

Teilweise sind die übergebenen Daten bereits auf die jeweilige Bundeslandfläche reduziert. Der Pixelwert für die datenlosen Bereiche der Kartenblatträndern und Fläche außerhalb des Landes alterniert beim Farblayer zwischen null und eins. Außerdem unterscheiden sich die Daten in der Zone des Projektionssystems UTM (33/32).

Grundlage für die Ableitung der DTK ist u.a. das ATKIS Basis-DLM und das Digitale Geländemodell (DGM). Dessen Aktualisierung erfolgt maßgeblich auf der Grundlage der Informationen des Liegenschaftswesens und der Auswertung der Bilddaten des jeweils aktuellen Messbildfluges (SENATSVERWALTUNG BERLIN 2023: o.S.).

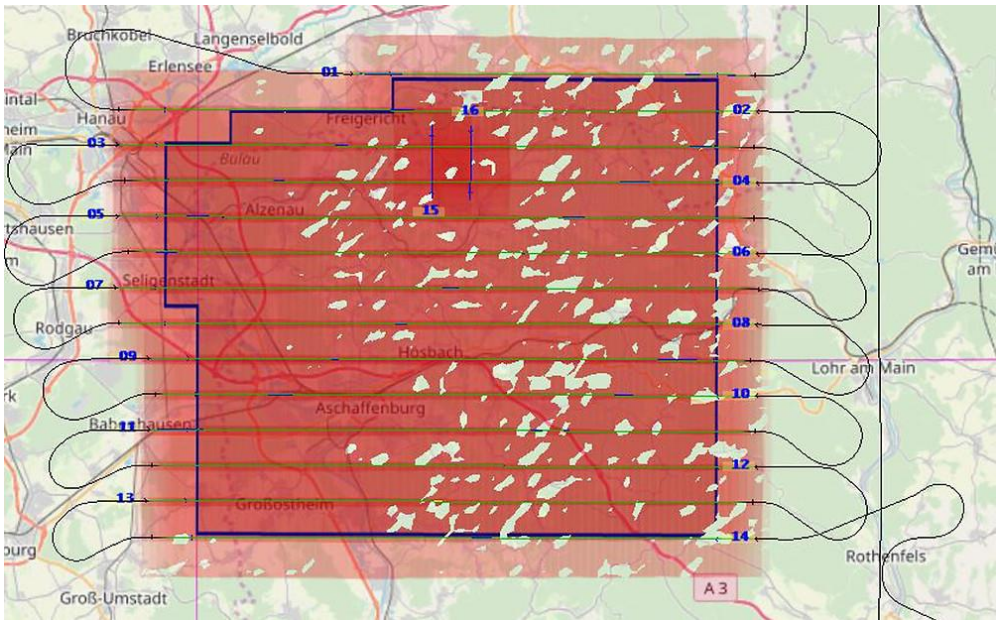


Abbildung 6: prinzipieller Ablauf bei der Datenintegration der DTK (GEOPLANA 2023: o.S.)

Wie in Abbildung 6 deutlich wird, kann der bei den Messbildflügen erfasste Bereich durch die geraden Flugstreifen nicht genau mit den politischen Ländergrenzen abschließen, weshalb die von den Vermessungsverwaltungen erfassten Daten über diese hinausgehen. Die Zuordnung bei der Datenintegration erfolgt aber nach diesen Ländergrenzen. Es ist anzumerken, dass die Gebiete in anderen Ländern, wie z. B. Frankreich und der Schweiz, nach Flugblöcken statt politischen Grenzen aufgeteilt werden (vgl. Abb. 7). Neben der einfacheren Datenintegration werden so vor allem Mehrfacherfassungen in den Grenzbereichen mit nicht zu vernachlässigenden ökologischen und ökonomischen Kosten vermieden.

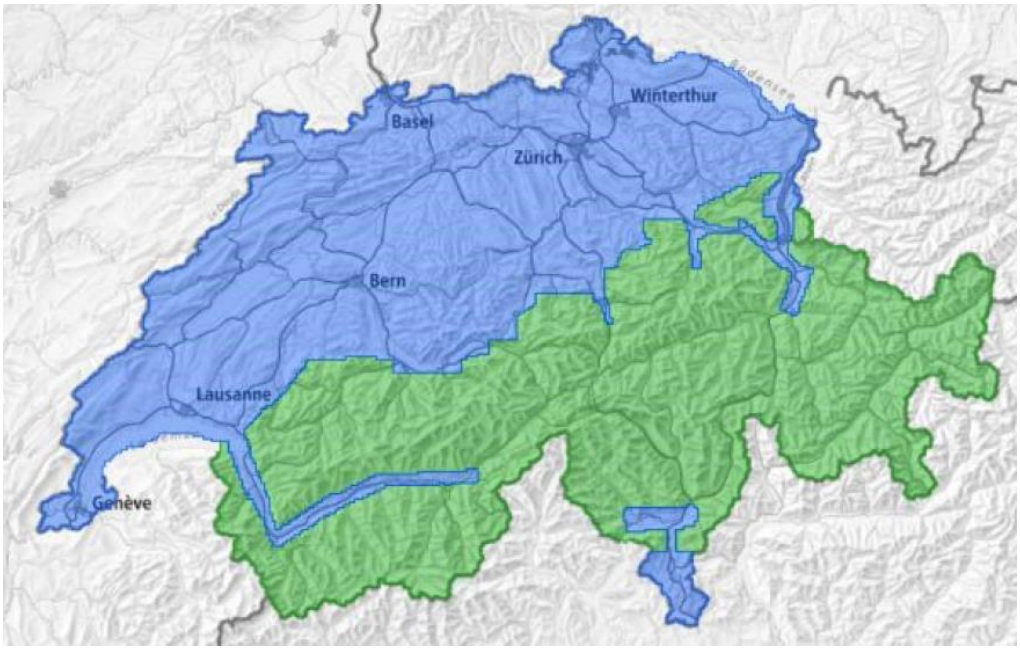


Abbildung 7: Aufteilung der Schweiz nach Flugblöcken (BOVIER 2023: 6)

Die Begrenzung der gelieferten Daten auf die jeweilige Landesfläche ist einer der zentralen Prozessierungsschritte. Die Bildschirmaufnahmen aus QGIS in Abbildung 8 zeigen den generellen Ablauf des Workflows. Abbildung 9 stellt den Prozess schematisch im Zusammenhang mit der beim BKG vorliegenden Datenablagestruktur dar.

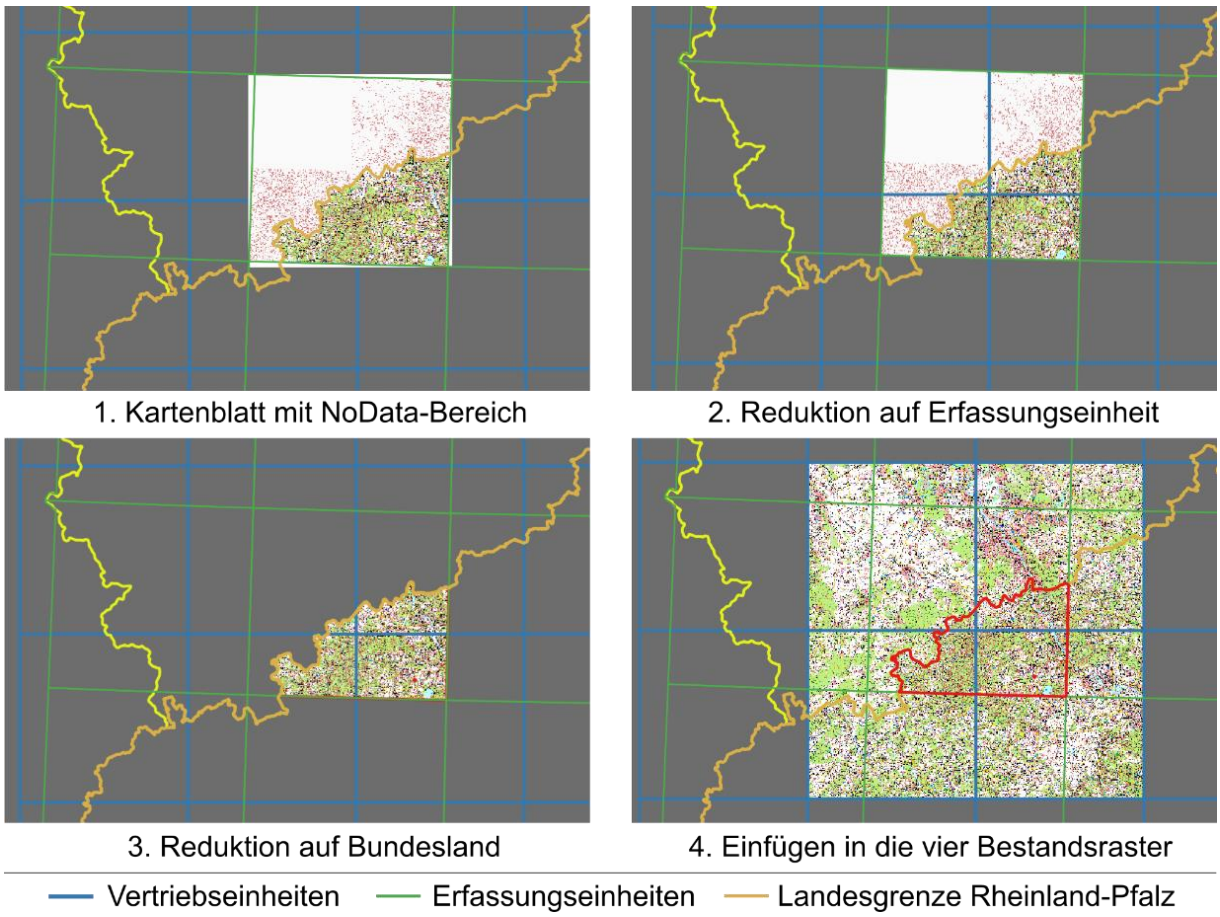


Abbildung 8: Ablauf der DTK-Datenintegration in QGIS (Quelle: eigene)

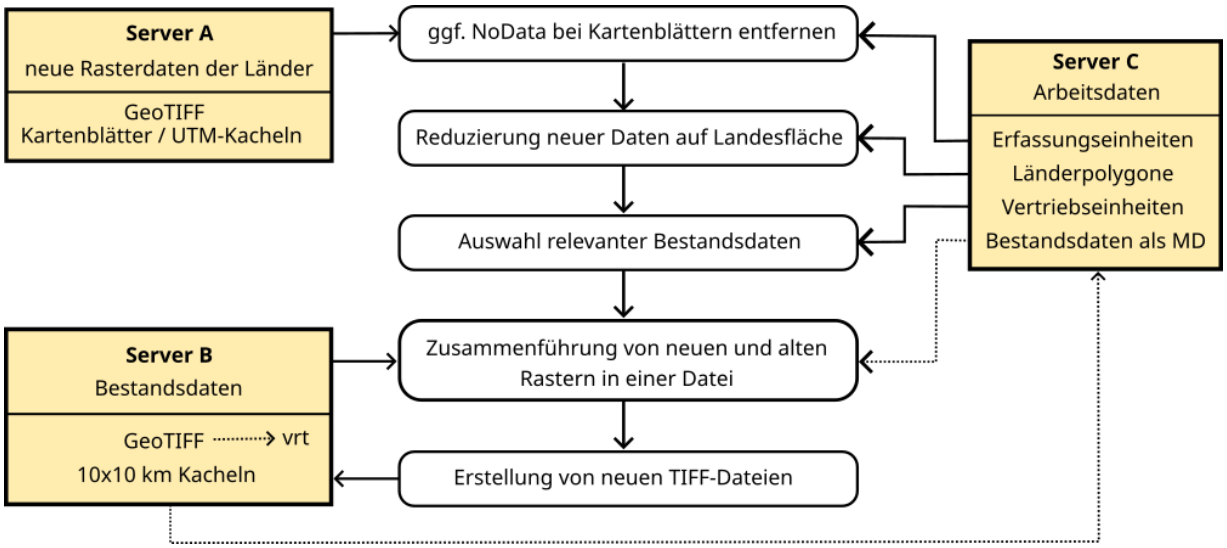


Abbildung 9: schematischer Ablauf der DTK-Datenintegration (Quelle: eigene)

Die Vektordateien der Vertriebseinheiten liegen in einer PostgreSQL-Datenbank, auf die ArcGIS über eine SDE-Datei (*Spatial Database Engine*, von Esri) zugreift. Für QGIS wurden diese als Shape-Dateien exportiert. Die Bestandsraster sind als TIFF-Dateien (vgl. Abschnitt 3.4.1) gespeichert und werden sowohl als VRT auf Server B selbst, als auch als *Mosaic Datasets* auf Server C referenziert.
Die Implementation erfolgt generell unter der Annahme, dass die Art der Eingangsdaten hinsichtlich des Nodata-Wertes und der Reduktion auf die Landesfläche nicht bekannt ist. Da diese Eigenschaften auch nicht auf einfache Weise automatisch ermittelt werden können, werden in beiden Skripten die Kartenblattränder und Flächen außerhalb des Bundeslandes grundsätzlich durch *Mosaic Dataset*-, bzw. clip-Funktionen entfernt, auch bei den Eingangsdaten, bei denen es eigentlich nicht notwendig wäre. Diese Strategie wird vom BKG auch deshalb verfolgt, weil Esri in der Dokumentation angibt, dass sich die Nutzung von Transparenzen in *Mosaic Datasets* negativ auf die Performanz auswirken. Auf eine Alternative mit Voraussetzung der Bekanntheit dieser Eigenschaften wird in Abschnitt 5.2 eingegangen.

3.2 DTK und DOP

Die DTK ist eine „*einheitliche topographische Beschreibung des Gebiets der Bundesrepublik Deutschland für Bundeseinrichtungen und den Vertrieb* [für Privatkunden]“. Das BKG übernimmt die Aufgabe der Integration der DTK-Datensätze der Landesvermessungsverwaltungen und bietet diese als geotopografische Basisinformationen in Form von Rasterdaten an (BKG (a) 2023: o.S.). Die Landeskartenwerke wurden bis 1990 ausschließlich in analoger, gedruckter Form herausgegeben. Seit 1990 werden sie als DTK im Amtlichen Topografisch-Kartografischen Informationssystem (ATKIS) geführt. Dieses beschreibt die Topografie der Bundesrepublik Deutschland in einer geotopografischen Datenbasis und dient der Ableitung nutzungsorientierter digitaler Erdoberflächenmodelle wie z. B. DLM und DTK (BKG (b) 2023: o.S.). Abbildung 10 beschreibt den recht komplexen Prozess der kartografischen Informationsgewinnung für die DTK, DLM und DOP, auf welchen hier nicht im Detail eingegangen werden kann. Das Dienstleistungszentrum (DLZ), Teil des BKG, stellt *„geotopografische und geodätische Referenzdaten des Bundes zentral zur Verfügung, welche von Bundeseinrichtungen, öffentlichen Verwaltungen, Wirtschaftsunternehmen, Wissenschaftseinrichtungen und Bürgern genutzt werden können und als Geo-Webdienste oder Geodaten-Download verfügbar sind. Seine Mitarbeiter analysieren den individuellen Bedarf und stellen den Kunden praxisorientierte Dienste und Produkte bereit. Das DLZ betreibt leistungsstarke Webserver, um Nutzern ein breites Spektrum an hochverfügbaren Webdiensten anzubieten, die komfortabel in eigene*

Anwendungen eingebunden werden können und kontinuierlich auf die aktuellsten Geobasisdaten zugreifen.“

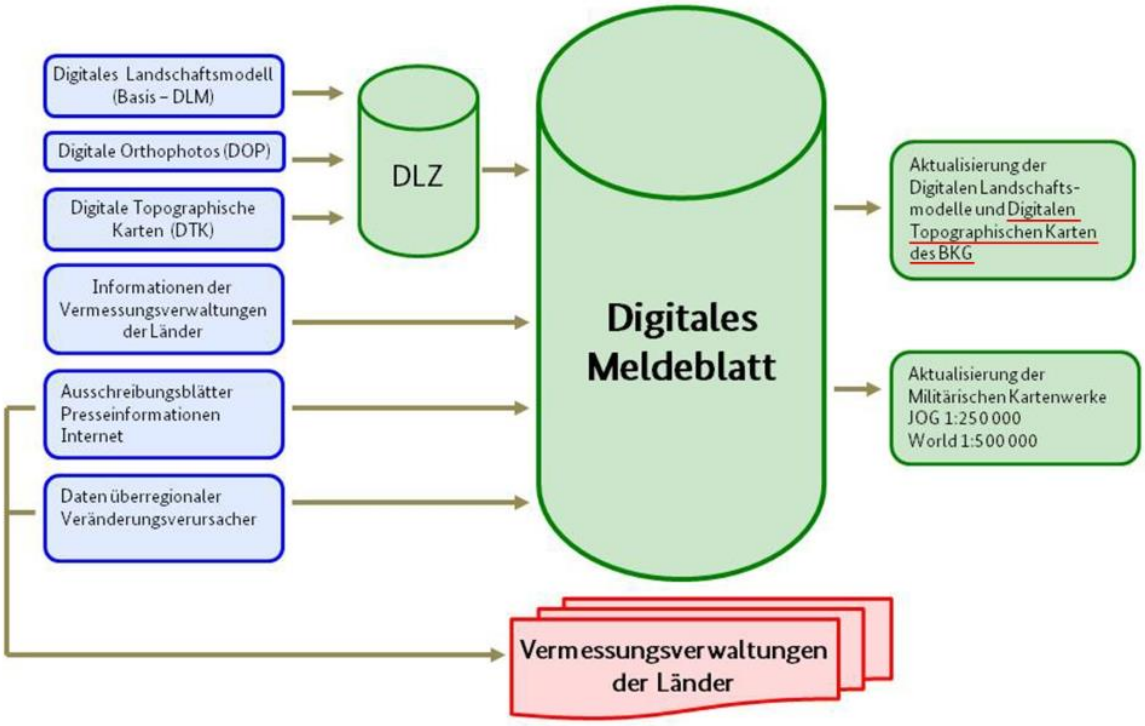


Abbildung 10: kartografische Informationsgewinnung (BKG (c) 2023: o.S.)

Der Prozess umfasst die „*Beschaffung, Bewertung, Aufbereitung und Bereitstellung von sogenannten Fortführungsgrundlagen zur Laufendhaltung der DLM und DTK. Die Informationen über Veränderungen werden nach dem Verursacherprinzip dem Topographischen Informationsmanagement (TIM) zugeführt.*“ Überregionale Verursacher, wie z. B. die DB Netz AG, übergeben diese direkt an das BKG, regionale Verursacher an die Landesvermessungsverwaltungen. Für die Aktualität der verschiedenen topografischen Objekte und Attribute des sogenannten ATKIS-Objektartenkatalogs gibt das BKG einen Zeitraum von drei bis zwölf Monaten an (BKG (c) 2023: o.S.).

Der Grund für die Haltung und Bereitstellung der DTK-Daten im Rasterformat ist die ehemalig ausschließlich analoge, gedruckte Ausgabeform in den Maßstäben 1:25.000, 1:50.000 und 1:100.000. Im oben skizzierten Produktionsprozess existiert die DTK aber zunächst im Vektorformat. Beim hier untersuchten Integrationsprozess erhält das BKG die Daten bereits als Raster. Die DTK50 und DTK100 werden anhand bestimmter kartografischer Generalisierungsregeln von der DTK25 abgeleitet. In Zukunft soll die DTK auch, bzw. primär im Vektorformat angeboten werden. Eine Besonderheit der DTK-Daten im Rasterformat ist ihre Gliederung nach thematischen Inhaltselementen in verschiedenen Ebenen. Jedes dieser Inhaltselemente, wie z. B. „Haus“, „Wiese“, „Park“, „weiß“, „grau“, „UTM-Gitter“, bildet einen der 24 einfarbigen Einzellayer mit 1 Bit Farbtiefe (BKG (d) 2021:

4), (vgl Abb. 11). Die Kombination dieser Einzellayer ergibt das farbige Gesamtbild der Karte. Dazu wird jeder Einzellayer über eine RGB-Farbpalette einer Farbe zugeordnet. Diese Palette ist in den TIFF-Dateien des Farblayers mit 8 Bit Farbtiefe integriert. Für jede Erfassungseinheit übergeben die Länder also 24 1 Bit-Dateien und eine 8 Bit-Datei des Farblayers, welche in einem gemeinsamen Verzeichnis gespeichert werden. Die Dateien sind mit der LZW-Methode komprimiert (vgl. Abschnitt 3.4.1). Eine vom Landesvermessungsamt Thüringen übergebene Kachel der DTK50 des Farblayers hat z. B. eine Größe von 9635 x 9211 Pixel, 5,99 MB und eine Auflösung von 508 dpi (vgl. Abschnitt 5.2, Tab. 1). Die Größe einer zugehörigen TIFF-Datei eines Einzellayers beträgt 126 KB.

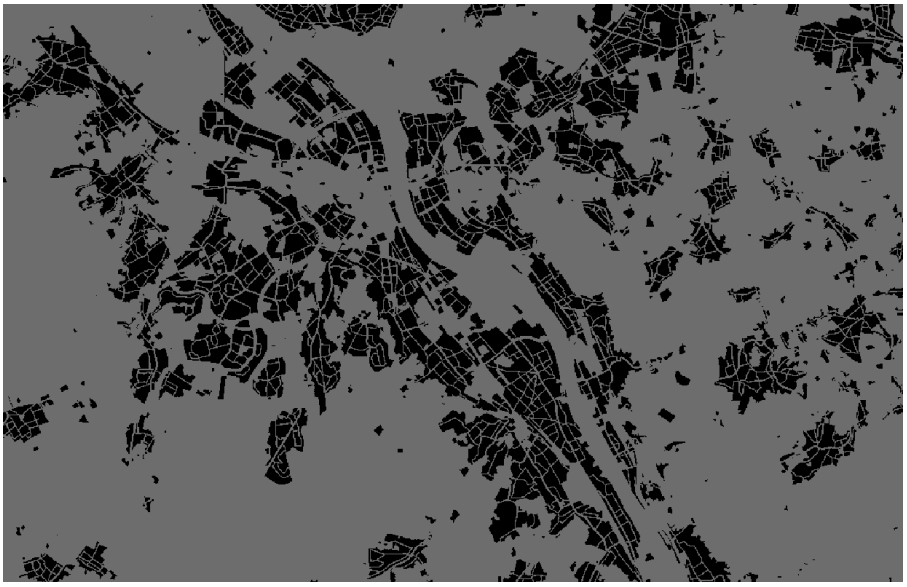


Abbildung 11: DTK-Layer „hrot“ (Haus rot) (Quelle: eigene)

Die zunächst ausschließlich für die DTK erstellten Skripte wurden anschließend so erweitert, dass sie auch für die Datenintegration der Digitalen Orthophotos mit einer Bodenauflösung von 20 cm (DOP20) verwendbar sind. Darüber hinaus ist auch eine Anwendung auf die Rasterdaten des Digitalen Geländemodells (DGM) und des Digitalen Oberflächenmodells (DOM) denkbar, welche als 1-Band Raster mit 32 Bit gespeichert werden. Das BKG selbst schreibt zur Definition von DOP: „*DOP sind verzerrungsfreie und georeferenzierte Luftbilder. Sie sind maßstabstreu und können so direkt mit Karten oder Fachdaten kombiniert werden.*“ Die Vermessungsverwaltungen der Bundesländer erzeugen DOP mit einer Bodenauflösung von 20 cm. Das BKG führt die Rasterdaten länderübergreifend zusammen und generiert deutschlandweite Datensätze mit Bodenauflösungen von 20 cm (DOP20) und 40 cm (DOP40). (BKG (e) 2023: o.S.). Für die Beseitigung der durch die fotografische Zentralprojektion und Höhenunterschiede entstehenden Verzerrungen der Luftbilder werden verschiedene photogrammetrische Verfahren wie die differenzielle Entzerrung mit Hilfe von DGM für normale und DOM für *True Orthophotos* mit Ray Tracing-Verfahren angewendet (LUHMANN 2017: 302 ff.). DGM und DOM wiederum werden ebenfalls aus vom Flugzeug

aufgenommenen Laserscan-Punktwolken gewonnen (ALS, *Aerial Laser Scanning*). Abbildung 12 zeigt die Aktualität der Bildflüge der DOP20 (Vertriebsdaten).

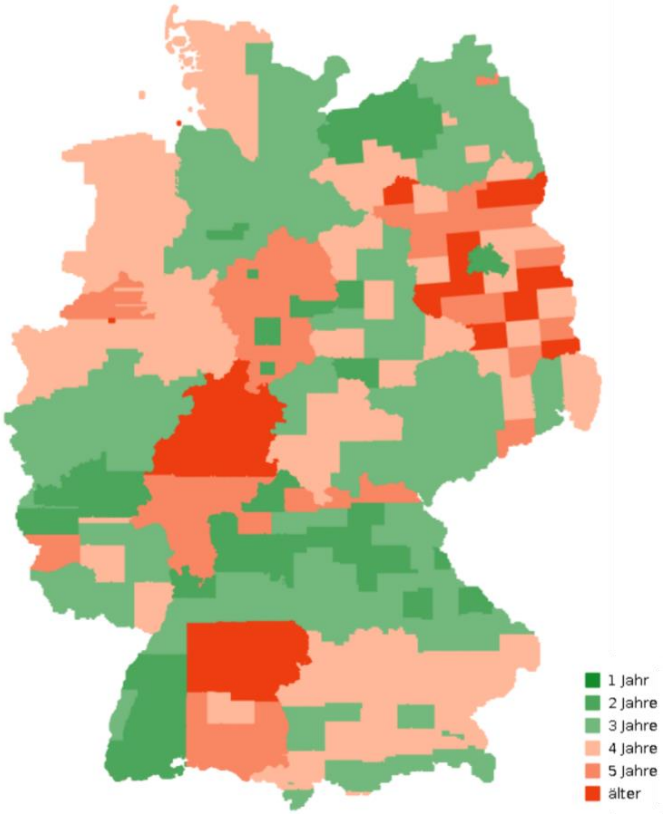


Abbildung 12: Aktualität der Bildflug-Vertriebsdaten der DOP20 (BKG (f) 2023: o.S.)

Eine DOP-Kachel des Eingangsdatenformates ist 10 mal 10 tausend Pixel und 400 MB groß, und hat eine Auflösung von 1 dpi in Höhe und Breite. Die Bodenauflösung von 20 cm bedeutet, dass die Seitenlänge eines Pixels einer Distanz von 20 cm in der Realität entspricht. Anders als die Dateien der DTK sind sie nicht komprimiert. Der bundesweite Datenbestand der DOP20 umfasst aktuell 50,73 TB. Gerade hier sind also, noch mehr als bei den DTK-Daten, durchdachte Strategien und effiziente Algorithmen mit geringen Laufzeiten für die Übertragung, Speicherung und ggf. notwendige Prozessierung der Daten unabdingbar, um den Bedarf an Hardware-Infrastruktur und Energieverbrauch so gering wie möglich zu halten.

3.3 BKG und Open Data

Die Open Data-Bewegung fordert, dass Informationen generell für jeden frei zugänglich sein sollen. Da bei der Bereitstellung digitaler Informationen mit dem Internet nur noch vergleichsweise geringe physische Kosten entstehen, sollten keine künstlichen Barrieren durch Gesetze und Lizenzen für den Zugang zu diesen geschaffen werden. Auf diesem Grundsatz wird vor allem der offene Zugang zu wissenschaftlichen Publikationen,

technischen Normierungswerken und Daten staatlicher Organisationen (Open Government) gefordert. Dadurch sollen soziale Gerechtigkeit, Transparenz und Demokratie gefördert werden. Da auch Computerprogramme durch ihren Quellcode ein rein informatives Gut darstellen, folgt die Open Source-Philosophie dieser Forderung der Open Data-Bewegung unmittelbar. Zwei prominente Vertreter dieser Bewegungen sind z. B. Tim Berners-Lees, bekannt als „Erfinder des World Wide Web“ und Mitverantwortlicher des Open Data-Projektes *data.gov.uk* der britischen Regierung (W3C 2009: o.S.), sowie Aaron Swartz, Mitbegründer der Organisation Creative Commons und Co-Entwickler des Web-Framework *web.py* sowie des Markup-Formats *Markdown*. Durch seine Bemühungen für die Dezentralisierung digitaler Information, z. B. durch den Download und die Verbreitung einer großen Anzahl kostenpflichtiger Artikel wissenschaftlicher Zeitschriften vom MIT-Netzwerk machte letzterer 2013 auf die *“Monopolisierung wissenschaftlicher Veröffentlichungen durch Lizensierungen* aufmerksam” (LEVI u. PASCUAL 2023: o.S.).

Geobasisdaten wie die DTK und DOP stellen eine wichtige Grundlage für das wirtschaftliche und politische Handeln verschiedenster privater und institutioneller Akteure dar. Der freie Zugang zu diesen lässt damit ähnlich positive Effekte für die Gesellschaft wie die durch Open Data im Allgemeinen verursachten erwarten. Durch die 2021 beschlossene Open Data-Strategie der Bundesregierung “*soll das Nutzungspotenzial von Open Data zum Wohle der Wirtschaft, Wissenschaft, Zivilgesellschaft und des Staates verbessert werden.*” (BUNDESMINISTERIUM DES INNERN UND FÜR HEIMAT 2021: o.S.). Das BKG “*unterstützt die Verpflichtung der Bundesregierung für offene Datenpolitik*” (BKG (g) 2023: o.S.). Das zeigt sich durch die Zielsetzung der Arbeitsgemeinschaft Smartmapping (BKG und AdV-Mitglieder) einer agilen Verfahrensentwicklung und der Nutzung von Open Source Software bei der Entwicklung neuer AdV-Standard-Produkte wie *basiskarte.de (BKG (h) 2017: o.S.)*, der allgemeinen Verwendung von QGIS und der Nutzung von GeoServer z. B. für die Bereitstellung der integrierten DTK- und DOP-Daten im Vertrieb. Weitere Beispiele für bereits am BKG mit quelloffener Software realisierte Projekte sind

- das mit dem Geoportal-Baukasten *Masterportal* umgesetzte *geoportal.de (BKG (i) 2022: 4),* die Web-Applikation *RoutingPlus (*ELLING u. WEINER 2022*: 3),*
- der Landüberwachungsdienst Copernicus mit der Nutzung von QGIS und des ESA-Programms SNAP (BKG (j) 2016: o.S.) und
- die Echtzeit-GNSS-Anwendung *BKG Ntrip Client.*

Mit letzterer will sich das BKG als Mitglied der Radio Technical Commission for Maritime Services (RTCM), aktiv an der Definition offener internationaler Industriestandards im Umfeld von GNSS beteiligen (BKG (g) 2023: o.S.). Außerdem bietet das BKG über das DLZ mehr als 50 Produkte und Dienstleistungen als Open Data an, darunter zum Beispiel die DTK250,

DTK500 und DTK1000 (BKG (k) 2023: 2 ff.). Bei den Landesvermessungsverwaltungen kann eine ähnliche Tendenz beobachtet werden. So nutzt z. B. das Landesamt für Digitalisierung, Breitband und Vermessung (LDBV) in Bayern seit 2007 die Linux-Distribution OpenSUSE (DIEDERICH 2016: 8).

3.4 Grundlagen zur Durchführung des Prozesses

Die Bestands- und Aktualisierungsdaten werden durch die in der Einführung erwähnten Raster-Container bearbeitet und zusammengefügt, um die eigentlichen Dateien dabei nicht zu kopieren. Die wichtigste im vorliegenden Prozess angewandte Funktion ist die Veränderung des Ausmaßes der Rastersammlungen und der Zuweisung eines NoData-Bereichs, in dem das Raster keine Informationen enthält. Die in ArcGIS genutzten *Mosaic Datasets* werden in der Folge mit MD bezeichnet. Das im QGIS-Workflow genutzte *Virtual Format* der GDAL wird in der Dokumentation auch als *Virtual Raster Dataset* (VRT) bezeichnet.

3.4.1 Tagged Image File Format (TIFF)

Sowohl die DTK- als auch die DOP-Daten werden dem BKG als TIFF-Dateien (auch als TIF bezeichnet) übergeben und nach der Integration auch in diesem Format gespeichert. TIFF ist ein erweiterbares und verlustfreies Rasterformat. Es wurde 1985 zur Vereinheitlichung der Formate für gescannte Bilder von Scanner-Geräten entwickelt und später von Adobe Inc. übernommen. Die für die DTK-Daten genutzte verlustfreie LZW-Komprimierung wird ab Revision 5.0 des Formates unterstützt (ADOBE 2023: o.S.). Das Standardformat für den Grauwertebereich sind positive (vorzeichenlose) Ganzzahlen (unsigned integer). Durch Erweiterungen können auch vorzeichenbehaftete Ganzzahlen, Fließkommazahlen oder ein anderweitig definierter Wertebereich festgelegt werden (WIKIPEDIA (f) 2023: o.S.).
Anfang der 90er Jahre wurde die Erweiterung GeoTIFF bei der NASA entwickelt, um geographische Bilddaten durch das Hinzufügen von Metadaten zu beschreiben. 2019 wurde es in erweiterter Form durch die OGC veröffentlicht (OGC (b) 2023: o.S.). Die Metadaten für die Georeferenzierung sind dabei als Tags in der eigentlichen Datei enthalten. Die DTK-Daten werden von den Landesvermessungsverwaltungen aber ohne diese Erweiterung übergeben. Die Georeferenzierung erfolgt stattdessen über Esri *World Files*, welche sich im selben Verzeichnis wie die TIFF-Dateien befinden. Diese enthalten sechs Zeilen mit den Parametern der Affintransformation mit sich von der im VRT-Format vorhandenen unterscheidenden Reihenfolge (vgl. Abschnitt 3.4.4). Außerdem referenzieren die Parameter der Koordinaten den Mittelpunkt statt, wie bei GDAL, die obere linke Ecke des ersten Pixels der ersten Spalte (ULT-Pixel), (BRINKHOFF 2022: 424).

3.4.2 Mosaic Dataset

Bei den *Mosaic Datasets* von Esri wird zwischen Quell-MD, in welchen die Speicherorte originärer Rasterdateien hinterlegt sind und abgeleiteten MD für solche, die wiederum andere MD referenzieren, unterschieden. Ein Großteil der in ArcGIS Pro zur Verfügung stehenden Geoanalyse- und alle Rasterverarbeitungsfunktionen können auf ein MD angewendet werden. Im DTK-Workflow wird vor allem die Möglichkeit, das Ausmaß der referenzierten Raster durch den MD-Sublayer *Footprint* zu definieren, genutzt. MD können über die Bedienoberfläche von ArcGIS Pro, den Modelbuilder für die grafisch-interaktive Erstellung von Befehlsketten, ArcPy oder *Mosaic Dataset Configuration Scripts* (MDCS) erstellt werden. Im vorhandenen Workflow wird die zuletzt genannte Methode in Kombination mit Funktionen der ArcPy-Bibliothek genutzt.

3.4.3 Mosaic Dataset Configuration Script (MDCS)

MDCS ist ein von Esri zur Verfügung gestelltes Paket quelloffener Python-Skripte und XML-Konfigurationsdateien, mit welchem Workflows in einem einzigen Schritt parametrisiert und ausgeführt werden können. XML (Extensible Markup Language) wurde vom W3C zur Repräsentation von Daten entwickelt und strukturiert ein Dokument mit Hilfe von Tags (BRINKHOFF 2022: 28) Das vereinfacht die Automatisierung der Erstellung neuer MD. Die Parametrisierung des Workflows hilft außerdem bei der Implementierung von „Best Practices“ bei der Rasterdatenverwaltung (ESRI (c) 2022: 3). Abbildung 13 zeigt die Ordnerstruktur des MDCS-Pakets mit den im Workflow angewandten Elementen und den Abhängigkeiten der Skripte untereinander.

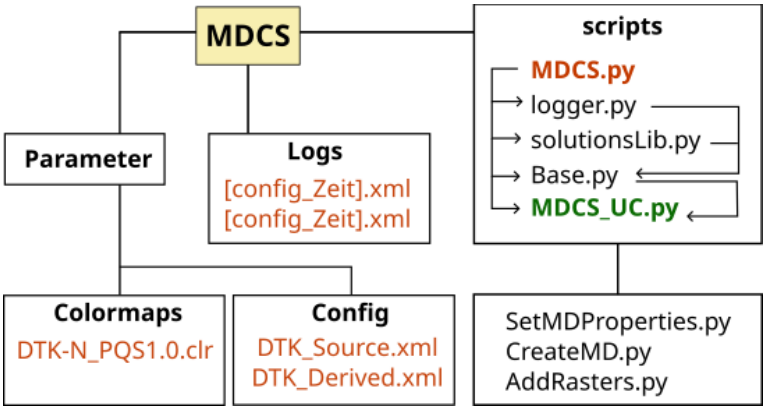


Abbildung 13: Dateistruktur des MDCS-Pakets (Quelle: eigene)

Die Verwendung von MDCS verkürzt das Skript deutlich, weil alle auf MD bezogene Prozesse nicht als ArcPy-Befehle aufgerufen werden müssen. So ist ein Großteil des hier untersuchten Workflows in zwei MDCS-Prozessen gekapselt. Als weiteren Vorteil gibt Esri das standardisierte Logging mit der Laufzeit und den Details der einzelnen Prozesse an

(ESRI (c) 2022: 3). Allgemein kann das MDCS als vorteilhafte Strukturierung mit einer teilweise von ArcPy abweichenden Syntax betrachtet werden. Es hat damit einen ähnlichen Effekt wie die objektorientierte Strukturierung eines Skriptes in verschiedene Module, Funktionen und Klassen. Zur Anwendung muss vor allem ein Verständnis für die Initiierung des MDCS, die Übergabe von Parametern und die relativ intuitive Editierung der Konfigurationsdatei entwickelt werden. Die umfangreichen Teilskripte des MDCS-Pakets können weitgehend als „Blackbox“ behandelt werden. Die durch das MDCS erleichterte Umsetzung der „Best Practices“ könnte aber unter Umständen auch zu effizienteren Vorgängen mit geringeren Rechenlasten führen. Durch das Aufrufen der im Hauptmodul definierten Funktion *main()* mit der Übergabe von mindestens einem Parameter wird in ArcGIS Pro ein MD erstellt oder ein vorhandenes editiert. Bei dem Parameter handelt es sich um den Speicherort der Konfigurationsdatei, in welcher die Parameter der auf das MD anzuwendenden Befehle und seine Eigenschaften definiert sind. Listing 1 zeigt einen Ausschnitt aus einer solchen XML-Datei für die Erstellung eines MD.

```
<?xml version="1.0" encoding="utf-8"?>
<Application>
    <Name>erstelle_MD</Name>
    <Command>CM+AR</Command>
    <Workspace>
        <WorkspacePath>$Pfad$</WorkspacePath>
        <Geodatabase>Beispiel_GDB<Geodatabase>
        <MosaicDataset>
            <MosaicDatasetType>Source</MosaicDatasetType>
            <Name>Bsp_MD</Name>
            <SRS>25832</SRS>
            <num_bands>1</num_bands>
            <pixel_type>8_BIT_UNSIGNED</pixel_type
```

Listing 1: MDCS-Konfigurationsdatei ohne schließende Tags

Die im Tag *Command* enthaltenen Funktionen (Kürzel CM und AR) werden in der angegebenen Reihenfolge ausgeführt. Ihre Parameter werden in den zugehörigen Tags definiert. Variablen, die im Python-Skript definiert werden, haben das Format $Variablenname$ (hier z. B. bei *WorkspacePath*). Durch den ersten Befehl CM (*Create Mosaic Dataset*) wird ein MD mit den in *MosaicDataset* festgelegten Eigenschaften erstellt. Mit SP (*Set Properties*) werden seine in *DefaultProperties* definierten Eigenschaften gesetzt: *SRS* legt das Koordinatensystem des zu erstellenden MD fest, *num_bands* die Anzahl der Farbkanäle der referenzierten Raster und *pixel_type* die Farbtiefe.

3.4.4 GDAL Virtual Raster Format

VRT ähneln den MD in vielen Aspekten. Mit dem VRT-Treiber kann ein „virtueller Rasterdatensatz“ auf der Grundlage anderer Raster (z. B. im TIFF-Format oder andere VRT) mit Veränderung der Position und des Ausmaßes, Anwendung von Algorithmen z. B. zur Reprojektion oder Berechnung neuer Pixelwerte, sowie der Veränderung oder dem

Hinzufügen von Metadaten generiert werden. VRT sind XML-Dateien. Listing 2 zeigt eine solche Datei, welche ein 512 mal 512 Pixel großes Raster referenziert.

```
<VRTDataset rasterXSize="512" rasterYSize="512">
   <GeoTransform>440720.0, 60.0, 0.0, 3751320.0, 0.0, -60.0</GeoTransform>
   <VRTRasterBand dataType="Byte" band="1">
       <ColorInterp>Gray</ColorInterp>
       <SimpleSource>
          <SourceFilename relativeToVRT="1">raster.tif</SourceFilename>
          <SourceBand>1</SourceBand>
         <SrcRect xOff="0" yOff="0" xSize="512" ySize="512"/>
          <DstRect xOff="0" yOff="0" xSize="512" ySize="512"/>
```

Listing 2: VRT-Datei ohne schließende Tags

Der Tag *SRS* (vgl. Anhang 25) beschreibt das Koordinatensystem mit Parametern wie Referenzellipsoid, Maßstabsfaktor, False Easting, geodätischem Datum, Einheit und Nullmeridian. *GeoTransform* liefert die sechs Parameter einer ebenen Affintransformation zur Transformation zwischen Bildraum und georeferenzierten Koordinaten: Rechtswert des ULT-Pixels, horizontale Auflösung, Abszissendrehwinkel, Hochwert, Ordinatendrehwinkel und vertikale Auflösung. Der Parameter *dataType* bechreibt die Farbtiefe (hier 8 Bit). *ColorInterp* legt die farbliche Auswertung der Pixelwerte fest. Mit dem Wert *Palette (*statt wie hier *Gray)* erfolgt in einem weiteren Kindelement *ColorTable* eine Zuordnung der Farben zu den Grauwerten der Pixel. Jedes referenzierte Raster wird durch einen Tag *SimpleSource* mit Dateipfad, Anzahl der Farbkanäle, Position und Größe beschrieben (GDAL 2023: o.S.).

4 Implementierung des Workflows

Im Folgenden wird der prinzipielle Ablauf der Skripte für ArcGIS und QGIS erläutert.
Bei einigen weniger ersichtlichen Teilen wird dabei auch auf den Programmcode im Detail eingegangen.

4.1 Automatisierung mit ArcGIS Pro

In ArcGIS wird der Prozess durch die Pythondatei ***init.py*** angestoßen (vgl. Abb. 14). Diese wiederum ist in ArcGIS Pro in eine sogenannte *Toolbox* eingebunden. Für diese werden über die Bedienoberfläche Parameter und Eigenschaften, wie z. B. Parametertyp (*derived, input, output*) und Datentyp (z. B. Layer, Zeichenkette, Datei), festgelegt. Dieser Parameter wird dann im Skript über den Befehl *arcpy.GetParameter([Index])* abgerufen.

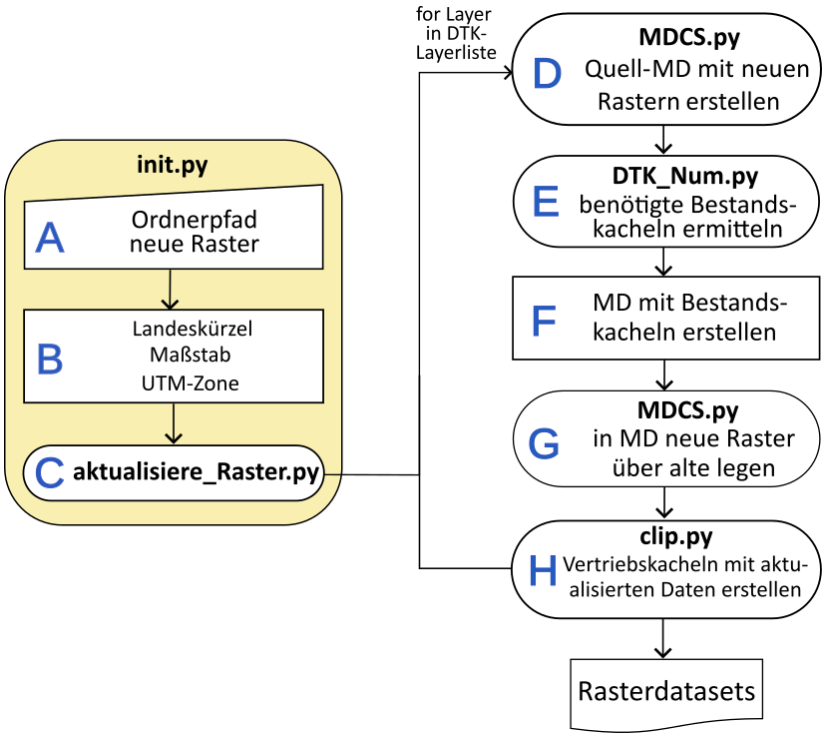
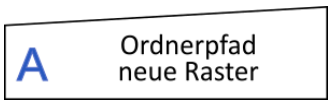
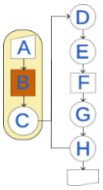


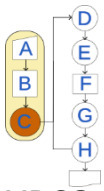
Abbildung 14: Ablauf des ArcGIS-Workflows (Quelle: eigene)



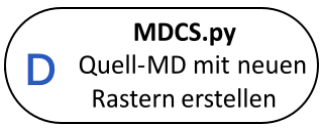
Bei diesem Workflow gibt es mit der Zeichenkette des Verzeichnisses der neuen Raster nur einen Parameter des Typs *input.*



Aus diesem Ordnerpfad lassen sich das Bundeslandkürzel und der Maßstab aufgrund der gegebenen Ordnernamen *dtk[Maßstab]/[Bundeslandkürzel]/[Übergabedatum]/[Kachelnummer]* (z. B. *dtk100/rp/2021-12-07/c5506)* durch einfaches String-Slicing über den Index ableiten. Aus diesen folgen dann auch die UTM-Zone, der Pfad für die Datei der Vertriebseinheiten und alle MDCS-Parameter. Nachdem diese neben anderen Variablen wie den Dateipfaden des Projektverzeichnisses und der MDCS-Konfigurationsdateien, sowie die Liste der DTK-Layerkürzel definiert sind, wird die Funktion *aktualisiere_Raster()* im gleichnamigen Modul mit den definierten Parametern ausgeführt.



Die Funktion *aktualisiere_Raster()* iteriert durch Liste der DTK-Layerkürzel (*['col', 'haus', 'brac',...]*). Das erste Element dieser Liste ist *'col',* so dass im ersten Schleifendurchgang der Farblayer bearbeitet wird. Der Wert der Variable *Pixeltyp* der MDCS-Parameter, mit dem die Farbtiefe der zu referenzierenden Raster festgelegt wird, ist deshalb zunächst auf *8_Bit_UNSIGNED* gesetzt, entsprechend dem in Abschnitt 3.4.1 erwähnten Standardwertebereich von TIFF-Rastern. Die Pixel können also mit ganzzahligen Werten von 0 bis 255 belegt sein. Nach dem ersten Schleifendurchlauf wird der Variablenwert für die Schwarz-Weiß-Layer auf 1 Bit gesetzt. Nachdem die Zeichenkette des Layerkürzels an die Liste der MDCS-Parameter angehängt wird ist der erste innerhalb der Schleife erfolgende Schritt der Aufruf des MDCS zur Erstellung des Quell-MD.



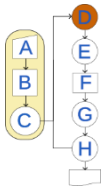


Abbildung 15 zeigt den im MDCS-Vorgang gekapselten Programmablauf. Mit der Erstellung des MD werden seine in Abschnitt 3.4.3 beschriebenen Grundeinstellungen gesetzt. In *DefaultProperties* werden unter anderem der erlaubte Datenkompressionstyp (*NONE*) und der Resampling-Algorithmus (*Nearest Neighbour*) festgelegt. Hier wird auch definiert, ob die referenzierten Raster bzw. das MD als Ganzes auf die Geometrien des Footprint- oder Boundary-Layers begrenzt werden und ob es Pixel innerhalb dieser Bereiche gibt, die keine Informationen liefern (*NoData*-Pixel). Der Vektorlayer *Footprint* enthält Polygongeometrien, die jeweils einem Raster zugeordnet sind, der Boundary Layer die Summe dieser Einzelflächen. Nach der Festlegung dieser Eigenschaften folgte ursprünglich der Befehl SE (*Set Environments*). Mit diesem wird mit dem Umgebungsparameter *ParallelProcessingFactor* bestimmt, auf wie viele Prozesse die Berechnung des

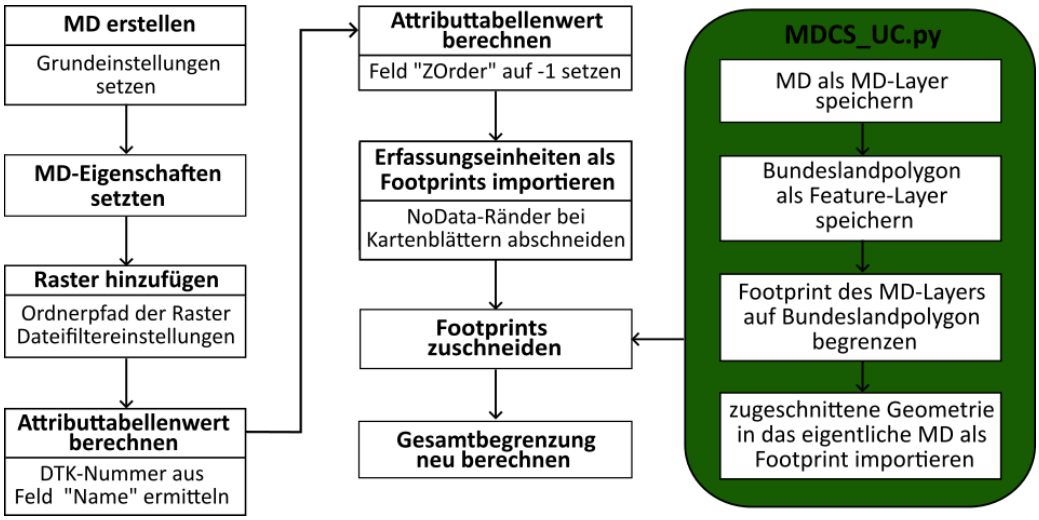


Abbildung 15: Ablauf des MDCS für das Quell-MD (Quelle: eigene)

darauffolgenden Befehls verteilt wird, und damit die Anzahl der genutzten Prozessorkerne (ESRI (d) 2023: o.S.). Testdurchläufe haben ergeben, dass die für jede Funktion automatisch auf den Prozessortyp angepasste Standardeinstellung eine schnellere Berechnung bewirkt als die manuelle Umstellung auf z. B. 8 Prozesse (vgl. Abschnitt 5.2). Die Veränderung dieser Einstellung wurde anfangs deshalb getestet, weil die folgende Funktion *Add Raster* (AR), mit welcher dem MD die zu referenzierenden Rasterdateien hinzugefügt werden, besonders rechenintensiv ist. In den Parameter-Tags der Funktion ist der Pfad der aktualisierten Raster in *data_path* angegeben (vgl. Anhang 6). Dieses Verzeichnis wird mit **$layer$.tif* nach den TIFF-Dateien des entsprechenden Layers durchsucht.

Mit dem Befehl *Calculate Values* (CV) folgt die Ableitung der Kachelnummer (Erfassungseinheiten) aus dem Dateinamen. Dazu wird die Python-Funktion *regular expressions* genutzt (vgl. Listing 3). Diese spart im Vergleich zu einer if-else-Abfrage einige Code-Zeilen ein, da sich die Nummerierung der Erfassungseinheiten je nach Format, Bundesland und Maßstab unterscheidet.

```
1   def splitLayerName(value):
2     mo = re.search('^dtk(25|50|100)[_]\d{4,5}[_]\d{4}[_]\d+[_][az]{2}|^[lc]?\d{4}')
```

Listing 3: regular expression zur Extrahierung der Kachelnummern

wobei ^ = “ignoriere nachfolgende”, | = “oder”, *\d{4}* = 4 Ziffern, + = mindestens 1, *lc* = “Kleinbuchstabe”, *?* = findet 0 oder 1 Instanz des vorhergehenden Zeichens

Die Suchphrase „in Worten” für die Dateinamen mit langen Kachelnummern (z. B. *dtk100_32390_5275_5_bw_acke.tif)* lautet also:
Ignoriere dtk25_, dtk50_ oder dtk100_ , finde 4 oder 5 Ziffern, Unterstrich, 4 Ziffern, Unterstrich, mindestens eine Ziffer;

Und für Dateinamen mit Kartenblattnummern (z. B. *c6706acke.tif*):

(Oder) ignoriere eventuell einen Kleinbuchstaben, finde 4 Ziffern.

Anschließend wird der Befehl *Calculate Values* noch ein zweites Mal verwendet um den *ZOrder*-Wert auf -1 zu setzen. Dadurch bekommen die neuen Raster im abgeleiteten MD Anzeigepriorität über die Bestandsraster mit dem Standardwert null.

Mit der nun als Attributwert vorhandenen Kachelnummer können die Raster den Polygonen der Erfassungseinheiten zugeordnet werden. Dies geschieht durch die Tags *target*- und *input_join_field* des folgenden Befehls *Import Geometry* (IG). Durch diesen werden die Erfassungseinheiten als Vektorgeometrien in das MD importiert. Sie ersetzen den Footprint-Layer und sorgen so für die gewünschte Begrenzung der neuen Rasterdateien, wenn diese im Kartenblattformat vorliegen. Andernfalls ergibt sich keine Übereinstimmung der Nummerierung, so dass der Befehl keine Auswirkung hat. Tests haben ergeben, dass sich die Laufzeit dadurch in solchen Fällen nicht verlängert.

ClipFootprints ist eine benutzerdefinierte Funktion, mit der die Flächen der Footprint-Polygone, die außerhalb des Bundeslandes liegen, abgeschnitten werden. Benutzerdefinierte Funktionen werden in der MDCS-Struktur in der Pythondatei ***MDCS_UC.py*** definiert (vgl. Abb. 16). Es bietet sich an, die Clipfunktion auf diese Weise in den MDCS-Vorgang zu integrieren, denn mit der Ausführung dieses Schrittes in den Skripten durch ArcPy-Befehle müsste das erste MDCS in zwei aufgeteilt werden. Da die Clipfunktion *arcpy.analysis.Clip()* (vgl. Anhang 3, Z. 74) nur als Layer gespeicherte Features und *Mosaic Layer* verarbeiten kann, müssen diese zunächst erstellt werden (vgl. Abb. 16, Schritt 1 und 2). Nach der Ausführung werden die begrenzten Geometrien wie vorher auch die Erfassungseinheiten als Footprint-Layer in das MD importiert. Außerdem werden sie zur späteren Verwendung als Shape-Datei in der *GDB* gespeichert (*source_footprints.shp*). *Build Boundary* (BB) aktualisiert im letzten, an *ClipFootprints* angeschlossenen Schritt den Boundary-Layer entsprechend der geänderten *Footprints*.

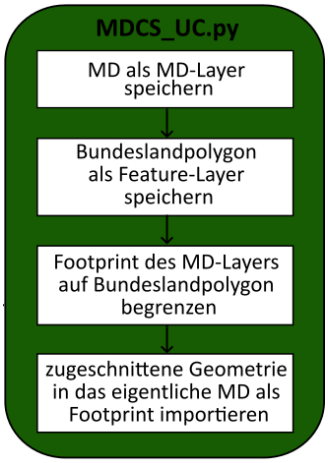
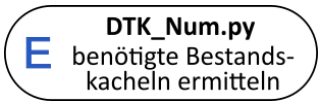
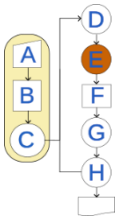


Abbildung 16: ClipFootprints (Quelle: eigene)





Als zweiter Schritt des Moduls *aktualisiere_Raster.py* folgt beim ersten Durchgang der Schleife die Ermittlung der relevanten Raster im Modul *DTK_Num.py.* Die Auswahl der Bestandskacheln, die mit den neuen Daten überlappen, reduziert die Rechenzeit der restlichen Prozesse deutlich, da so z. B. der rechenaufwändige Befehl *Add Raster* für das Quell-MD nicht mit den gesamten Bestandsdaten ausgeführt wird. Die

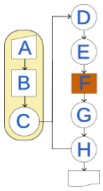
Auswahl der Kacheln erfolgt durch ihre Selektierung und die Speicherung der Attributwerte des Feldes *id* als *Array* der NumPy-Bibliothek (vgl. Listing 4, Z. 2 u. 4). Bei der geometrischen Selektion mit dem Modus *intersect* wird die Suchdistanz wie auch bei QGIS (vgl. Anschnitt 4.2.1) auf einen negativen Wert gesetzt, um die Auswahl von Kacheln, die lediglich die Umgrenzung des Maskenlayers berühren, zu vermeiden (Z. 3). Anschließend werden die in diesem *id*-Feld hinterlegten Nummern aus dem *Array* zur anschließenden Verwendung in der Funktion *SelectLayerByAttribute()* (vgl. Anhang 2, Z. 32), mittels einer For-Schleife in einer einzigen Zeichenkette gespeichert.

```
1  def ermittle_dtkNum(massstab, bundesland, datum, ws):
2        arcpy.management.SelectLayerByLocation("vertriebskacheln_lyr","INTERSECT",
3                                               r"%s\source_footprints"%ws, "-10")
4        kachel_liste = arcpy.da.TableToNumPyArray("vertriebskacheln_lyr", "id")
5        kachel_str = ""
6        for dtk_nummer in kachel_liste:
7            kachel_str += str(dtk_nummer)[1:-1]
```

Listing 4: Ermittlung und Speicherung benötigter Bestandskacheln

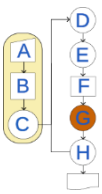
Die Selektion nach Attributwerten wird genutzt, um die relevanten Kacheln eines Mosaik-Layers zu aktivieren. Die relevanten Kacheln müssen aber erst mit *SelectLayerByLocation()* anhand des Vektorlayers der Vertriebseinheiten und der Vektorgeometrie *source_footprints.shp* ermittelt werden, weil diese Funktion keine *Mosaic Layer* als Dateneingang verarbeiten kann, die Bestandsdaten aber in dieser Form vorliegen (bzw. als MD, welches aber als *Mosaic Layer* gespeichert werden kann). Eine Alternative wäre die direkte Auswahl der TIFF-Dateien, sowie im QGIS-Skript (vgl. Abschnitt 4.2.1).

F MD mit Bestands-kacheln erstellen



Es folgt nun die Erstellung eines MD mit den ausgewählten Bestandskacheln (vgl. Anhang 2, Z. 41 u. 42). Diese können dem abgeleiteten MD nicht direkt über das MDCS hinzugefügt werden, da in der Konfigurationsdatei des MDCS-Pakets bei *Add Raster* nur Rasterdateien oder andere MD als Datenquelle akzeptiert werden, nicht aber *Mosaic Layer*. Beim ArcPy-Befehl *AddRaster()* hingegen ist das möglich. Da aber die Selektion der Kacheln nur mit dem *Mosaic Layer*, und nicht mit dem in der *GDB* gespeicherten MD an sich erfolgen kann, ist dieser Zwischenschritt notwendig.

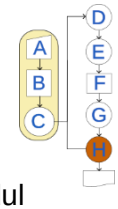
G **MDCS.py** in MD neue Raster über alte legen



Für die Zusammenführung der alten und neuen Daten im abgeleiteten MD sind deutlich weniger Prozesse nötig als beim Quell-MD. Die Befehlskette in der

Konfigurationsdatei (vgl. Anhang 7, Z. 4) ist *CM+SP+SE+AR+BB*, also *Create Mosaic Dataset, Set Properties, Set Environment, Add Rasters* und *Build Boundaries.* Die Raster werden über das vorab per ArcPy-Befehlen erstellte MD mit den Bestandsdaten und das Quell-MD hinzugefügt. Dabei ist *Raster Type* auf *Table* gesetzt, wodurch die einzelnen Rasterdateien der originären MDs mit ihren Attributwerten erhalten bleiben. Mit dem beim Quell-MD verwendeten Standard-Modus *Raster Dataset* würden die einzelnen Raster der Quell-MD dagegen als ein einziges zusammengefasst werden (ESRI (e) 2023: o.S.). Die individuelle Begrenzung durch die über die Kachelnummern zugeordneten Erfassungseinheiten wäre dann nicht mehr möglich.

clip.py
H Vertriebskacheln mit aktualisierten Daten erstellen



Im letzten Schritt werden mit den im abgeleiteten MD zusammengefügten Daten Kacheln im Format der Vertriebseinheiten erstellt. Dazu wird wieder über die im Modul ***DTK_Num.py*** erstellte Liste der relevanten Vertriebseinheitennummern iteriert (vgl. Listing 5, Z. 2). In der Schleife wird die entsprechende Kachel des Vektorlayers selektiert (Z. 4). Dann wird der Funktion *arcpy.management.Clip()* der Vektorlayer der Vertriebseinheiten als Overlay-Geometrie übergeben, wobei standardmäßig nur das selektierte Feature genutzt wird. Dateneingang ist das abgeleitete MD (Z. 6-8). Zuletzt wird beim Farblayer noch die Funktion Add*Colormap()* angewendet, um die Grauwerte den zugehörigen Farben der DTK zuzuordnen (Zeilen 10 u. 11). Bei QGIS wird diese Zuordnung, die wie erwähnt in den TIFF-Dateien gespeichert ist, automatisch aus diesen übernommen.

```
1  def clip_D(kachel_liste):
2      for dtk_nummer in kachel_liste:
3          dtk_nummer_str = str(dtk_nummer)[1:-2]
4          arcpy.management.SelectLayerByAttribute("vertriebskacheln_lyr", 'NEW_SELECTION',
5                                                  f"id LIKE {dtk_nummer_str}")
6          arcpy.management.Clip("%s\D_%s"%(arcpy.env.workspace, layer), "#",
7                                f"{arcpy.env.workspace}/{layer}_{dtk_nummer_str[1:-1]}",
8                                "vertriebskacheln_lyr", "#", "ClippingGeometry", "NO_MAINTAIN_EXTENT")
9          if layer == "col":
10             arcpy.management.AddColormap(f"{arcpy.env.workspace}/col_{dtk_nummer_str[1:-1]}",
11                                          "#", r"M:\Pfad\DTK-N_PQS1.0.clr")
```

Listing 5: Funktion zur Erstellung der Ergebniskacheln

4.2 Automatisierung mit QGIS

In der Folge wird der QGIS-Workflow in seiner ursprünglichen Form beschrieben. Beim nach seiner Erstellung erfolgten Vergleich der Laufzeiten stellte sich heraus, dass die Anwendung der Vektorisierungsfunktion *polygonize()* recht ineffizient ist. Nach verschiedenen in Abschnitt 5.2 näher beschriebenen Lösungsansätzen wurde das Skript in Abschnitt 4.2.2 erstellt, welches von allen Alternativen die kürzesten Berechnungszeiten erfordert.

4.2.1 Workflow mit Vektorisierung

Die Initiierung erfolgt durch die Datei ***init.py*** (vgl. Abb. 17). Diese wird über die QGIS-Bedienoberfläche im Python-Editor geladen und ausgeführt.

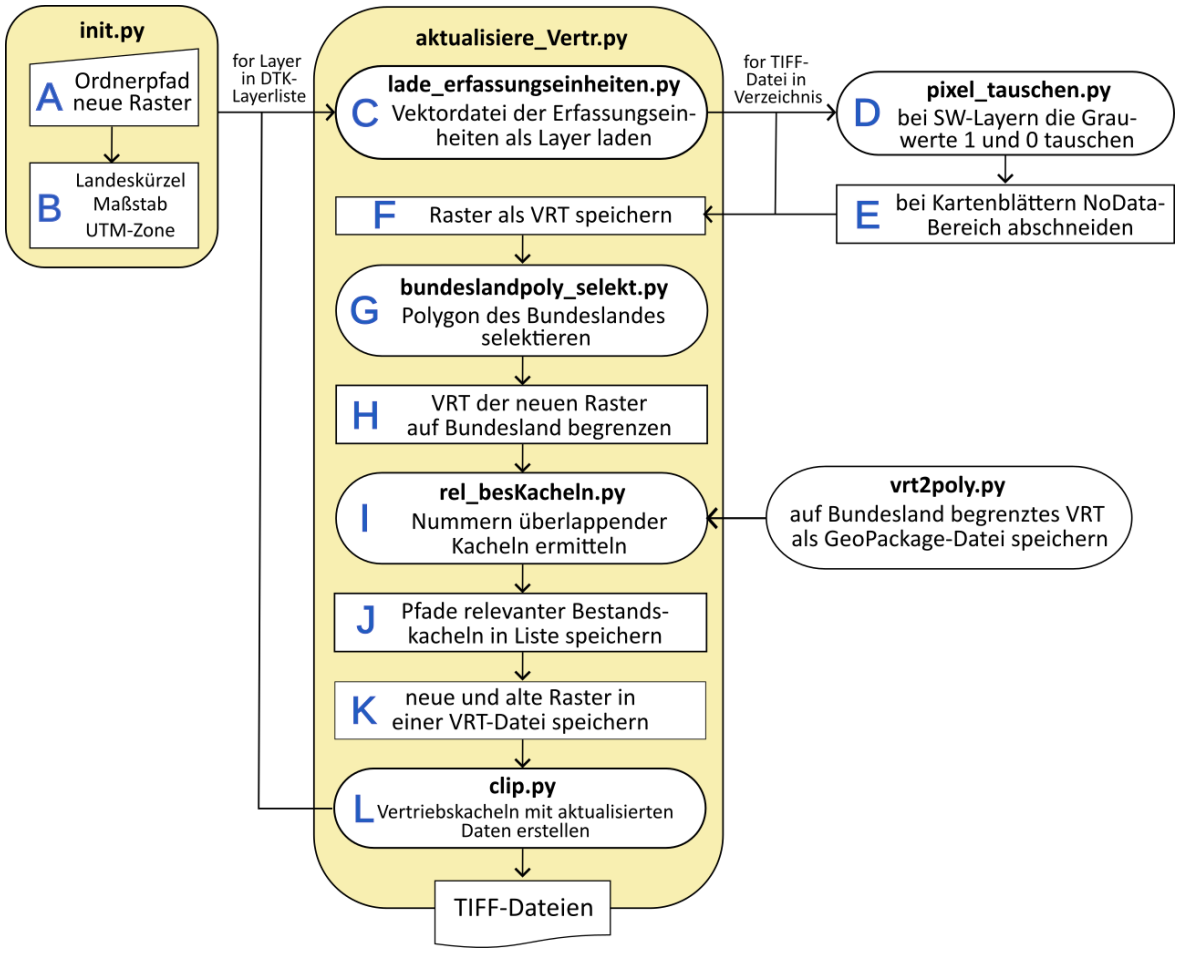
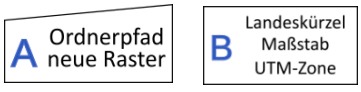
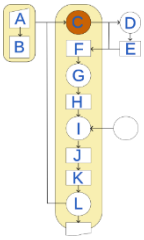


Abbildung 17: Ablauf der QGIS-Workflows (Quelle: eigene)



Zunächst wird wie bei ArcGIS der Ordnerpfad der neuen Raster abgefragt und aus diesem das Bundeslandkürzel, der Maßstab und die UTM-Zone abgeleitet. Dann wird mit der Funktion *aktualisiere_Vertriebskacheln()* in **aktualisiere_Vertr.py** über die DTK-Layer iteriert.

C **lade_erfassungseinheiten.py**
Vektordatei der Erfassungseinheiten als Layer laden



In dieser Funktion werden beim Farblayer zunächst die Erfassungseinheiten als Vektorgeometrie für diesen und die restlichen Schleifendurchgänge in das Projekt geladen. Dann wird durch die im Verzeichnis vorhandenen TIFF-Dateien mit der entsprechenden Layerbezeichnung iteriert.

D **pixel_tauschen.py**
bei SW-Layern die Grauwerte 1 und 0 tauschen

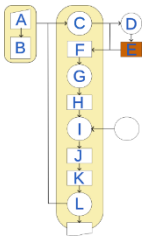
Bei den Schwarz-Weiß-Layern werden die Grauwerte der Pixel ausgetauscht, damit sie denen der Bestandsraster entsprechen (bei ArcGIS wird der NoData-Wert automatisch erkannt). Beim Farblayer ist der NoData-Wert der Vertriebsdaten dagegen wie bei den Eingangsdaten mit Null belegt. Für den Tausch wird im Modul ***pixel_tauschen.py*** die GDAL-Funktion *rastercalculator()* mit der auf jeden Pixel angewandten Berechnungsformel *A == 0* benutzt (vgl. Listing 6, Z. 4).

```
1    def tausche_pix(proj, dtk_nr, layer, pfad):
2        pfad_neu = r"%s\Zwischenschritte\Raster\neu\%s_%s.tif"%(proj, dtk_nr, layer)
3        processing.run("gdal:rastercalculator", {'INPUT_A': pfad, 'BAND_A': 1,
4                       'FORMULA': '(A==0)', 'RTYPE': 0, 'NO_DATA': 1, 'OUTPUT': pfad_neu})
5        return pfad_neu
```

Listing 6: Definition der Funktion tausche_pix

Durch diesen logischen Ausdruck wird der Grauwert jedes Pixels des (einzigen) Farbkanals A geprüft. Ist der Wert 0, ist das Ergebnis der Abfrage 1 für „wahr“, andernfalls 0 für „falsch“. Der Grauwert des geprüften Pixels wird durch den Zahlenwert des Abfrageergebnisses ersetzt, was zum gewünschten Tausch führt.

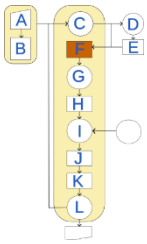
E bei Kartenblättern NoData-Bereich abschneiden



Als nächstes folgt das Abschneiden der informationslosen Ränder der Kartenblätter. Dazu werden die ersten fünf oder vier (DTK25) Zeichen des Dateinamens gespeichert. Diese entsprechen im Falle der Kartenblätter der Nummer der Erfassungseinheit. Mit dieser wird dann mit der Funktion *selectByExpression()* die Kachel des zuvor geladenen Vektorlayers der Erfasssungseinheiten selektiert. Handelt es sich um UTM-Kacheln, bleibt die Selektion leer und der Pfad der Rasterdatei wird direkt in einer Liste gespeichert. Andernfalls wird davor noch die GDAL-Funktion *cliprasterbymasklayer()* auf die TIFF-Datei des Kartenblatts angewendet, mit den Erfassungseinheiten als Vektor-Maskenlayer. Damit, wie in ArcGIS, nur das aktive Feature als Overlay-Maske genutzt wird, muss der Maskenlayer mit der Funktion *QgsProcessingFeatureSourceDefinition()* und dem Parameter *selectedFeaturesOnly* definiert

werden. Wichtig ist auch die Definition des Koordinatensystems der Eingangsdatei über die UTM-Variable. Andernfalls wird die Funktion bei Dateien der UTM-Zone 33 abgebrochen, da das Projekt auf Zone 32 eingestellt ist.

F Raster als VRT speichern



Anschließend wird anhand der Liste mit den Dateipfaden ein VRT aller Eingangsraster erstellt. Aufgrund der Anwendung der Clip-Funktion auf die TIFF-Dateien, und wegen der Referenzierung anderer VRT im VRT selbst, sind dessen Pfade nicht wie in Abschnitt 3.3.1 beschrieben im XML-Element *SimpleSource*, sondern im Tag *ComplexSource* mit der zusätzlichen Eigenschaft *NODATA* gespeichert.

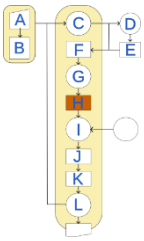
G **bundeslandpoly_selekt.py**
Polygon des Bundeslandes selektieren

Im ersten Schleifendurchlauf wird im Modul ***bundeslandpoly_selekt.py*** das betroffene Bundeslandpolygon über die Funktion *selectByExpression()* ausgewählt (vgl. Listing 7, Z. 5).

```
1    def selekt_BLpoly(bundesland, proj, massstab):
2        laenderpolygone = QgsVectorLayer(r'%s/Ressourcen/laenderpolygone%s.shp'\
3                                         %(proj,massstab), "laenderpolygone", "ogr")
4        QgsProject.instance().addMapLayer(laenderpolygone)
5        laenderpolygone.selectByExpression(f'"LAND" = \'{bundesland}\'')
6        pass
```

Listing 7: Funktion zur Selektierung des Bundeslandes

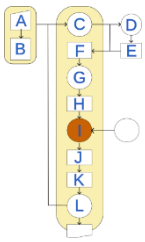
H VRT der neuen Raster auf Bundesland begrenzen



Dieses wird dann ebenfalls als Maskenlayer in der darauffolgenden Clip-Funktion angewendet. Dateneingang ist jetzt die VRT-Datei aller Eingangsraster. Aufgrund der beschriebenen Gegebenheiten der Eingangsraster in Bezug auf den NoData-Wert muss zwischen Farb- und Einzellayern (null für Farblayer, *None* für Schwarz-Weiß-Layer) unterschieden werden. Durch die direkte Anwendung der Clip-Funktion auf das VRT erhält das Ergebnis-VRT der Funktion (vgl. Anhang 25) verschiedene neue Elemente: Den Elementen *VRTDataset* und *VRTRasterBand* werden jeweils die Unterklassen *VRTWarpedDataset und VRTWarpedRasterBand* hinzugefügt. Der neue Tag *GDALWarpOptions* enthält eine Vielzahl von Kindelementen, wie z. B. den Tag *Cutline*. Dieser wird hinzugefügt, wenn sich die Raster tatsächlich mit der Bundeslandgrenze überschneiden, da in ihm die angewendete Clip-Geometrie durch Koordinatenwertepaare definiert wird. Weitere in *GDALWarpOptions* festgelegte Parameter sind u.a. *warpMemoryLimit* (maximaler für das warp-API nutzbarer Arbeitsspeicher für das Caching in Byte), *GenImgProjTransformer* in *ApproxTransformer* (Parameter einer Näherungsweisen

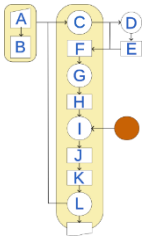
Affintransformation für die referenzierten Raster mit maximaler Abweichung), (GDAL 2023: o.S.).

rel_besKacheln.py
Nummern überlappender Kacheln ermitteln



Anschließend werden die sich mit den neuen Daten überlappenden Vertriebseinheiten ermittelt. Dazu wird die Funktion *SelectLayerByLocation()* angewendet. Mit dieser können wie bei ArcGIS nur Vektorlayer als Overlay-Layer verarbeitet werden. Benötigt wird also die Geometrie des auf die Bundeslandfläche und ggf. auf Kartenblattgeometrien begrenzten VRT als einfaches Polygon. Diese Umwandlung erfolgt im Modul ***vrt2poly.py***.

vrt2poly.py
auf Bundesland begrenztes VRT als GeoPackage-Datei speichern



Da bei den Funktionen *TileIndex()* und *ExtractLayerExtend(),* mit welchen Polygone des rechteckigen, maximalen Ausmaßes eines Rasters erstellt werden können kein NoData-Bereich zugewiesen werden kann, wurde die GDAL-Funktion *polygonize()* gewählt. Eine von mehreren, in Abschnitt 5.2 erläuterten Alternativen ist die Funktion *RasterPixelsToPolygons()* mit der anschließenden Anwendung von *Dissolve().* Da diese Funktion aber unabhängig vom Grauwert für jedes Pixel ein Polygon erstellt, ist sie sehr rechenaufwändig (mehr noch als *polygonize()*). Die Umwandlung erfolgt in drei Schritten (vgl. Abb. 18 und Listing 8).

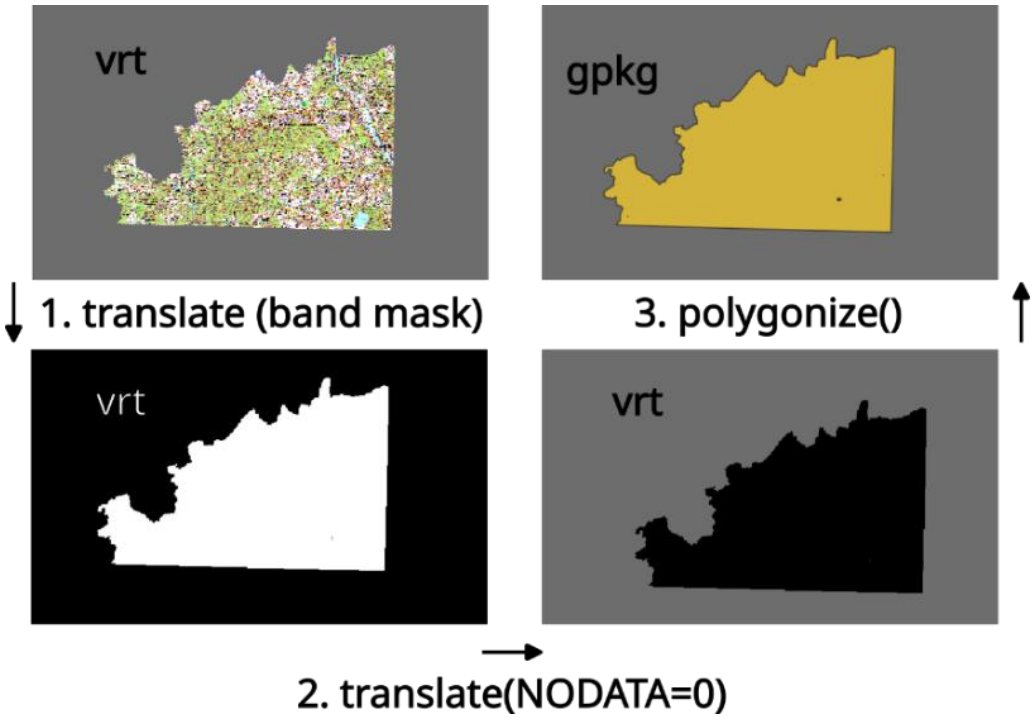


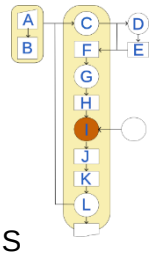
Abbildung 18: Erstellung eines Umrisspolygons des VRT (Quelle: eigene)

```
1   def vrt2poly(proj, neue_Raster_p):
2   processing.run('gdal:translate',
3                  {'INPUT': r"%s/Zwischenschritte/Raster/neu/r_clip2x_col_%s.vrt"%(proj, neue_Raster_p[-3:]),
4                  'OUTPUT': r"%s/Zwischenschritte/Raster/neu/translate_mask.vrt" % proj,  'EXTRA': '-b mask'})
5   processing.run('gdal:translate',
6                  {'INPUT': r"%s/Zwischenschritte/Raster/neu/translate_mask.vrt" % proj,
7                  'OUTPUT':r"%s/Zwischenschritte/Raster/neu/translate2.vrt"%proj,'EXTRA':'-b 1','NODATA':0})
8   processing.run('gdal:polygonize',
9                  {'INPUT': r"%s/Zwischenschritte/Raster/neu/translate2.vrt" % proj,
10                 'OUTPUT': r"%s/Zwischenschritte/Vektor/polygon_footprint.gpkg" % proj, 'BAND': 1})
```

Listing 8: Funktion vrt2poly

Zuerst werden alle Grauwerte außer null des VRT durch den Parameter *band mask* der Funktion *translate* auf 255 gesetzt. Das Ergebnis ist wiederum ein VRT mit nur zwei Grauwerten (0 und 255). Da bei Anwendung des *mask*-Parameters nicht gleichzeitig ein NoData-Wert zugewiesen werden kann, wird die Funktion ein zweites Mal ausgeführt. Im dritten Schritt wird die Funktion *polygonize* angewendet mit einer GeoPackage-Datei als Funktionsausgabe. Die Grauwerte des VRT müssen deshalb zunächst auf einen reduziert werden, da die Funktion *polygonize()* zusammenhängende Pixel mit gleichem Wert in Polygone wandelt. Ohne die Reduzierung würde also der eigentliche Karteninhalt ins Vektorformat umgewandelt werden.

rel_besKacheln.py
Nummern überlappender Kacheln ermitteln



Nun wird die Funktion *SelectLayerByLocation()* zwei Mal angewandt, zuerst mit dem Prädikat *intersect (0)* und der Methode *creating new selection (0)*, dann mit dem Prädikat *touch (4)* und der Methode *removing from current selection (3)* (QGIS 2023 (b): o.S.). Ohne die Deselektion der Geometrien, die die Eingangsgeometrie von außen berühren, würden alle angrenzenden Kacheln aktiviert bleiben (vgl. Abb. 19), was im finalen Schritt zur Erstellung leerer TIFF-Dateien führen würde.

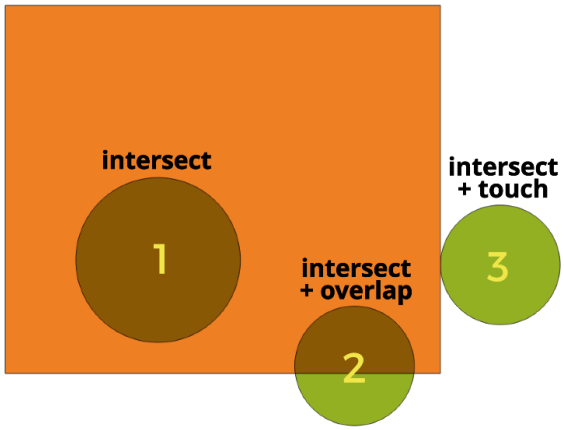


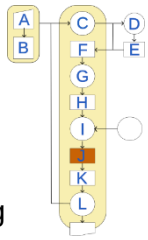
Abbildung 19: Illustrierung der Prädikate von SelectLayerByLocation() (QGIS 2023: o.S.)

Als letzter Schritt des Moduls ***rel_besKacheln.py*** werden die Nummern des Attributfeldes *id* der selektierten Kacheln in einer Liste gespeichert, welche als Ergebnis der Funktion ausgegeben wird. Dazu wird mit einer for-Schleife über die aktiven Features des Vektorlayers iteriert (vgl. Listing 9).

```
1   for feature in vertr_kach.selectedFeatures():
2           dtk_nummern.append(feature['id'])
3   return dtk_nummern
```

Listing 9: Speichern der Kachelnummern

J Pfade relevanter Bestands-kacheln in Liste speichern

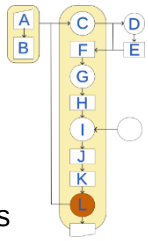


Als nächster Schritt des Hauptmoduls *aktualisiere_Vertr.py* wird das Verzeichnis der Bestandsraster anhand dieser Liste der Kachelnummern nach den benötigten Dateien durchsucht. Dazu wird mit der Funktion *walk*() der Bibliothek *os* (operating system*)* über die Pfade und Dateinamen aller Unterordner des Verzeichnisses iteriert (vgl. Anhang 12, Z. 94-100). Die Pfade der den if-Bedingungen entsprechenden Dateien werden wiederum in einer Liste gespeichert. Eine Alternative wäre die Bearbeitung der bereits existierenden VRT der Bestandsdaten (vgl. Abschnitt 3.1, Abb. 9).

K neue und alte Raster in einer VRT-Datei speichern

Als nächstes werden die als VRT gespeicherten neuen und die als TIFF-Dateien vorliegenden alten Raster in einem VRT zusammengeführt. Die Pfade der zu referenzierenden Dateien werden der Funktion *BuildVRT*() in einer Liste übergeben. Die in ArcGIS über das ZOrder-Attribut gesteuerte Anzeigepriorität der Rasterdaten wird hier durch die Reihenfolge der Listenelemente bestimmt (die letzte Datei ist sichtbar bzw. oben).

clip.py
L Vertriebskacheln mit aktualisierten Daten erstellen



Als letztes erfolgt die Erstellung der fertigen TIFF-Kacheln. Dafür wird wieder über die Liste der relevanten Vertriebseinheitennummern iteriert (vgl. Listing 10, Z. 2). Bei jedem Schleifendurchgang wird mit *vertr.selectByExpression()* eine Kachel des Maskenlayers selektiert (Z. 4) und dann die Clip-Funktion auf das im vorigen Schritt erstellte VRT angewendet (Z. 8-12). Die Kachelnummernliste wird von der Funktion *aktualisiere_Vertriebskacheln()* als Ergebnis ausgegeben, um ihr bei den folgenden Schleifendurchläufen für die Schwarz-Weiß-Layer als Parameter übergeben zu werden.

```
def clip_D(dtk_nummern, layer, proj, massstab):
    for dtkNummer in dtk_nummern:
        vertr = QgsProject.instance().mapLayersByName("vertriebskacheln%s" % massstab)[0]
        vertr.selectByExpression(f'"id" = \'{dtkNummer}\'')
        processing.run("gdal:cliprasterbymasklayer",
                       {'INPUT':r'%s/Zwischenschritte/Raster/zusammen/vr_%s.vrt'%(proj, layer),
                        'MASK': QgsProcessingFeatureSourceDefinition(
                                r'%s\Zwischenschritte\Vektor\vk%s.shp'%(proj,dtkNummer),
                                selectedFeaturesOnly=True),
                        'CROP_TO_CUTLINE': True,
                        'OUTPUT': r"%s/Ergebnis/raster_%s_%s.tif" % (proj, dtkNummer, layer)})
        iface.addRasterLayer(r"%s/Ergebnis/raster_%s_%s.tif" % (proj, dtkNummer, layer)
```

Listing 10: Funktion für die Erstellung der Ergebniskacheln

4.2.2 Workflow ohne Vektorisierung

Bei dieser Variante des Ausgangsskriptes werden die ausgewählten Polygone der Erfassungseinheiten wie die Raster auf die Bundeslandfläche reduziert und anschließend vereinigt, um dann für die Auswahl der Bestandskacheln genutzt zu werden. Dadurch ist im Vergleich zum Originalskript und den in Abschnitt 5.2 genannten Alternativen keine rechenintensive Vektorisierung mit Berücksichtigung des NoData-Wertes der Raster nötig, es werden nur die tatsächlich benötigten Kacheln selektiert und die Bekanntheit der Eigenschaften der Eingangsdaten muss nicht vorausgesetzt werden. Die Vektor-Clip-Funktion benötigt im Vergleich zu der für Raster wenig Rechenzeit. Außerdem kann die im Falle von Kartenblättern zweifache Anwendung der Funktion *gdal_cliprasterbymasklayer()* nun auf eine reduziert werden. Insgesamt ist die Laufzeit im Vergleich zur ursprünglichen Methode bei allen Tests deutlich reduziert (vgl. Abschnitt 5.2, Tab. 1).

Zur Umsetzung dieses alternativen Workflows sind folgende Änderungen des zuvor beschriebenen Ausgangsskriptes notwendig: Die Erfassungseinheiten der Kartenblätter werden nach Schritt D als GeoPackage-Datei gespeichert und nach der Schleife vor Schritt F mit *mergevectorlayers()* zusammengefasst. Für DOP und UTM-orientierte Erfassungseinheiten wird der Einfachheit halber die Funktion *tileindex()* angewendet. Alternativ könnten die Erfassungseinheiten geladen und selektiert werden, was aber deutlich mehr Code-Zeilen ohne Verkürzung der Laufzeit bedeuten würde. Nach Schritt G wird die Vektor-Clip-Funktion mit den Länderpolygonen als Overlay-Layer auf diese Geometrie angewandt. Anschließend muss noch die Funktion *dissolve()* angewendet werden, da sonst bei der zweiten Anwendung der Funktion *selectByLocation()* bei Schritt I auch Kacheln deselektiert werden, die die “inneren Grenzen” des Vektorlayers zwischen den einzelnen Features berühren.

4.3 Anwendung auf DOP

Die DOP werden mit vier Farbkanälen (Rot, Grün, Blau und Infrarot) mit einer Farbtiefe von jeweils 8 Bit gespeichert. Bei den VRT werden diese Eigenschaften automatisch gesetzt. Für ArcGIS werden zwei weitere Konfigurationsdateien mit den entsprechenden Eigenschaften erstellt (vgl. Anhang 8 u. 9). Sowohl bei ArcGIS als auch bei QGIS wird die Layerliste für DOP-Daten mit *['DOP']* definiert und bei *if layer == 'col'* jeweils or *layer == 'DOP'* hinzugefügt. Das Eingangsformat sind bei allen Ländern am UTM-Gitter orientierte 2 mal 2 km - Kacheln, das Vertriebsformat 1 mal 1 km - Kacheln (vgl. Abb. 20).

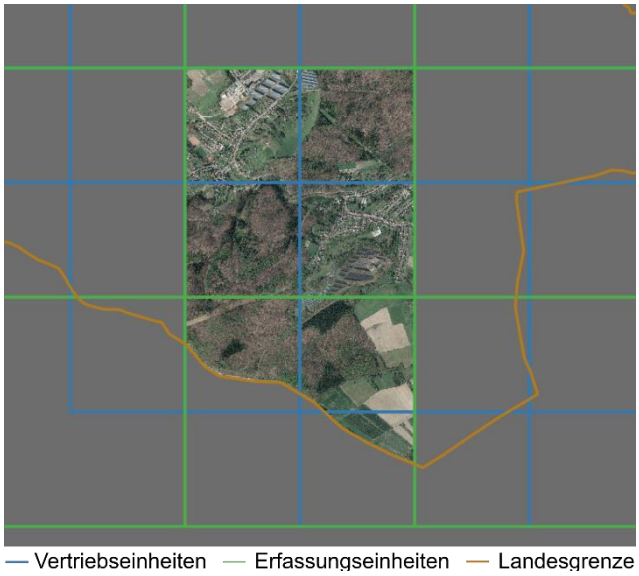


Abbildung 20: Prozessierung der DOP-Daten in QGIS (Quelle: eigene)

Anders als bei der DTK werden die aktualisierten DOP-Daten im Eingangs- und Vertriebsformat auf verschiedenen Servern gespeichert. Die Skripte wurden so angepasst, dass die Daten im Eingangsformat als Bestandsdaten genutzt werden. Da deren Kachelnummern mit denen der Eingangsdaten übereinstimmen, können die benötigten Bestandskacheln direkt über diese ausgewählt werden (vgl. Listing 11, Z. 7).

```
1   l_raster_alt = []
2   if layer == "DOP":
3     for root, dirs, files in os.walk(r"...\%s"%bundesland.lower()):
4        for file in files:
5           if file.endswith(".tif"):
6              Dateiname = file[10:12]+file[13:23]
7              if Dateiname == DOP_dateinamen[0] or Dateiname == DOP_dateinamen[1]:
8                      l_raster_alt.append(os.path.join(root, file))
```

Listing 11: direkte Auswahl der Bestandskacheln über die Kachelnummer

Die Liste *DOP_Dateinamen* wird bei der Suche der Eingangsdateien erstellt (vor Schritt D). Für die Erstellung der Vertriebskacheln in ***clip.py*** werden dann wieder die geometrischen Auswahlmethoden für den Vektorlayer der Vertriebseinheiten genutzt.

5 Vergleich

5.1 Handhabung

Ein zentraler Vorteil der VRT ist, dass sie durch das XML-Format sowohl für den Menschen als auch für Maschinen les- und editierbar sind. Wenn auch in diesem Workflow nicht angewandt, können sie also plattformunabhängig programmatisch oder auch händisch verändert, oder Teile der enthaltenen Information genutzt werden. So könnte z. B. die Auswahl der Bestandsdaten in QGIS nicht durch die Iteration über alle Dateien (vor Schritt D), sondern durch eine Kopie des jeweiligen Bestands-VRT und das anschließende Löschen der Source-Tags aller nicht benötigten Dateien erfolgen. Ein allgemeiner Nachteil der Handhabung von QGIS sind die häufiger als bei ArcGIS vorkommenden Programmabstürze bei Überlastung. Außerdem gibt es keine (in ArcGIS vorhandene) Möglichkeit, die Ausführung eines Skriptes vor Beendigung abzubrechen. Dieses Problem könnte allerdings durch die Implementierung des Skriptes als Plugin mit Benutzeroberfläche, ähnlich der *Toolbox* von ArcGIS gelöst werden. Ein weiterer Nachteil der Handhabung von QGIS sind die in 5.2 genannten, teilweise langen Ladezeiten zur Darstellung der VRT.

Die MD von Esri sind im Gegensatz zu den VRT ein geschlossenes System. Sie liegen in einer *File Geodatabase*, sind also im Dateiverzeichnis des Betriebssystems nicht zugänglich (vgl. Abschnitt 2.3). Ihr Inhalt lässt sich nur über die Programmierschnittstelle oder die Bedienoberfläche von ArcGIS Pro ausgeben und ist auch nur so veränderbar. Um z. B. die Dateipfade der referenzierten Raster anzuzeigen, müssen sie als Tabelle exportiert werden. Ihr Vorteil liegt in den umfangreichen und oft intuitiv bedienbaren Elementen der grafischen Oberfläche von ArcGIS Pro, wie z. B. dem *Modelbuilder* oder Drag- und Dropfunktionen. Dies ist besonders für kleinere oder nicht wiederholte Vorgänge und Tests vorteilhaft. Aber auch für die Automatisierung von Workflows mit größeren Datenmengen stehen mit dem MDCS und ArcGIS Enterprise durch die Nutzbarkeit von sowohl lokalen Netzwerkarchitekturen als auch Cloud-Services umfangreiche Mittel zur Verfügung. Die einfache Erweiterung des ArcGIS-Workflows auf DOP-Daten durch Abänderung der Konfigurationsdateien bestätigt die Effizienz der MDCS-Struktur für die Erstellung von MD.

5.2 Laufzeit

Tabelle 1 zeigt die Prozessierungsdauer für die Workflows in ArcGIS und QGIS mit verschiedenen Testdaten und der Berechnung der Ergebniskacheln des Farblayers und einem Einzellayer. Für QGIS sind die Laufzeiten des Workflows ohne (linke Spalte) und mit Vektorisierung gegenübergestellt. In ArcGIS wird die Laufzeit nach Ausführung der Skript-Toolbox angezeigt. Für die Anzeige in QGIS werden die Zeiten zu Beginn und am Ende des Skriptes mit *datetime.datetime.now()* in Variablen gespeichert und die Differenz ausgegeben.

Tabelle 1: Vergleich der Laufzeiten

DTK (1 Band, 8 / 1 Bit)				**Kartenblattformat**			**Dauer** [(Stunde:) Minute:Sekunde]		
Nr.	*Anz. K.*	*Größe (Px)*	*Größe Farblayer (MB)*	*Maßstab*	*Land*	*UTM-Zone*	*QGIS*	*QGIS (polyg.)*	*ArcGIS*
1	1	9275 x 9064	5,34	1:100	Berlin	33	00:30	1:50	6:33
2	1	9674 x 9149	4,67	1:100	Rh.-Pf.	32	1:11	4:52	9:06
3		wie Test 2 mit allen Layern					7:44	21:37	1:43:33
4	1	9635 x 9211	5,99	1:50	Thüringen	32	1:00	8:36	5:13
5	2	9635 x 9211	5,99	1:50	Thüringen	32	2:05	1:24:45 (1:22:00)	18:57
6	wie Test 5 mit Anwendung von polygonize() auf einzelne Raster ***aktualisiere_Vertr_e.py***						-	14:29 (12:19)	-
7	wie Test 6 ohne Reduzierung auf Kartenblätter/Landesfläche ***aktualisiere_Vertr_o.py***							1:35 (0:11)	9:30
8	wie Test 5 mit *tileindex()* statt *polygonize()* ***aktualisiere_Vertr_t.py***							2:21	-
9	4	9255 x 9075	4,02	1:25	Berlin	33	5:45	38:41	25:30
10	5	9674 x 9149	4,67	1:100	Rh.-Pf.	32	7:12	19:51	21:21
UTM-Kachel-Format									
11	1	1600 x 1600	0,23	1:25	Bremen	32	0:32	2:48	9:14
12	4	1600 x 1600	0,30	1:50	Hessen	32	0:26	4:03	5:02
13	5	1600 x 1600	0,27	1:100	Bremen	32	0:37	2:14	7:30
DOP (4 Bänder à 8 Bit)									
14	2	10000 x 10000	400	-	Saarland	32	0:58	2:11	4:59

Der für die Berechnung benutzte Computer ist mit 17 GB verfügbarem physischen Arbeitsspeicher, 20 GB verfügbarem virtuellen Arbeitsspeicher, und einem Intel Xeon E5-2643 Prozessor mit 8 Kernen und 3,4 Ghz ausgestattet. Das Betriebssystem ist Microsoft Windows Server 2019 Standard.

Die Laufzeiten des QGIS-Skriptes ohne Vektorisierung sind bei allen Tests um ein Vielfaches kürzer als die des ArcGIS-Workflows. Da die Ergebnisraster nicht als MD oder VRT, sondern als Raster Datasets und TIFF-Dateien gespeichert werden, werden sie in beiden Programmen relativ schnell dargestellt. Die Darstellung in MDs und VRT, die für Test- und Kontrollzwecke durchaus wichtig sein kann, ist dagegen bei beiden Programmen eingeschränkt. So werden die Raster der MD ohne Berechnung von Bildpyramiden (durch welche die Laufzeit erheblich verlängert wird) erst ab einer festgelegten Zoom-Stufe dargestellt. In QGIS erfolgt die Darstellung der VRT-Raster zwar bei allen Zoom-Stufen, ist aber, besonders nach Anwendung der Funktion *clipRasterByMaskLayer()* mit oft sehr langen Ladezeiten und entsprechender Auslastung des Arbeitsspeichers verbunden.

Aufgrund der besonders langen Laufzeiten des ursprünglichen QGIS-Workflows mit Vektorisierung bei Test 5 und 9 wurde deren Ursache erörtert und verschiedene Alternativen (Tests 6-8) getestet. Diese Überlegungen führten schließlich zu dem Skript ohne Vektorisierung, welches bei allen Tests die kürzesten Berechnungszeiten aufweist und dessen Code außerdem kürzer und übersichtlicher ist (Abschnitt 4.2.2). Die langen Laufzeiten des originalen Skripts werden wie bereits erwähnt durch die Funktion *polygonize()* verursacht (z. B. 1:22 von 1:24 Stunden bei Test 5). Auffällig ist vor allem der Anstieg der Laufzeiten von Test 1 zu 9 und von Test 4 zu 5, welcher sich nicht proportional zur Anzahl der Kacheln ähnlicher oder gleicher Größe verhält. Auch die Laufzeiten des ArcGIS-Skriptes dieser Tests zeigen diese Disproportionalität, wenn auch in geringerem Maße. Die lange Berechnungszeit von *polygonize()* tritt also bei der Kombination von mehreren Eingangskacheln und der Belegung des Bereichs außerhalb des Bundeslandes mit dem NoData-Wert ein. Der Zusammenhang dieser beiden Komponenten ist unklar, folgende Ursachen können aber ausgeschlossen werden:

- Ein großer Abstand zwischen den einzelnen Kacheln, da z. B. auch zwischen den fünf Kacheln von Test 10 ein großer NoData-Bereich vorliegt, der Anstieg der Berechnungszeit aber proportional zur Anzahl der Kacheln ist
- Die Reprojektion von UTM-Zone 33 zu 32, da die TIFF-Datei vom Land Thüringen bereits im System der Zone 32 vorliegt.
- Das Dateneingangsformat von *polygonize()*, da sich die Laufzeit bei Übergabe einer TIFF- statt VRT-Datei im Ausgangsskript nicht verkürzt.
- Die Berechnung einer größeren Zahl von Polygonkanten, da durch die zweite Clip-Funktion die Abgrenzung wie bei den anderen Tests auch durch das Polygon des Bundeslandes, und nicht durch die NoData-Pixel (und damit das Grenzliniensymbol der DTK) definiert ist (siehe Abb. 21).

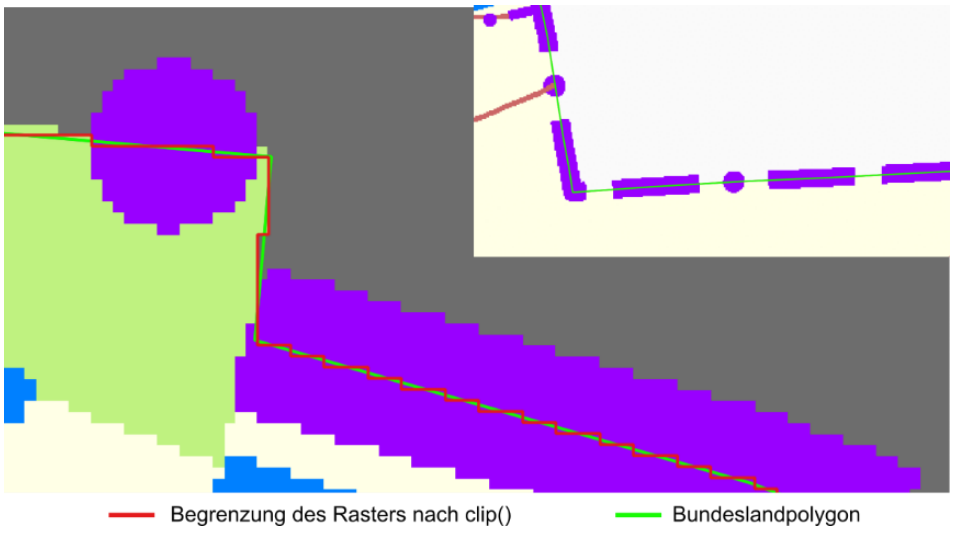


Abbildung 21: Begrenzung des VRT (Quelle: eigene)

Die Reduzierung der Auflösung der Raster mit *gdal_warp(…, 'Extra': -tr 0.1 0.1)* vor der Anwendung der Vektorisierung führt nicht zur Verringerung, sondern im Gegenteil zur weiteren Verlängerung der Laufzeit. Eine direkte Alternative für *polygonize()* könnte die in der GDAL-Version 3.8.0 veröffentlichte Funktion *gdal_footprint()* sein, welche jedoch noch nicht in der aktuellen QGIS-Version integriert ist. Eine weitere Option wäre die Anwendung der Funktion *raster2pgsql()* und *ST_DumpAsPolygons()* von PostGIS (BRINKHOFF 2022: 429, 436), was allerdings die Einbindung eines PostGIS-Servers in den Workflow erfordern würde. Auch eine Untersuchung und eventuelle Editierung des Pythonskripts **polygonize.py** (vgl. Anhang 26), welches der Funktion zugrunde liegt, könnte zur Lösung des Problems beitragen (ERICKSON et al. 2013: o.S.), was aber den Rahmen dieser Arbeit ebenfalls sprengen würde. Vor der Lösung des Problems durch den in Abschnitt 4.2.2 beschriebenen Workflow wurden stattdessen die folgenden Alternativen getestet:

Test 6: Anwendung von *polygonize()* auf einzelne Raster (***aktualisiere_Vertr_e.py)***
Hier werden die einzelnen Raster statt wie vorher das alle Eingangskacheln zusammenfassende VRT vektorisiert. Dazu wird das Skript so modifiziert, dass die Anwendung der clip-Funktion bei Schritt H des Workflows innerhalb einer Schleife erfolgt. In dieser wird über die Liste der einzelnen Raster iteriert. Jeweils im Anschluss an die clip-Funktion wird die Funktion *vrt2poly()* angewandt. Dadurch, dass der Datenausgang *polygon_footprint.gpkg* von *polygonize()* nicht variiert wird, werden die resultierenden Polygone dabei automatisch bei jedem Schleifendurchgang zu einer Gesamtgeometrie aufsummiert, so dass im Anschluss keine Merge-Funktion o.ä. notwendig ist. Diese Alternative führt bereits zu einer erheblichen Reduzierung der Laufzeit (14 Minuten statt 1:24 Stunden des ursprünglichen Skriptes und 18 Minuten bei ArcGIS).

Test 7: ohne Reduzierung auf Kartenblätter/Landesfläche (***aktualisiere_Vertr_o.py***)
Als zweite Alternative für den Fall, dass bekannt ist, ob die Eingangsdaten auf die Landesfläche reduziert sind und welcher Wert für die NoData-Bereiche gesetzt ist, werden die ersten beiden clip-Funktionen in Schritt E und H ausgelassen. Dieser Workaround bietet sich an, da er zu der von allen Varianten kürzesten Laufzeit führt und genau auf die Eingangsdaten, welche die besonders langen Laufzeiten verursachen, beschränkt ist. Anstelle von Schritt E wird der Nodata-Wert der VRT durch *gdal_translate()* für die transparente Darstellung der entsprechenden Bereiche mit null definiert (vgl. Listing 12, Z. 1-3).
In Fällen, bei denen der NoData-Wert beim Farblayer wie bei den Daten aus Rheinland-Pfalz eins statt null ist könnten die Werte beispielsweise mit der QGIS-nativen Funktion *rastercalculator()* und dem Ausdruck *if ('raster@1'==0,1), if ('raster@1'==1,0)* getauscht werden. Anders als der bei D angewandte *gdal_rastercalculator()* unterstützt die native

QGIS-Funktion als PyQGIS-Befehl und als GUI-Version der *Processing Toolbox* (allerdings nicht die über die Werkzeugleiste im Menu Raster abrufbare) solche expliziten if-Ausdrücke.

```
processing.runAndLoadResults('gdal:translate', {'INPUT': pfad,
                                        'OUTPUT': r"Pfad/transl_%s_%s.vrt"%(dtk_nr,layer),
                                        'EXTRA': '-b 1','NODATA': 0})
raster = QgsRasterLayer(r"Pfad/translate_%s_%s.vrt"%(dtk_nr, layer), f"vrt_{dtk_nr}")
QgsProject.instance().addMapLayer(raster)
processing.run("gdal:rastercalculator",{'INPUT_A':r"Pfad/transl_%s_%s.vrt"%(proj,dtk_nr,layer
                                         'BAND_A':1,'FORMULA':'(A!=0)','NO_DATA':0, 'RTYPE':5,
                          'OPTIONS':'','EXTRA':'', 'OUTPUT':r"Pfad/calc_%s.tif"%(proj, dtk_nr)})
processing.runAndLoadResults('gdal:polygonize', {'INPUT':  r"Pfad/calc_%s.tif"%dtk_nr,
                                                 'OUTPUT': r"Pfad/p_%s.shp"%dtk_nr,
                                               'BAND':1,'FIELD':'DN','EIGHT_CONNECTEDNESS':False })
poly_pfade.append(r"Pfad/p_%s.shp"%dtk_nr)
```

Listing 12: Übergabe des Rasters an polygonize() ohne clip-Funktionen

Nach der Definition des NoData-Wertes wird beim Farblayer der GDAL-*Rastercalculator* mit dem impliziten Ausdruck *(A!=0)* angewandt. Damit wird die erste translate-Funktion mit dem Parameter *mask band* in ***vrt2poly.py*** ersetzt, da diese ohne vorherige Anwendung der clip-Funktionen keine Auswirkung hat. Alle Pixel mit Werten ungleich null werden also auf eins gesetzt. Im Anschluss wird *gdal_polygonize()* angewendet, allerdings wie bei der ersten Alternative auf die einzelnen Raster, da all diese Schritte direkt in der Schleife von Schritt E und D erfolgen. Bei dieser Variante muss für die Anwendung von *gdal_rastercalculator()* eine TIFF-Datei zwischengespeichert werden, da diese Funktion (ebenso wie die QGIS-native) das VRT-Format nicht als Datenausgang unterstützt. In diesem Fall wird die GDAL-Version genutzt, weil bei dieser eine variable Definition des Eingangsbandnamens im Berechnungsausdrucks möglich ist *('BAND_A': 1)*. Bei der nativen Funktion ist das Eingangsband z. B. mit *('raster@1'==0,1)* fest, und kann auch nicht durch f-Strings oder ähnliche Zeichenkettenfunktionen für die Iteration durch die einzelnen Raster variiert werden.

Die weitere Verkürzung der Laufzeit von etwa 12 Minuten bei dieser Variante im Vergleich zu Test 6 zeigt, dass die lange Berechnungszeit von *polygonize()* nicht nur mit der Anwendung auf mehr als eine Kachel, sondern auch mit der Anwendung der Funktionen *clipbymasklayer()* und/oder *translate()* mit der Option *mask band* auf die VRT zusammenhängt. Da diese Version die Bekanntheit der Dateneigenschaften voraussetzt wurde trotz ihrer kurzen Laufzeit zusätzlich die dritte Variante von Test 8 entwickelt.

Da bei den Tests 5 und 9 auch bei ArcGIS der bereits beschriebene, zur Anzahl der Kacheln disproportionale Anstieg der Laufzeit besteht, wurde auch hier ein alternativer Workflow für bereits auf die Landesfläche beschränkte Raster getestet. Dabei können die Schritte *Import Geometry* (IG) und *ClipFootprints* im ersten MDCS-Prozess übersprungen werden. Dazu

müssen sie aus der Befehlskette in der Konfigurationsdatei entfernt werden (vgl. Listing 13, Z. 1).

```
1 <Command>CM+SP+DN+AR+CV+CV+BB+EMDG</Command>
2   <DefineMosaicDatasetNoData>
3      <num_bands>1</num_bands>
4      <bands_for_nodata_value>0</bands_for_nodata_value>
5   </DefineMosaicDatasetNoData>
6   <ExportMosaicDatasetGeometry>
7      <out_feature_class>[Pfad]\source_footprints</out_feature_class>
8      <geometry_type>FOOTPRINT</geometry_type>
9   </ExportMosaicDatasetGeometry>
```

Listing 13: abgeänderte Konfigurationsdatei

Die Befehle *Defined NoData* (DN), *Export Mosaic Dataset Geometry* (EMDG) und ihre Parameter werden hinzugefügt, wodurch die NoData-Bereiche der Eingangsraster transparent dargestellt und, wie bei der in der Folge beschriebenen QGIS-Variante von Test 8, die rechteckigen Rasterausmaße für die Ermittlung der sich überschneidenden Bestandsraster exportiert werden (anstelle der zuvor durch *ClipFootprints* exportierten Vektordatei *polygon_footprints.shp*). Streng genommen ist diese Workflowvariante daher eher eine Kombination der QGIS-Varianten von Test 7 und 8. Da sich bei diesem Test die Anzahl der getauschten Kacheln aber ohnehin nicht ändert, werden sie in Tabelle 1 so gegenübergestellt. Die Berechnungszeit wird auf diese Weise im Vergleich zum Skript mit Reduktion der Geometrien halbiert. Der in Abschnitt 3.1 erwähnte, angenommene Performanzverlust durch die Nutzung von Transparenzen tritt hier also offenbar nicht ein oder ist kleiner als der durch die clip-Funktion verursachte.

Eine Veränderung des ArcGIS-Workflows zur Laufzeitverkürzung für alle Eingangsdaten ähnlich zu dem in 4.2.2 beschriebenen QGIS-Skript ist nicht möglich, da die Anwendung von *ClipFootprint*, bei welcher das Polygon exportiert wird, in erster Linie der notwendigen Reduktion der eigentlichen Raster dient. Die MDCS-Log-Dateien zeigen aber auch, dass der Großteil der Berechnungszeit des MDCS-Prozesses für die Quell-MDs ohnehin nicht durch die Funktion *ClipFootprints*, sondern eher für *AddRasters* und *SetProperties* aufgewendet wird (vgl. Tabelle 2 und Anhang 10 und 11).

Tabelle 2: Laufzeiten einzelner MDCS-Prozesse

Prozess	**Laufzeit** (Sekunden)	
	Test 4	**Test 9**
gesamt	43	71
Create MD	6	5
SetProperties	11	12
AddRasters	19	34
CalculateValues	4	10
ImportGeometry	1	5
ClipFootprint	3	7
BuildBoundary	1	6

Test 8: *tileindex()* statt *polygonize()* ***(aktualisiere_Vertr_t.py)***

Bei dieser Version des QGIS – Skriptes wird die Vektorisierung in ***vrt2poly.py*** durch die bereits in 4.2.1 erwähnte Funktion *tileindex()* ersetzt. Mit dieser werden Polygone der rechteckigen Gesamtausmaße der Raster erzeugt. Dadurch werden unter Umständen mehr Kacheln als eigentlich notwendig ausgetauscht (bei diesem Test bleibt es bei 8 Kacheln pro Layer). Da die Zeit für die Berechnung einer Kachel aber sehr klein ist, müsste die Anzahl umsonst selektierter Kacheln sehr groß sein um eine ähnlich lange Berechnungszeit wie die der Vektorisierungsfunktion mit mehreren Kacheln zu verursachen. Die Funktion *tileindex()* wird nach Schritt D auf die einzelnen Raster des Farblayers angewandt. Die Pfade der resultierenden Polygone werden wie bei der finalen Variante in 4.2.2 in einer Liste gespeichert, anhand der sie nach Abschluss der Schleife vor Schritt F mit *mergevectorlayers()* zusammengefügt und mit *dissolve()* zu einem einzigen Feature verbunden werden. Die erhaltene Geometrie wird dann bei I in der clip-Funktion genutzt. Diese Variante wäre der aus Test 8 trotz der durch die Clip-Funktionen verursachten, geringfügig längeren Laufzeit wegen ihrer Anwendbarkeit auf unbekannte Daten vorzuziehen.

5.4 Résumé

Die Bereitstellung der Geodaten durch eine öffentliche, im Dienste des Gemeinwohls stehenden und nicht von der Erwirtschaftung von Gewinnen abhängigen Institution hat einen großen gemeinen Nutzen. Die Geodaten bilden eine wichtige Grundlage für verschiedenste private und öffentliche Akteure. Der Zweck und die Zielsetzungen staatlicher Institutionen ähneln in vielerlei Hinsicht denen der Akteure und Organisationen der Open Data-Bewegung. Der Unterschied zu einer Organisation wie OSGeo besteht lediglich in der finanziellen und strukturellen Haftung an den Staat. Gegenüber der eher global und international ausgerichteten Open Data-Philosophie folgt daraus eine gewisse regionale und demografische Einschränkung, sowie ggf. eine Orientierung an den politischen Zielen der Regierung. Durch die starken Überschneidungen steht aber eigentlich außer Frage, ob eine

solche öffentliche Einrichtung Anstrengungen unternehmen sollte, um ihre Dienstleistungen für den öffentlichen Nutzen mit Hilfe von technischen Lösungen aus dem Open Source-Bereich zur Verfügung zu stellen. Wie in Abschnitt 3.3 erläutert, scheint dies beim BKG und den übergeordneten Regierungsinstitutionen auch bereits das gesetzte Ziel zu sein.

Das allgemeine Interesse an der Bereitstellung von Geodaten, und die stetige Verbesserung und der Ausbau dieser Dienstleistungen steht also außer Frage. Sie ist aber in den wenigsten Fällen von „dringlichem“ Charakter, weder durch den Bedarf und die Art der Nutzung selbst, noch durch das betriebswirtschaftliche Modell des BKG. Daher gibt es kaum ein Argument dafür, nicht in die Erforschung der Möglichkeiten und den Erwerb von technischen Kompetenzen zu investieren, auch wenn dies auf kürzere Sicht zunächst mehr Zeitaufwand und Kosten bedeuten kann. Diese Kosten müssen immer im Verhältnis zum allgemeinen volkswirtschaftlichen oder gesellschaftlichen Nutzen stehen und auch ökologisch vertretbar sein. Da der Nutzen nicht unendlich groß sein kann, rechtfertigt er auch nicht unlimitierte Kosten. Es sollte also versucht werden, diese wirtschaftlich, kulturell und wissenschaftlich nützlichen Dienstleistungen mit möglichst geringen laufenden Kosten und Verbrauch von Ressourcen zur Verfügung zu stellen. Die Lizenzkosten von proprietärer Software sind damit von vornherein ein schwerwiegendes Argument gegen ihre Nutzung. Dazu kommt noch der lange, eigentlich „ewige“ Zeithorizont eines staatlichen Amtes als Argument für die Nutzung von Open Source-Anwendungen.
Dies bedeutet allerdings nicht, dass die betriebswirtschaftliche Organisationsform und Zielsetzungen privatwirtschaftlicher Unternehmen von Softwareherstellern wie Esri nicht andere Vorteile mit sich bringen können. Diese könnten z. B. ein größeres Potenzial für Innnovation und wirtschaftliche Dynamik im Sinne der schnellen Reaktion auf akute Entwicklungen des Marktes sein, welche für viele andere privatwirtschaftliche Akteure und damit auch für die Allgemeinheit von Interesse sein kann. Wie in Abschnitt 2.3 erwähnt, ähnelt QGIS ArcGIS in vielen Aspekten. Solch eine Nachbildung bzw. starke Anlehnung an die Funktionen und Architektur der marktführenden, proprietären Anwendung existiert bei vielen Open Source-Projekten (z. B. Microsoft Office – Libre Office, IBM SPSS – PSPP, Adobe PhotoShop – Gimp). Dies bestätigt eine möglicherweise wichtige, pionierartige Rolle privatwirtschaftlicher Software-Unternehmen für Innovation und Erforschung neuer technischer Möglichkeiten und Tendenzen. Auf der anderen Seite wiederum existieren auch viele innovative Projekte, die von Beginn an der Open Source-Philosophie folgen (vgl. Abschnitt 3.3).

Unabhängig von dieser Ambiguität sollte eine Institution wie das BKG die Dienstleistungen und Produkte einer privatwirtschaftlich und gewinnorientiert agierenden Organisation erst dann nutzen, wenn Open Source-Lösungen sich als nicht zielführend, impraktikabel oder

sehr unwirtschaftlich erweisen. Wenn die Bindung an ein Unternehmen und damit eine langfristige und umfassende Abhängigkeit erwogen wird, sollten neben den wirtschaftlichen auch rechtliche und politische Aspekte berücksichtigt werden. So handelt es sich z. B. im Falle von Esri um ein ausländisches Unternehmen, welches dementsprechend einer anderen Gesetzgebung unterliegt und nach anderen politischen Zielsetzungen handeln muss, auf welche die hiesige Bevölkerung als primäre Zielgruppe des BKG keinerlei demokratischen Einfluss hat. Bei umfassender Nutzung eines Systems wie ArcGIS Enterprise (vgl. Abschnitt 7.2) kann nicht abschließend sichergestellt werden, dass Daten nicht ungewollt in die Hände Dritter gelangen. Das gilt bei einer proprietären Software wie ArcGIS aufgrund des nicht zugänglichen Quellcodes nicht nur für SaaS-Plattformen wie ArcGIS Online, sondern auch bei der Nutzung eigener Hardware-Infrastruktur. In Bezug auf die Gesetzgebung der USA sollte dabei der Patriot Act (2001), durch welchen US-Behörden ohne richterliche Anordnung den Zugriff auf US-Server erhalten, und der Cloud-Act (2018) erwähnt werden. Letzterer verpflichtet US-amerikanische Firmen den US-Behörden auch dann Zugriff auf Daten zu gewährleisten, wenn deren Speicherung nicht auf Servern in den USA erfolgt (FISCHER et al. 2023: 65 f.).

Grundvoraussetzungen für die Entscheidung für die eine oder andere Software sind aber auch geringe Kosten für Energie, Infrastruktur und Hardware auf längerfristige Sicht, sowie Performanz und Stabilität. Bei all diesen Planungen und Investitionsentscheidungen müssen der schnelle Rhythmus der technologischen Entwicklungen und Aspekte der Interoperabilität mit einbezogen werden, wobei Letztere tendenziell eher für Open Source als für proprietäre Software-Lösungen sprechen.
Die Nutzung solcher Technologien zur Vermeidung von Abhängigkeiten kann aber nur dann wirtschaftlich sein, wenn die Sicherung und Weitergabe von etabliertem Wissen und Kompetenzen auf effiziente Weise stattfindet. Dies ist von fundamentaler Bedeutung, da die zeitlichen und finanziellen Kosten für den Erwerb dieses Knowhows signifikant sein können. Eine gute Dokumentation, bzw. eine Strategie für die didaktische Aufbereitung und Weitergabe dieses Wissens ist deshalb entscheidend. Diese kann verschiedenste Formen annehmen, wie z. B. die einer beim BKG bereits vorhandenen Wiki-Plattform mit Texten, Übersichten und Videoanleitungen.

Die Möglichkeit, den Support als bezahlte Dienstleistungen nutzen zu können, ist einer der bedeutenden Vorteile proprietärer Software. Damit profitiert der Software-Nutzer von der Expertise des Herstellers selbst. Allerdings müssen die Problemstellungen für die Nutzung des Supports bei Esri zunächst sorgfältig dokumentiert, isoliert und gezielt formuliert werden. Die Nutzung von Funktionalitäten wie MDCS und die Veröffentlichung von *Web Tools* (vgl. Abschnitt 7.2) erfordert trotz Support und relativ guter Dokumentation einen beträchtlichen

Einarbeitungsaufwand, der von dem für die Entwicklung des QGIS-Prozesses nicht überstiegen wurde. Dazu kommt noch die Möglichkeit des Erwerbs bezahlter Support-Services für Open Source-Produkte. Auch die Bedeutung von Internet-Foren wie Stackexchange GIS, in denen Problemstellungen durch eine globale, aktive Mitgliedergemeinschaft erörtert und so oft schneller als durch den Support gelöst werden, sollte an dieser Stelle genannt werden.
Der Anwender muss sich sowohl bei ArcGIS als auch bei QGIS in das Vokabular und die Konzepte der Software einarbeiten und ein Grundverständnis der Syntax und Methoden ihrer Programmierschnittstellen erwerben. Die Vorteile der Nutzung des proprietären ArcGIS trotz Vorhandensein der Lösungen mit QGIS sind für diese Anwendung deshalb nicht ersichtlich.

Das BKG sollte seine wirtschaftliche Unabhängigkeit und seinen langen Zeithorizont als staatliche Institution dazu nutzen, Dienstleistungen mit unabhängigen Open Source-Technologien anzubieten, sofern diese zu den gewünschten Ergebnissen führen. Das ist im Falle des VRT-Formats für die Referenzierung von Rasterdaten gegeben, mit welchem der untersuchte Prozess der Aktualisierung der DTK-Daten deutlich schneller als mit den *Mosaic Datasets* von ArcGIS durchgeführt werden kann.

7. Ausblick: Möglichkeiten mit Web-GIS

Der nächste Schritt für eine weitere Automatisierung des Gesamtprozesses, welcher im Rahmen dieser Arbeit nicht abschließend untersucht wird, wäre seine Ausführung als Geodienst auf einer dedizierten Maschine. Dies bringt die Vorteile des zentralen Zugriffs für Nutzung und Pflege, die Befreiung von Rechenkapazitäten und die Möglichkeit der Flexibilisierung und Steuerung der Rechenkapazität durch Ausführung auf ggf. mehreren virtuellen oder physischen Maschinen. Außerdem wäre ein automatischer Anstoß des Prozesses bei Ankunft neuer Daten vorstellbar.

7.1 Geodienste

Geodienste sind raumbezogene Webdienste, mit denen Karten, Geodaten und Funktionalitäten über das Internet übertragen und bereitgestellt werden können. Die meiste Anwendung finden die standardisierten Geodienste der OGC: Der *Web Map Service* (WMS) dient ausschließlich der Bereitstellung von raster- und vektorbasierten Karten. Mit dem *Web Feature Service* für Vektordaten und dem *Web Coverage Service* (WCS) für Rasterdaten können diese auch editiert werden (BRINKHOFF 2022: 17). WCS und WMS werden am BKG bereits für die Bereitstellung der Vertriebsdaten von DTK und DOP genutzt. Für die hier gebrauchte serverseitige Ausführung der Workflows mit Geoverarbeitungsfunktionen auf Grundlage von sich auf anderen Maschinen oder dem ausführenden Server selbst befindlichen Geodaten kann der *Web Processing Service* (WPS) genutzt werden (SEIP et al. 2017: 370). Bei diesem kann der Client drei Funktionen aufrufen, welche denen der anderen OGC-Webdienste ähneln: *GetCapabilities* liefert eine Beschreibung der Fähigkeiten des Dienstes, *DescribeProcess* eine detaillierte Prozessbeschreibung inklusive In- und Outputparameter. Durch *Execute* wird der Prozess angestoßen (LANGE 2013: 250). Für die Ausführung dieser Geodienste können die quelloffenen Server MapServer, GeoServer und QGIS Server, alle Projekte der OSGeo, oder der proprietäre ArcGIS (Image) Server von Esri genutzt werden.

7.2 Veröffentlichung als Web Tool mit ArcGIS

Esri bietet neben dem WPS auch einen eigenen Verarbeitungsdienst an, der je nach Lizensierungsform als *Geoprocessing Service* oder *Web Tool* bezeichnet wird. Dieser wird auf einer ArcGIS Server-Instanz veröffentlicht. Dieser stellt neben dem *Datastore* und den Webadaptoren eine der wichtigsten Komponenten der ArcGIS Enterprise Suite dar (vgl. Abb. 22).

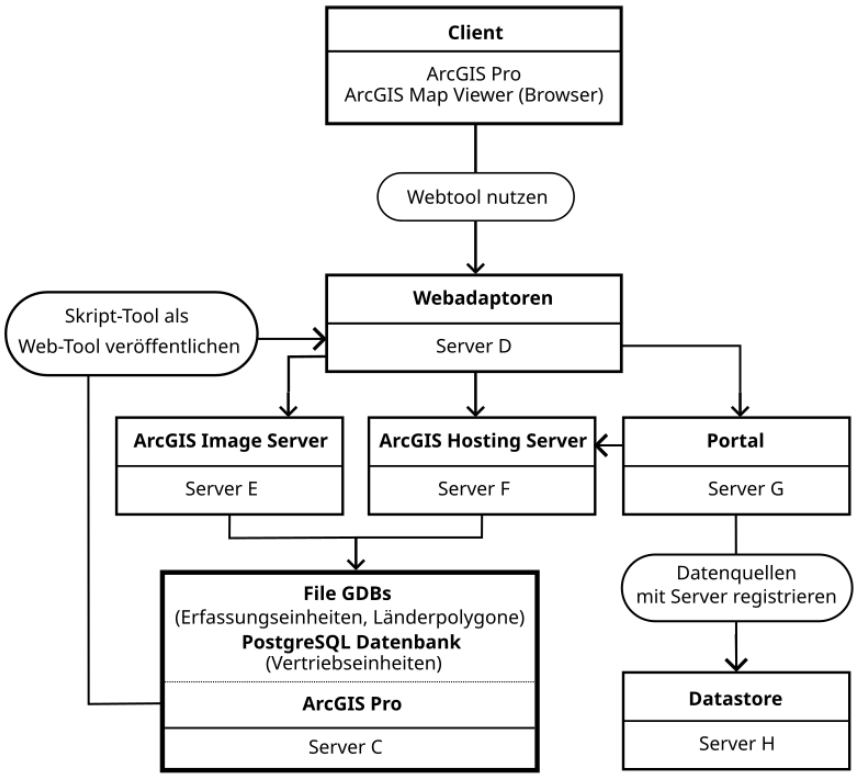


Abbildung 22: ArcGIS Enterprise Architektur (Quelle: eigene)

Nach Veröffentlichung des Geodienstes kann das Skript dann von einem Client wie z. B. wiederum ArcGIS Pro oder einem Browser durch den ArcGIS Map Viewer oder eine *Web App* ausgeführt werden. Die Datenquellen werden mit der Server-Software registriert, also in der Datastore-Komponente referenziert. Das ursprüngliche Skript wird bei der Veröffentlichung automatisch editiert und im *FileShare*-Verzeichnis von ArcGIS Server abgelegt. Dabei werden am Anfang des Skriptes zusätzlich benötigte Bibliotheken importiert (vgl. Listing 14, Z. 1 u. 2) und anschließend neue Variablen definiert, welche bestimmte Zeichenketten von Dateipfaden mit teilweise leicht geänderter Syntax ersetzen (Z. 4-8). Dies kann zu Fehlern bei der Ausführung des Dienstes führen, was aber teilweise durch manuelle Editierung des veröffentlichten Skriptes vermieden werden kann.

```
1  # Esri start of added imports
2  import sys, os, arcpy
3  # Esri end of added imports
4  # Esri start of added variables
5  g_ESRI_variable_1 = '\\mdcs-dtk\\Parameter\\Config\\DTK_Source.xml'
6  g_ESRI_variable_2 = '\\mdcs-dtk\\Parameter\\Config\\DTK_Derived2.xml'
7  g_ESRI_variable_3 = '%scratchFolder%\\f"{layer}_{dtk_nummer_str[1:-1]}"'
8  # Esri end of added variables
```

Listing 14: Ersatz von Zeichenketten durch Variablen

Die Palette der lokal in ArcGIS Pro ausführbaren Funktionen und die Anforderungen an die Syntax in ArcPy (z. B. von Dateipfaden) für eine fehlerfreie Ausführung überdecken sich

nicht vollständig mit denen des *Geoprocessing Service*. Aufgrund dieser spezifischen Anforderungen konnte die Möglichkeit der Veröffentlichung des Workflows als *Web Tool* im Rahmen dieser Arbeit nicht abschließend erörtert werden. Eine mögliche Lösung wäre der komplette Ersatz der zwei MDCS-Vorgänge im Workflow durch ArcPy-Befehle, auch wenn die Veröffentlichung eines Skriptes mit MDCS-Vorgängen laut der Dokumentation von Esri und zufolge erfolgreich durchgeführter Tests generell möglich ist. Solch ein Ersatz der MDCS-Vorgänge könnte auch Vorteile in Bezug auf eventuell besser interpretierbare Fehlerausgaben bringen, weil in den Log-Dateien der MDCS keine Fehler des Geodienstes registriert werden.

7.3 Ausführung des QGIS-Workflows als WPS

Der in PHP beschriebene, quelloffene *Lizmap Web Client* des französischen Unternehmens 3Liz ermöglicht die direkte Exponierung von Layern aus QGIS-Projekten als OGC-konforme Geodienste (WMS und WFS). Mit dem erweiternden Modul *Lizmap WPS Web Client* können auch Workflows mit Geoverarbeitungsprozessen als WPS serverseitig ausgeführt werden. Dieses Modul wurde auf Grundlage der WPS-Implementation *PyWPS* entwickelt, welche wiederum Teil eines durch die Gesellschaft für Datenanalyse und Fernerkundung (GDF) Hannover finanzierten Projektes von Jachym Cepicky zur Integration von GRASS GIS und MapServer war. Aktuell ist *PyQGIS* auch ein OSGeo-Projekt. Der *Lizmap Web Client* wird diesem aufgrund seiner direkten Verbindung mit QGIS vorgezogen.

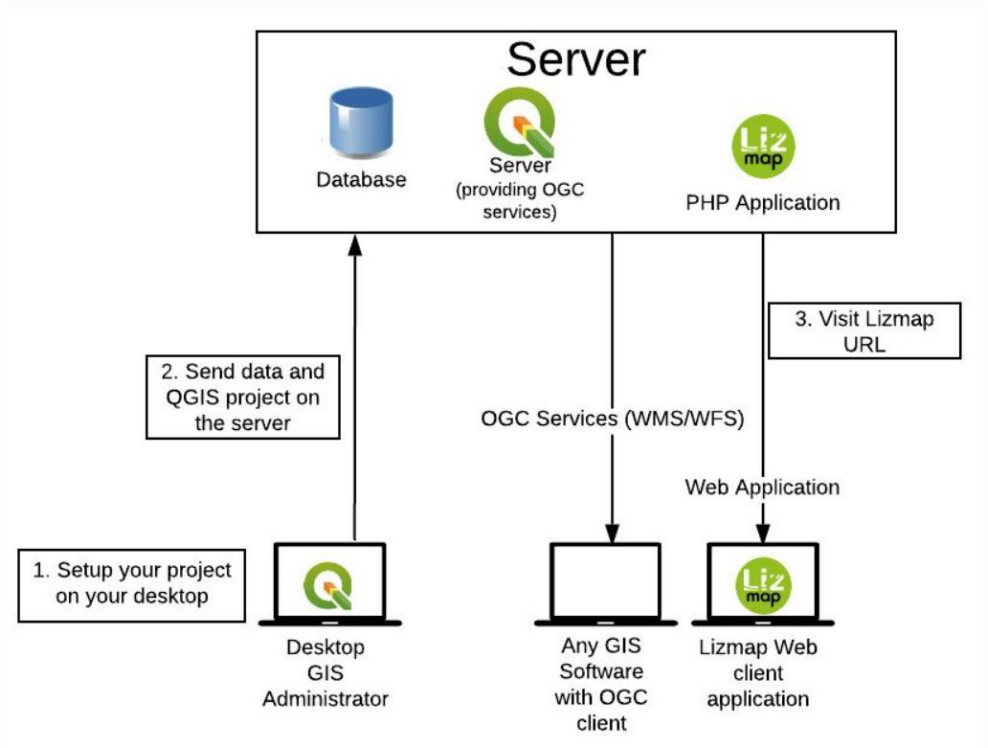


Abbildung 23: Lizmap-Architekur (DOUCHIN, 3Liz (a) 2023: o.S.)

Abbildung 23 zeigt die allgemeine Architektur der Lizmap-Anwendung. Für den Test wird Ubuntu Server 22.04.02 als eine mit *Oracle Virtual Box* erzeugte virtuelle Maschine verwendet, da Lizmap nur für Linux-Systeme verfügbar ist. Auf dieser werden der *Apache2* oder *Nginx HTTP Server* und QGIS Server installiert. Für QGIS Server wird der Plugin *Lizmap Server* installiert. Auch QGIS Desktop wird für die Verbindung mit QGIS Server und dem Web Client nun im Betriebssystem Ubuntu ausgeführt, für welches die Version 3.10.4 zur Verfügung steht, während für Windows die Version 3.32.0 verwendet wurde. Deshalb muss das Skript an einigen Stellen leicht modifiziert werden. So wird z. B. die nicht unterstützte Funktion *savefeatures()* durch *writeAsVectorFormat()* ersetzt (vgl. Listing 15) und die Parameter *featureLimit* und *geometryCheck* aus *QgsProcessingFeatureSourceDefinition()* in den Clip-Funktionen entfernt. Dafür kann die im Verlauf der ausgeführten GUI-Befehle angezeigte Syntax übernommen werden.

```
    processing.run("native:savefeatures",{'INPUT':QgsProcessingFeatureSourceDefinition(
        "erfassungskacheln%s"%massstab, selectedFeaturesOnly=True, featureLimit=-1,
        geometryCheck=QgsFeatureRequest.GeometryAbortOnInvalid), 'OUTPUT': datei,
        'LAYER_NAME': '', 'DATASOURCE_OPTIONS': '', 'LAYER_OPTIONS': ''})

    QgsVectorFileWriter.writeAsVectorFormat(
                              erfassungseinheiten, datei, "utf-8", onlySelected = True)
```

Listing 15: Ersatz von savefeatures()

Anders als bei der Erstellung der Workflows für die Desktop-Anwendung, bei denen der Arbeitsaufwand und die Komplexität in ArcGIS und QGIS vergleichbar sind, tritt hier eher der Fall ein, dass die Realisierung mit QGIS zunächst einen Mehraufwand erfordert, da alle System-Komponenten unabhängig voneinander installiert werden müssen. 3Liz macht in der Dokumentation zu *Lizmap* deutlich, dass für die Installation des Web Clients Kenntnisse der Systemadministration wie die Installation eines Web Servers (Nginx, Apache), eines LAMP-ähnlichen Software-Stacks (Linux, Apache, MySQL, PHP), Lesen und Setzen von Umgebungsvariablen, Verständnis für das Starten von Prozessen auf einem Server, Lesen von Logs von *stderr* (standard error, neben *stdin* und *stdout* einer der drei Standard-Datenströme in Unix-verwandten Betriebssystemen) für QGIS Server, die Fehlerbehebung von HTTP-Anfragen mit cURL (Client for URL, Programmierbibliothek und Kommandozeilen-Programm) und die Verwaltung einer PostgreSQL Datenbank vorausgesetzt werden. Die Speicherung der Daten in der PostgreSQL Datenbank ist notwendig, wenn Layer wie hier editiert werden sollen. Außerdem kann sie für Suchfunktionen und die Speicherung von Daten und Aktionen der Nutzer des Web-Clients genutzt werden (DOUCHIN, 3LIZ (b) 2023: o.S.).

Anhang (digital)

Nr.	Verzeichnis
1	ArcGIS/init.py
2	ArcGIS/aktualisiere_Raster.py
3	ArcGIS/mdcs-dtk/scripts/MDCS_UC.py
4	ArcGIS/DTK_Nummern_Bestand.py
5	ArcGIS/clip.py
6	ArcGIS/mdcs-dtk/Parameter/config/DTK_Source.xml
7	ArcGIS/mdcs-dtk/Parameter/config/DTK_Derived.xml
8	ArcGIS/mdcs-dtk/Parameter/config/DOP_Source.xml
9	ArcGIS/mdcs-dtk/Parameter/config/DOP_Derived.xml
10	ArcGIS/mdcs-dtk/Logs/DTK_Source_20230720T131340.xml
11	ArcGIS/mdcs-dtk/Logs/DTK_Source_20230722T071642.xml
12	QGIS/init.py
13	QGIS/aktualisiere_Vertr.py
14	QGIS/aktualisiere_Vertr_e.py
15	QGIS/aktualisiere_Vertr_o.py
16	QGIS/aktualisiere_Vertr_t.py
17	QGIS/aktualisiere_Vertr_v.py
18	QGIS/Dateien_loeschen.py
19	QGIS/lade_erfassungseinheiten.py
20	QGIS/bundeslandpoly_selekt.py
21	QGIS/rel_besKacheln.py
22	QGIS/vrt2poly.py
23	QGIS/pixel_tauschen.py
24	QGIS/clip.py
25	QGIS/Zwischenschritte/Raster/neu/r_clip2x_col.vrt
26	QGIS/polygonize.py

Literaturverzeichnis

ADOBE INC. (2023): TIFF files. https://www.adobe.com/creativecloud/file-types/image/raster/tiff-file.html (Abgerufen am 01. August 2023).

BILL, R. (2016): Grundlagen der Geo-Informationssysteme. (Wichmann) Berlin, Offenbach am Main.

BOVIER, R. (2023): Managing and distributing orthophotos in Switzerland: Webinar Orthophoto management and distribution, 19.04.2023. (Bundesamt für Landestopografie swisstopo) Wabern.

BKG (a) (2023): Topographische Karten. https://www.bkg.bund.de/DE/Das-BKG/Wir-ueber-uns/Geoinformation/Produktion/Topographische-Karten/topographische-karten.html (Abgerufen am 30. Juli 2023).

BKG (b) (2023): Produktion. https://www.bkg.bund.de/DE/Das-BKG/Wir-ueber-uns/Geoinformation/Produktion/produktion.html (Abgerufen am 31. Juli 2023).

BKG (c) (2023): Kartographische Informationsgewinnung. https://www.bkg.bund.de/DE/Das-BKG/Wir-ueber-uns/Geoinformation/Produktion/Informationsgewinnung/informationsgewinnung.html (Abgerufen am 31. Juli 2023).

BKG (d) (2023): Dokumentation, Digitale Topografische Karte 1:25 000. https://sg.geodatenzentrum.de/web_public/gdz/dokumentation/deu/dtk25.pdf (Abgerufen am 10. Juli 2023).

BKG (e) (2023): Digitale Orthophotos und Satellitenbilddaten. https://gdz.bkg.bund.de/index.php/default/webdienste/digitale-orthophotos.html (Abgerufen am 30. Juli 2023).

BKG (f) (2023): Kartenviewer des Geodatenzentrums (GeoBasis-DE, Geoportal.de). http://sg.geodatenzentrum.de/web_bkg_webmap/applications/info/dop.html (Abgerufen am 30. Juli 2023).

BKG (g) (2023): Echtzeit-GNSS, GNSS Standardisierung. https://www.bkg.bund.de/DE/Das-BKG/Wir-ueber-uns/Geodaesie/Satellitennavigation/Echtzeit-GNSS/ntrip_cont.html (Abgerufen am 30. Juli 2023).

BKG (h) (2017): SmartMapping. *https://www.bkg.bund.de/DE/Forschung/Projekte/Smart-Mapping/Smart-Mapping_cont.html* (Abgerufen am 30. Juli 2023).

BKG (i) (2022): Gewusst Wo! Geoportal.de - neues Release, neue Features. *https://www.bkg.bund.de/SharedDocs/Downloads/BKG/DE/Downloads-Veranstaltungen/Gewusst-wo-2022/Gewusst-Wo-2022-Geoportal-DE.pdf?__blob=publicationFile&v=2* (Abgerufen am 30. Juli 2023).

BKG (j) (2016): Fachkoordination Copernicus-Dienst Landüberwachung. https://www.bkg.bund.de/DE/Fernerkundungsdienste/Copernicus/Fachkoordination/fachkoordination_cont.html (Abgerufen am 30. Juli 2023).

BKG (k) (2023): Produktkatalog, Unsere Open-Data Produkte. https://www.bkg.bund.de/SharedDocs/Downloads/BKG/DE/Downloads-DE-Flyer/BKG-Produktkatalog-Open-Data-DE.pdf?__blob=publicationFile&v=40 (Abgerufen am 30. Juli 2023).

BRINKHOFF, B. (2022): Geodatenbanksysteme in Theorie und Praxis. (Wichmann) Berlin, Offenbach am Main.

BUNDESMINISTERIUM DES INNERN UND FÜR HEIMAT (2021): Kabinett beschließt Open-Data-Strategie der Bundesregierung. https://www.bmi.bund.de/SharedDocs/pressemitteilungen/DE/2021/07/open-data-strategie-der-bundesregierung.html (Abgerufen am 01. August 2023).

CAVALLINI, P. u. LAMI, L., GEOSPATIAL WORLD (2010): Free GIS desktop and analyses: QuantumGIS, the easy way. https://www.geospatialworld.net/article/free-gis-desktop-and-analyses-quantumgis-the-easy-way/ (Abgerufen am 30. Juli 2023).

DIEDERICH, C. (2016): Implementierung eines Geodatenportals mit Mapbender und freier Software für Geoinformationssysteme: Bachelorarbeit. https://digibib.hs-nb.de/file/dbhsnb_derivate_0000002009/Bachelorarbeit-Diederich-2016.pdf (Abgerufen am 15. Juni 2023).

DOUCHIN, M., 3LIZ (a) (2023): Lizmap Documentation, Introduction, Lizmap Architecture. https://docs.lizmap.com/current/en/introduction.html#lizmap-architecture (Abgerufen am 08. August 2023).

DOUCHIN, M., 3LIZ (b) (2023): Lizmap Documentation, Installing and upgrading Lizmap, Server administration knowledge. https://docs.lizmap.com/current/en/install/pre_requirements.html (Abgerufen am 08. August 2023).

ELLING, R., WEINER, A. (2022): Erreichbarkeitsanalysen für Deutschland, Gitterzellendatenbank. *https://www.bkg.bund.de/SharedDocs/Downloads/BKG/DE/Downloads-Veranstaltungen/Gewusst-wo-2022/Gewusst-Wo-2022-Gitterzellendatenbank.pdf?__blob=publicationFile&v=2* (Abgerufen am 30. Juli 2023).

ERICKSON, J., DANIEL, C., PAYNE, M. (2013): Raster Layers, Polygonize a Raster Band: Python GDAL/OGR Cookbook 1.0 documentation. http://pcjericks.github.io/py-gdalogr-cookbook/raster_layers.html (Abgerufen am 15. Juli 2023).

ESRI (a) (2015): Independent Report Highlights Esri as Leader in Global GIS Market https://www.esri.com/about/newsroom/announcements/independent-report-highlights-esri-as-leader-in-global-gis-market/. (Abgerufen am 08. Juli 2023).

ESRI (b) (2022): ArcGIS Pro Python reference. https://pro.arcgis.com/en/pro-app/latest/arcpy/main/arcgis-pro-arcpy-reference.htm (Abgerufen am 10. Juli 2023).

ESRI (c) (2022): Mosaic Dataset Configuration Script Usage Documentation. https://github.com/Esri/mdcs-py/blob/master/Documentation/mdcs.pdf (Abgerufen am 10. Juli 2023).

ESRI (d) (2023): Faktor für parallele Verarbeitung (Umgebungseinstellung). https://pro.arcgis.com/de/pro-app/latest/tool-reference/environment-settings/parallel-processing-factor.htm (Abgerufen am 12. Juli 2023).

ESRI (e) (2023): Dateien, Tabellen und Web-Services-Raster-Typen. https://pro.arcgis.com/de/pro-app/latest/help/data/imagery/files-tables-and-web-raster-types.htm (Abgerufen am 12. Juli 2023).

FISCHER, S., FROHSCHAMMER, D., ABERLE, H. (2023): Abschlussdokumentation Konzept Rasterdatenmanagementsystem, BKG. (adesso SE) Dortmund.

GDAL (2023): VRT - GDAL Virtual Format. https://gdal.org/drivers/raster/vrt.html (Abgerufen am 10. Juli 2023).

GEOPLANA (2023): 10 Fragen rund um die Bildflugplanung. https://www.geoplana.de/post/10-fragen-rund-um-die-bildflugplanung (Abgerufen am 10. Juli 2023).

GHISLA, A., WARMERDAM, F. (2008): Announcing the release of QGIS 0.11.0 'Metis'. https://lists.osgeo.org/pipermail/announce/2008-July/000097.html (Abgerufen am 24. Juli 2023).

GRATIER, T. (2015): QGIS diagram. https://github.com/webgeodatavore/qgis-class-diagram/tree/master (Abgerufen am 10. Juli 2023).

HOLTAN (2023): Norway in images: Webinar Orthophoto management and distribution, 19.04.2023. (Kartverket) Hønefoss.

IGN, ROK4 PROJECT (2023): Manage and distribute orthophotos: Webinar Orthophoto management and distribution, 19.04.2023. (Institute national de l'information géographique et forestière) Paris.

KAMINSKI (2016): Python 3. (De Gruyter Studium) Boston, Berlin.

LANGE, L. (2013): Geoinformatik in Theorie und Praxis. (Springer) Berlin, Heidelberg.

LEVI, S., PASCUAL, M. G. (2023): The legacy of Aaron Swartz: The computer programmer who became a martyr of the Freedom of Information movement. https://english.elpais.com/science-tech/2023-01-20/the-legacy-of-aaron-swartz-the-computer-programmer-who-became-a-martyr-of-the-freedom-of-information-movement.html (Abgerufen am 31. Juli 2023).

LUHMANN, T. (2017): Nahbereichsphotogrammetrie. (Wichmann) Berlin, Offenbach am Main.

HOWELL, J. (2019): Esri and ArcGIS. https://d3.harvard.edu/platform-digit/submission/esri-and-arcgis/# (Abgerufen am 08. Juli 2023).

HUISMAN, O. u. DE BY, R. (2009): Principles of Geographic Information Systems. (The International Institute for Geo-Information Science and Earth Observation) Enschede.

OGC (a) (2023): OGC Member List. https://www.ogc.org/about-ogc/ogc-member-list/ (Abgerufen am 10. Juli 2023)

OGC (b) (2023): OGC GeoTIFF Standard. https://www.ogc.org/standard/geotiff/ (Abgerufen am 01. August 2023).

OSGEO (2023): About OSGeo. https://www.osgeo.org/about/ (Abgerufen am 07. Juli 2023).

PYTHON (a) (2023): What is Python? Executive Summary. https://www.python.org/doc/essays/blurb (Abgerufen am 15. Juli 2023).

PYTHON (b) (2023): History and License. https://docs.python.org/3/license.html (Abgerufen am 15. Juli 2023).

QGIS (a) (2023): Vector selection. https://docs.qgis.org/3.28/en/docs/user_manual/processing_algs/qgis/vectorselection.html (Abgerufen am 08. Juli 2023).

QGIS (b) (2023): QGIS Documentation, PyQGIS Developer Cookbook. https://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/intro.html (Abgerufen am 24. Juli 2023).

QGIS (c) (2023): QGIS API Documentation. https://api.qgis.org/api/modules.html (Abgerufen am 24. Juli 2023).

SEIP, C., KORDUAN, P., ZEHNER, M. (2017): Web-GIS. (Wichmann) Berlin, Offenbach am Main.

SENATSVERWALTUNG BERLIN (2023): Digitale Landschaftsmodelle – ATKIS DLM. https://www.berlin.de/sen/sbw/stadtdaten/geoportal/landesvermessung/geotopographie-atkis/dlm-digitale-landschaftsmodelle/ (Abgerufen am 10. Juli 2023).

SHERMAN, G. (2009): Announcing the release of QGIS 1.0 'Kore'. https://lists.osgeo.org/pipermail/qgis-developer/2009-January/005774.html (Abgerufen am 24. Juli 2023).

TECHNAVIO (2023): GIS Market by Product, Deployment, and Geography - Forecast and Analysis 2023-2027. https://www.technavio.com/report/gis-market-industry-analysis (Abgerufen am 08. Juli 2023).

WIKIPEDIA (a) (2023): ArcGIS. https://en.wikipedia.org/wiki/ArcGIS (Abgerufen am 09. Juli 2023).

WIKIPEDIA (b) (2023): ArcInfo. https://en.wikipedia.org/wiki/ArcInfo (Abgerufen am 09. Juli 2023).

WIKIPEDIA (c) (2023): Jack Dangermond. https://en.wikipedia.org/wiki/Jack_Dangermond (Abgerufen am 09. Juli 2023).

WIKIPEDIA (d) (2023): QGIS. https://de.wikipedia.org/wiki/QGIS (Abgerufen am 10. Juli 2023).

WIKIPEDIA (e) (2023): Python. https://de.wikipedia.org/wiki/Python_(Programmiersprache) (Abgerufen am 11. Juli 2023).

WIKIPEDIA (f) (2023): TIFF. https://en.wikipedia.org/wiki/TIFF (Abgerufen am 02. August 2023).

W3C (2009): UK Government Moves to Put Data on the Web. https://www.w3.org/News/2009#entry-6622 (Abgerufen am 01. August 2023).

ZEILE, P., HOVESTADT, L., HIRSCHBERG, U., FRITZ, O. (2020): Atlas of Digital Architecture. (Birkhäuser) Basel.

Eidesstattliche Erklärung

JADE HOCHSCHULE
Wilhelmshaven Oldenburg Elsfleth

Name: Hackenberg

Vorname: Moritz

Matrikelnummer: 6033436

Erklärung gemäß § 21 (5) Allgemeiner Teil (Teil A) der Prüfungsordnung für die Bachelor-Studiengänge (BPO) an der Jade Hochschule Wilhelmshaven/Oldenburg/Elsfleth in der Fassung der Bekanntmachung vom 08. Dezember 2004 (VkBl. Nr. 37/2004), zuletzt geändert am 21.10.2014 (VkBl. Nr. 56/2014 vom 24. November 2014)

Die Bachelor-Arbeit ist

☒ eine Einzelarbeit.
☐ eine Gruppenarbeit zusammen mit der/dem Studierenden:

_____.

Ich versichere hiermit, die Bachelor-Arbeit

☐ bei einer Gruppenarbeit den/die Teil(e)

Automatisierung des Aktualisierungsprozesses von umfangreichen Rasterdaten-Sammlungen
- Ein Vergleich der Umsetzung mit ArcGIS Pro (Mosaic Datasets) und QGIS (GDAL Virtual Raster)

selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Walsrode, 10.08.2023
(Ort, Datum)

Hackenberg
(Unterschrift Studierende/r)