GIS GRASS als WPS Backend

Neue Entwicklungen im GIS GRASS zur Unterstützung von WPS Servern

Sören Gebbert

www.giscoder.de

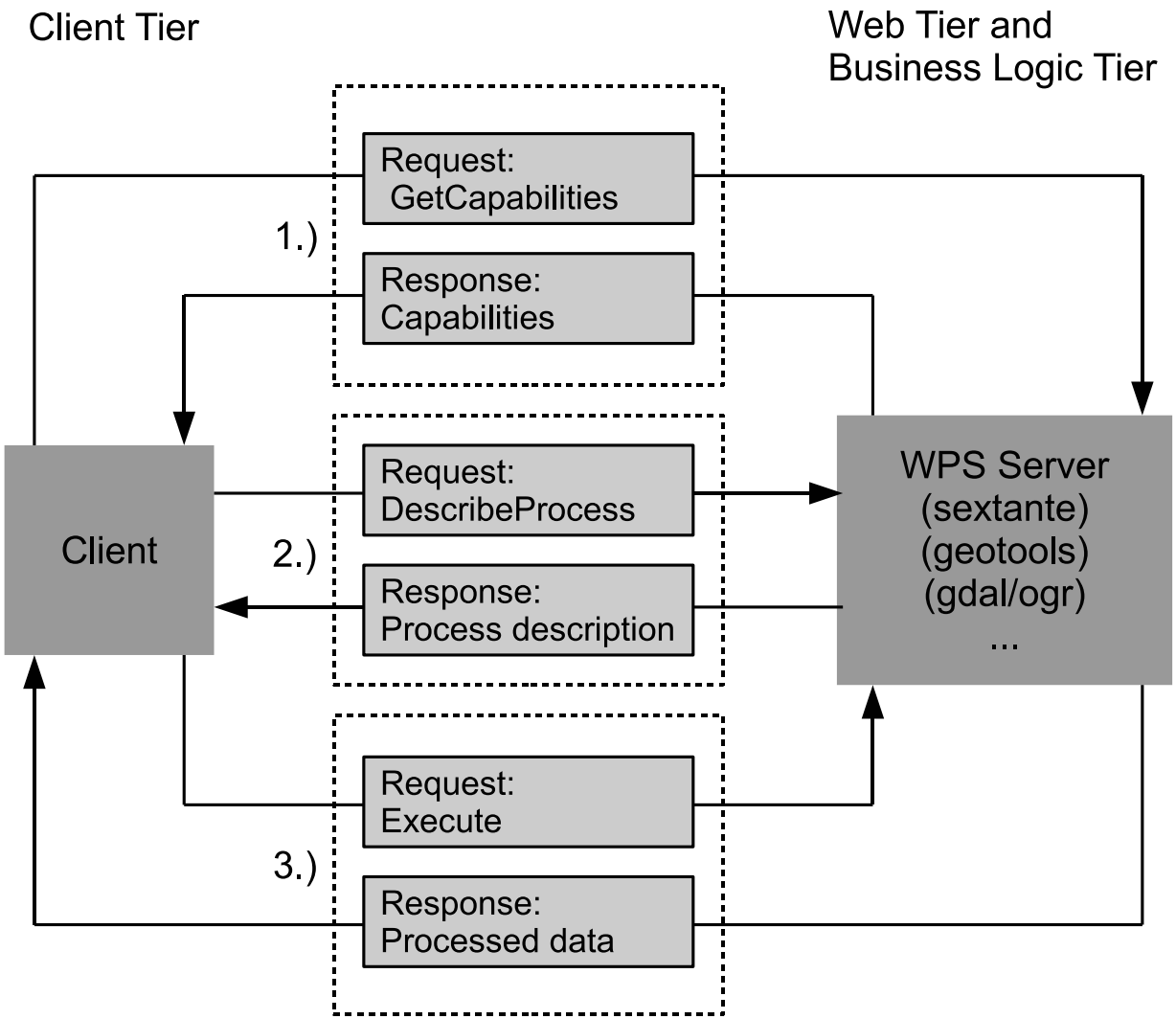
Inhalt

Geodaten und Webservices

Web-Processing-Service (WPS)

- Für die Verarbeitung von Geodaten wurde vom OGC der Web-Processing-Service Standard entwickelt
- Aktuelle Version WPS 1.0.0 wurde in verschiedenen freien WPS Servern implementiert
  - 52North WPS Server
  - deegree3 WPS Server
  - PyWPS
  - ZOO-Project WPS Server
- Als Backend/Business Logic werden zumeist gdal/ogr, geotools und sextante eingesetzt
- Das GIS GRASS spielt bis jetzt keine große Rolle als WPS Backend

Web-Processing-Service



GIS GRASS Kurzvorstellung

Was ist GIS GRASS

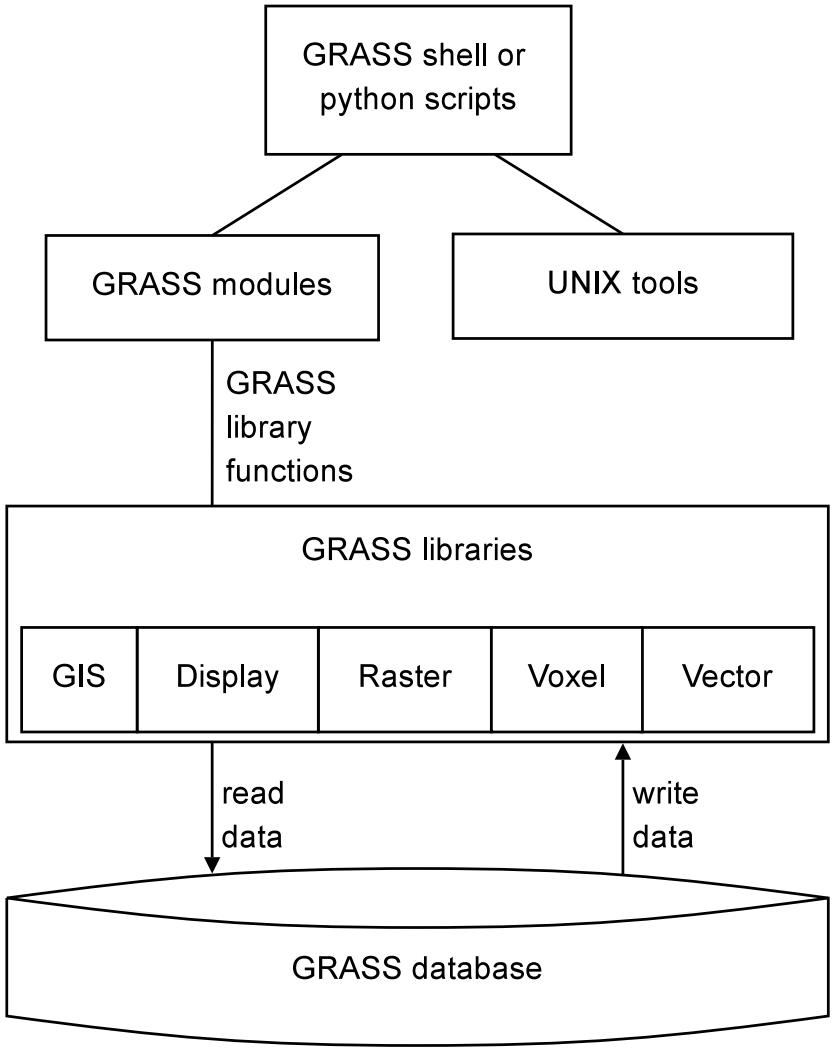
- Ein kombiniertes Raster-/Vektor- und Voxel-GIS
- Open Source, vollständig unter der GPL lizenziert
- Kann von der Kommandozeile als auch von einer Benutzeroberfläche aus bedient werden
- Stellt über 350 Module zur Verfügung
- 1982-1995 am CERL in Illinois/USA entwickelt
- Seit 1999 als Open-Source von aktiver Entwicklergemeinde weiter entwickelt

GIS GRASS als WPS Backend

Warum GIS GRASS als WPS Backend?

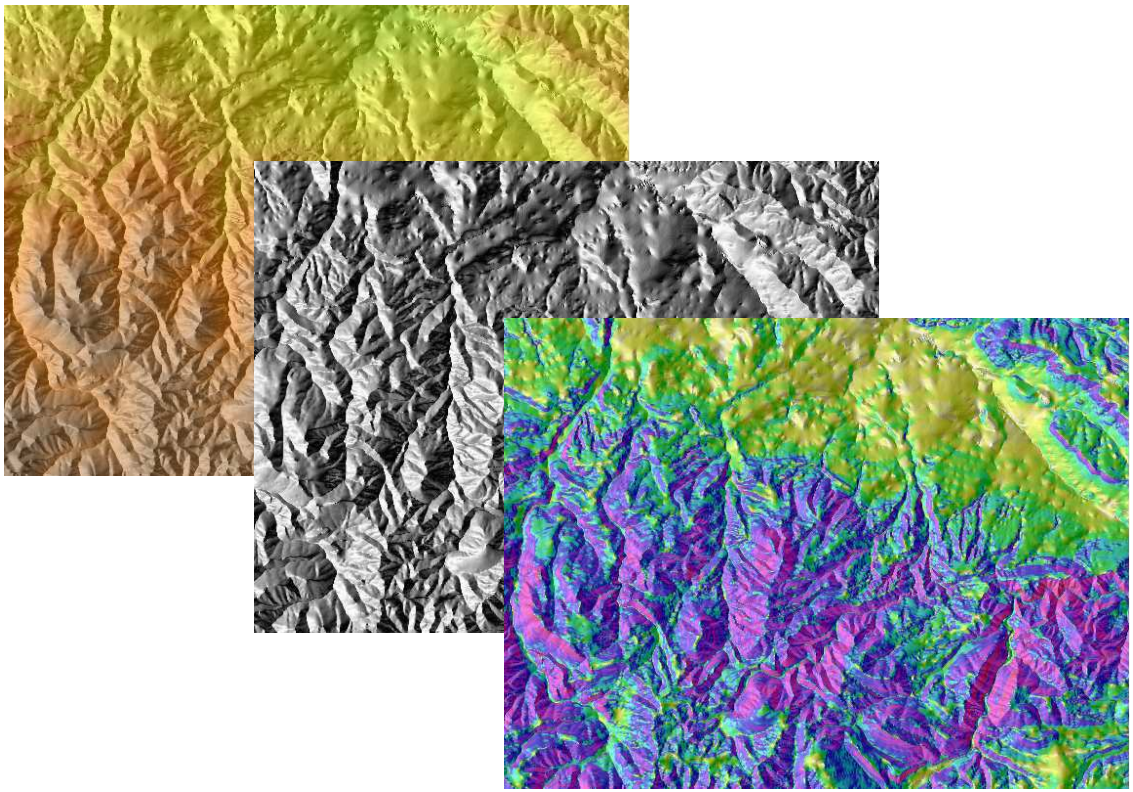
- Modularer Aufbau
- Fähigkeit zum Batch-Betrieb
- Großer Funktionsumfang im Bereich Raster-, Bild- und Vektordatenverarbeitung
- Parallele Datenverarbeitung (z.B. auf einem Cluster)
- Klar definierte maschinenlesbare Modulschnittstellen
- GDAL/OGR Import/Exportschnittstelle
- Hoch performant
- Geringer Speicherverbrauch bei der Verarbeitung großer Rasterdatensätze

Modularer Aufbau des GIS GRASS



Rasterfunktionalität

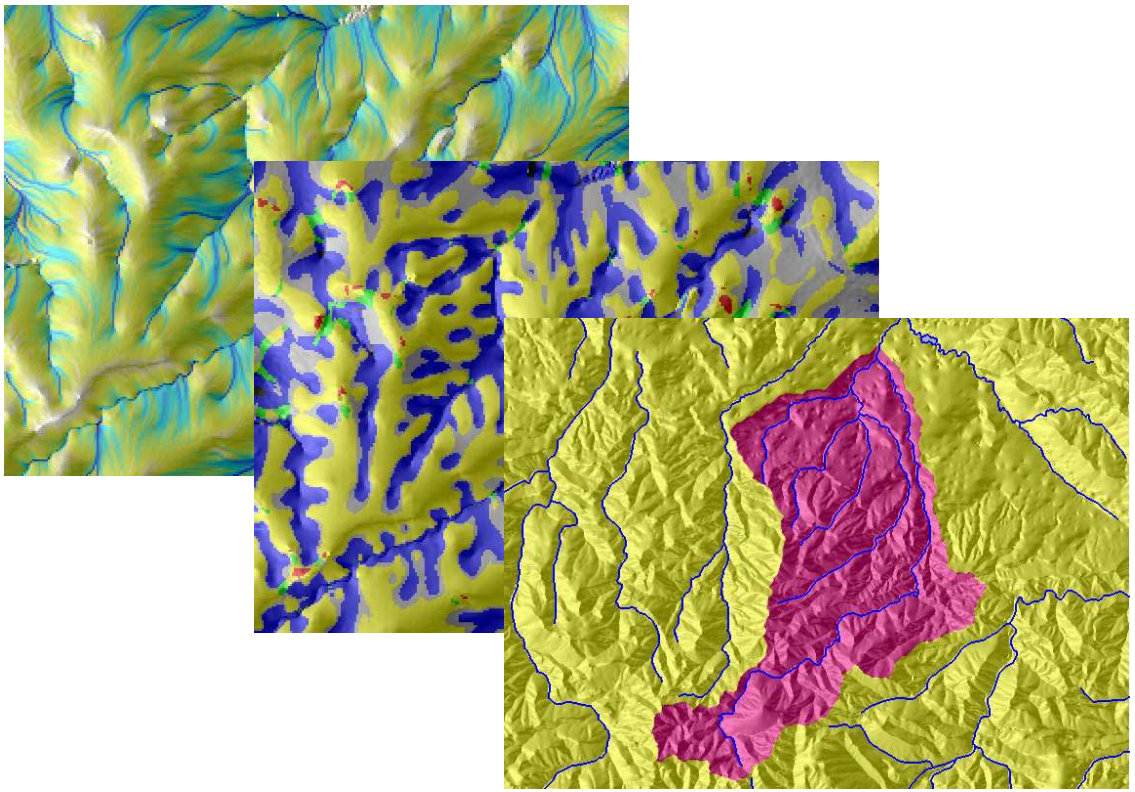
Satellitenbildverarbeitung, digitalen Geländemodelle, ...



- Spline Interpolation
- Hangneigungen
- Exposition
- Resampling
- Kartenalgebra
- Principal components
- ...

Hydrologische Modellierung

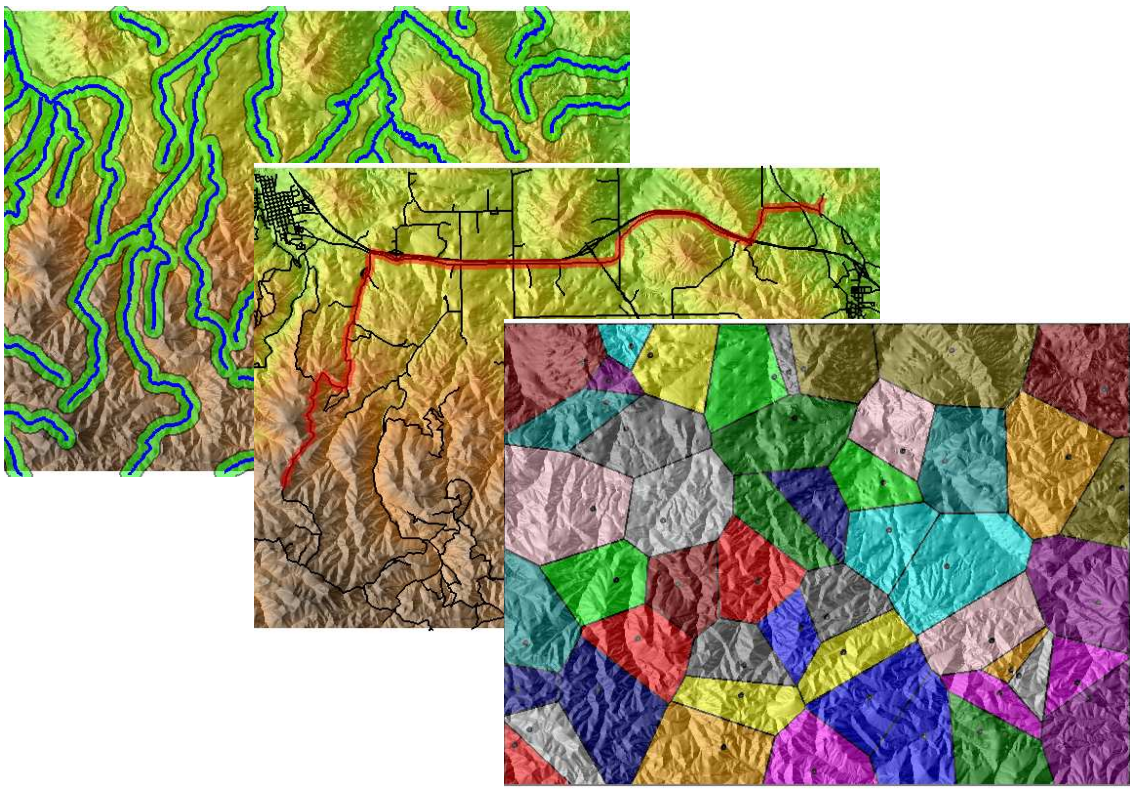
Modellierung hydro(geo)logischer Fragestellungen



- Einzugsgebiete
- Topografie
- Oberflächenabfluss
- Grundwasser-modellierung
- Starkregenereignisse
- Erosionsberechnung
- ...

Vektorfunktionalität

Erstellung und Verarbeitung von topologischen Vektordaten



- Generalize,Buffer
- Patch,Overlay, Select,Extract
- Shortest Path
- Traveling salesman problem
- Delaunay und Voronoi Triangulierung
- ...

Einheitliche Modulbeschreibung

C Beispiel

```
struct Module *module;
struct Option *raster;
struct Flag *f;

module = G_define_module();
module->description = _("g.parser test program");

raster = G_define_option();
raster->key = "raster";
raster->type = TYPE_STRING;
raster->gisprompt = "old,cell,raster";
raster->description = _("Raster input map");

f = G_define_flag();
f->key = 'f';
f->description = _("A flag");
```

Einheitliche Modulbeschreibung

Python, Shell und Perl Beispiel

```
#%module
#%  description: g.parser test script
#%end
#%flag
#%  key: f
#%  description: A flag
#%end
#%option
#%  key: raster
#%  type: string
#%  gisprompt: old,cell,raster
#%  description: Raster input map
#%  required : yes
#%end
```

Einheitliche Modulbeschreibung

Ausgabe auf Kommandozeile

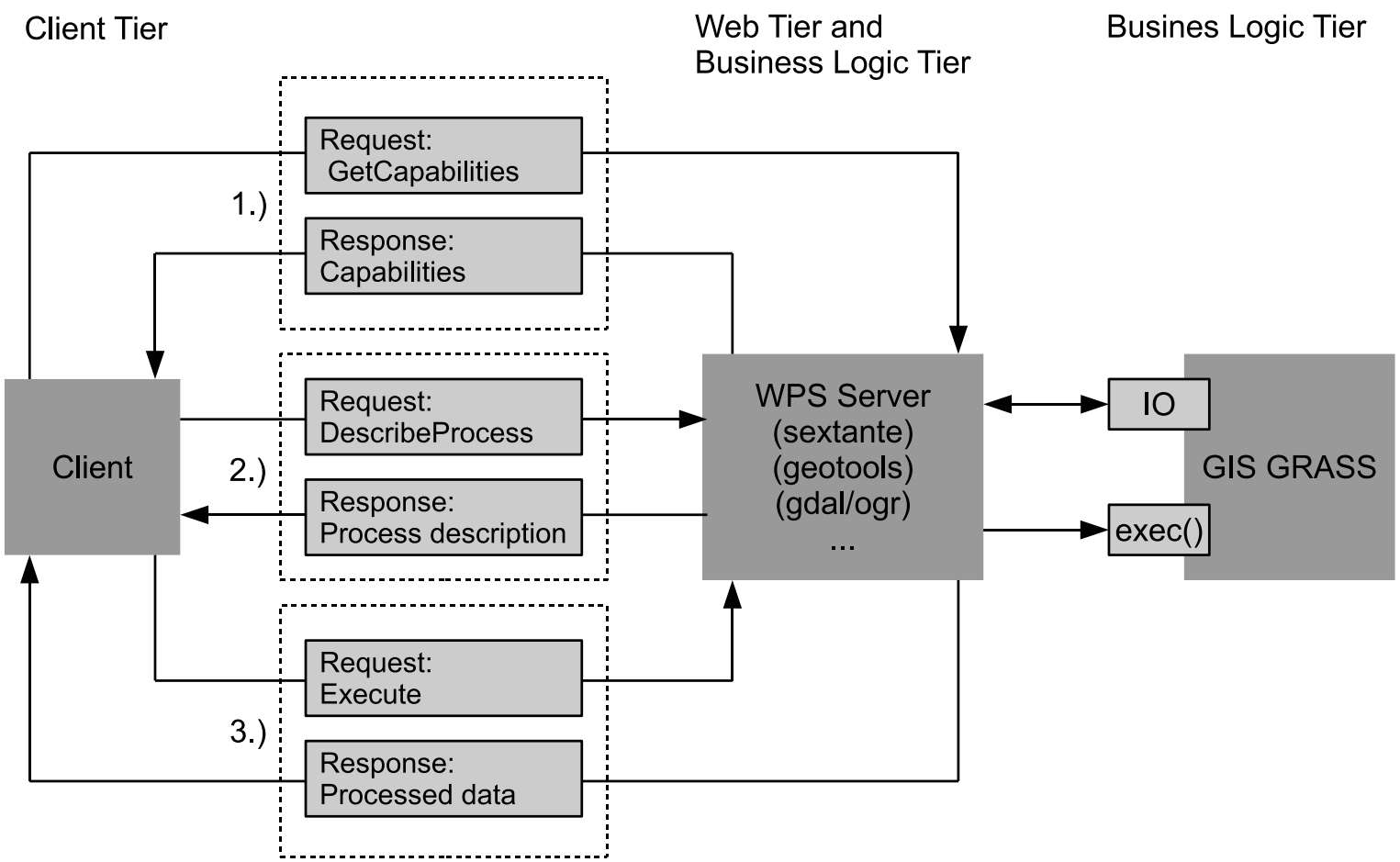
```
Description:
 g.parser test script

Usage:
 PythonExample.py [-f] raster=string [--verbose] [--quiet]

Flags:
  -f   A flag
 --v   Verbose module output
 --q   Quiet module output

Parameters:
  raster   Raster input map
```

WPS Server Integration

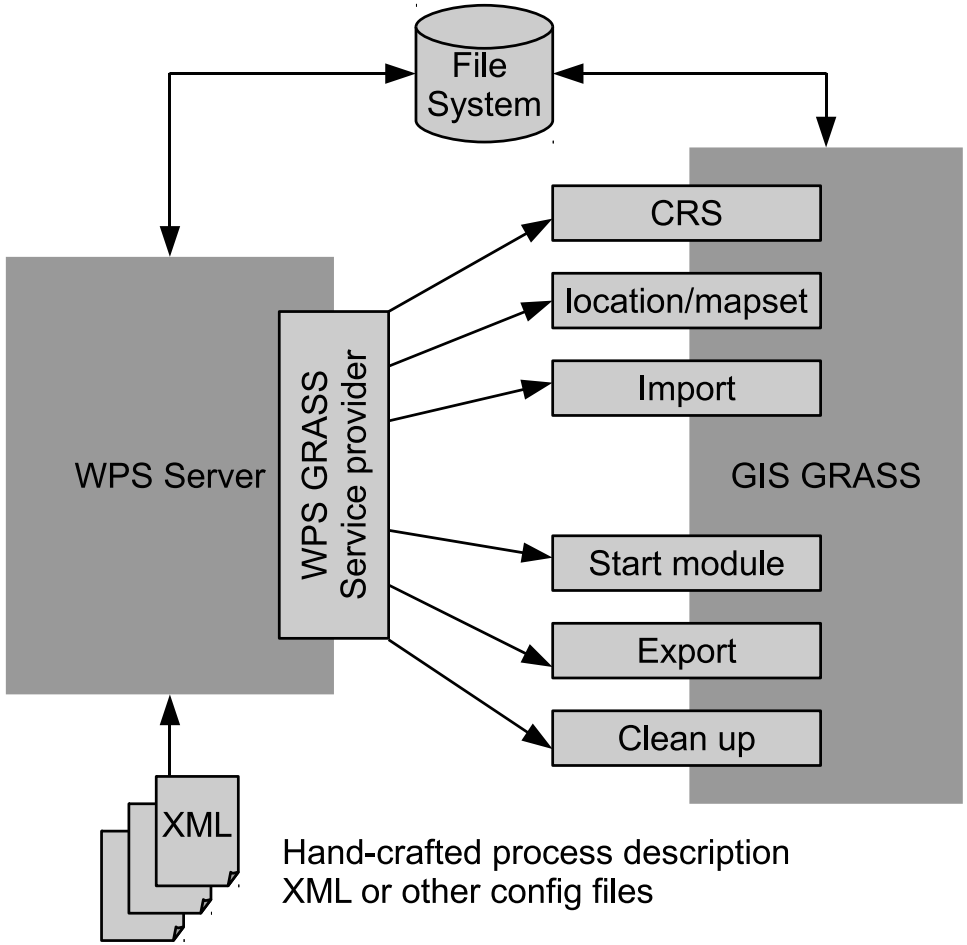
Vertiefte Kenntnisse von GRASS nötig

- Die Verarbeitung von Daten in GRASS erfolgt in Locations
- Locations haben ein festes Referenzkoordinatensystem
- Locations können mehrere Mapsets enthalten
- Für den Start von GRASS Modulen müssen eine Reihe von Umgebungsvariablen gesetzt werden
- GRASS hat eigene Datenstruktur, Datei-Import und Export ist notwendig
- GRASS Module können nur als eigenständige Prozesse gestartet werden

WPS Server Integration



Bestehendes Integrationskonzept

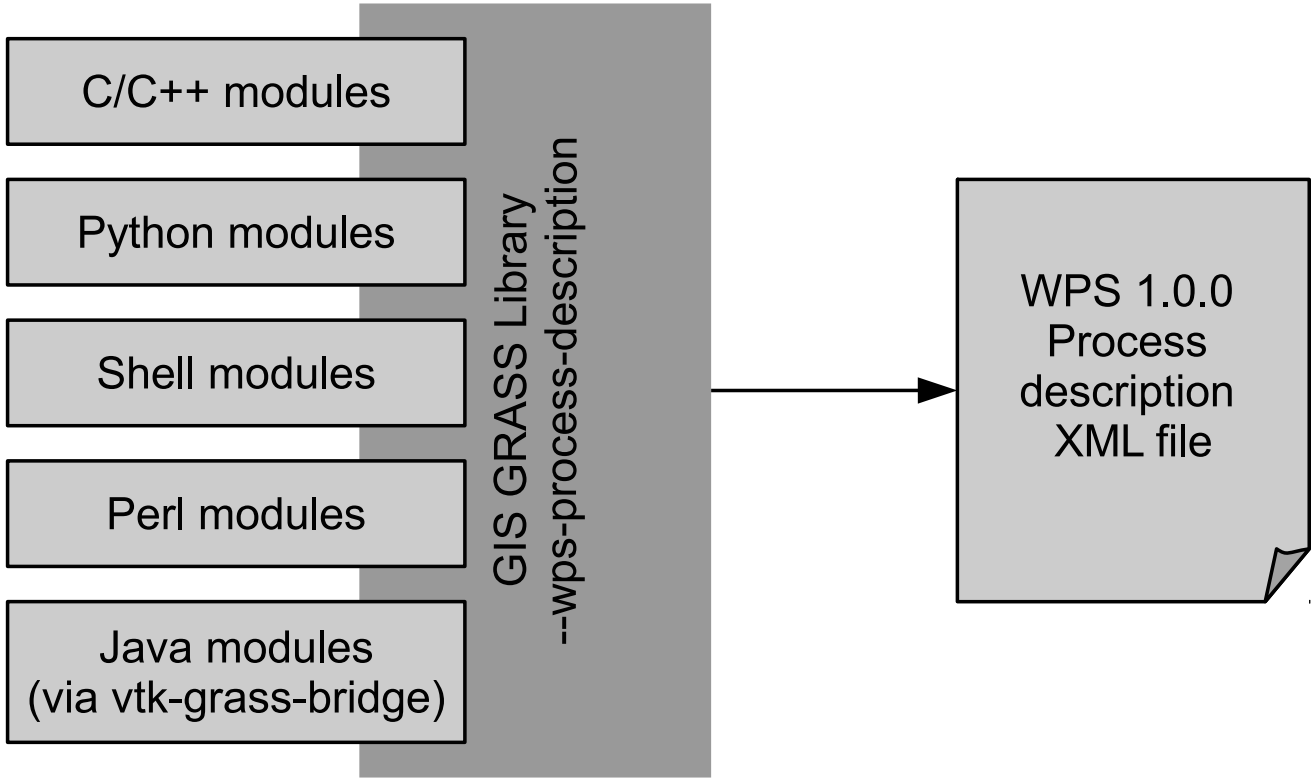


Was ist neu?

Neue Entwicklungen in GIS GRASS Version 7

- WPS Konforme GRASS Modulbeschreibungen
- Direktes Lesen und Schreiben von Rasterdateien
- Interaktivität aus allen GRASS Modulen entfernt
- Allgemeines Framework für die Integration in WPS Server
- Umstellung von Shell auf Python
- Signifikante Geschwindigkeitssteigerung in vielen Modulen
- Generierung der Vektortopologie erfolgt nun im Dateisystem, geringer Hauptspeicherbedarf bei Vektoranalyse

WPS Modulbeschreibung



WPS Modulbeschreibung

Welche Module werden unterstützt?

- Großteil der verarbeitenden Rastermodule (r.*)
- Großteil der verarbeitenden Vektormodule (v.*)
- Einige Bildverarbeitungsmodule (Gruppenkonzept ist nicht WPS kompatibel)
- Wrapper für Gruppen auf Python oder Shell-Basis leicht zu implementieren
- Kein Support für Voxel- (r3.*) oder Displaymodule (d.*)

WPS Modulbeschreibung

Welche Datenformate werden unterstützt?

- GeoTiff und GML 3.1.0 für Input und Output
- Unterstützung aller GDAL/OGR Formate problemlos möglich
- Normale Textdateien (text/plain) als Input und Output
- Input Features:
  - einzeln oder multiple
  - complex oder literal (Boolean, Integer, Float und Strings)
  - obligatorisch oder optional
- Momentan nur komplexe Outputs unterstützt (obligatorisch oder optional)
- BoundingBox-Unterstützung ist in Vorbereitung

WPS Modulbeschreibung

Rasterkarte als WPS ComplexData

```
<Input minOccurs="1" maxOccurs="1">
  <ows:Identifier>raster</ows:Identifier>
  <ows:Title>Raster input map</ows:Title>
  <ComplexData maximumMegabytes="2048">
    <Default>
      <Format>
        <MimeType>image/tiff</MimeType>
      </Format>
    </Default>
    <Supported>
      <Format>
        <MimeType>image/tiff</MimeType>
      </Format>
    </Supported>
  </ComplexData>
</Input>
```

WPS Modulbeschreibung

Vektorkarte als WPS ComplexData

```
<Input minOccurs="1" maxOccurs="1">
  <ows:Identifier>map</ows:Identifier>
  <ows:Title>Data source for OGR access</ows:Title>
  <ComplexData maximumMegabytes="2048">
    <Default>
      <Format>
        <MimeType>text/xml</MimeType>
        <Encoding>UTF-8</Encoding>
        <Schema>http://schemas.opengis.net/gml/3.1.0/polygon.xsd</Schema>
      </Format>
    </Default>
    <Supported>
      <Format>
        <MimeType>text/xml</MimeType>
        <Encoding>UTF-8</Encoding>
        <Schema>http://schemas.opengis.net/gml/3.1.0/polygon.xsd</Schema>
      </Format>
    </Supported>
  </ComplexData>
</Input>
```

WPS Modulbeschreibung

Textdatei als WPS ComplexData

```
<Input minOccurs="0" maxOccurs="1">
  <ows:Identifier>weight</ows:Identifier>
  <ows:Title>Text file containing weights</ows:Title>
  <ComplexData maximumMegabytes="2048">
    <Default>
      <Format>
        <MimeType>text/plain</MimeType>
      </Format>
    </Default>
    <Supported>
      <Format>
        <MimeType>text/plain</MimeType>
      </Format>
    </Supported>
  </ComplexData>
</Input>
```

WPS Modulbeschreibung

Flags als WPS LiteralData

```
<Input minOccurs="0" maxOccurs="1">
  <ows:Identifier>-f</ows:Identifier>
  <ows:Title>A flag</ows:Title>
  <LiteralData>
    <ows:DataType ows:reference="xs:boolean">boolean</ows:DataType>
    <ows:AllowedValues>
      <ows:Value>true</ows:Value>
      <ows:Value>false</ows:Value>
    </ows:AllowedValues>
    <DefaultValue>false</DefaultValue>
  </LiteralData>
</Input>
```

WPS Modulbeschreibung

Ganzzahlen als WPS LiteralData

```
<Input minOccurs="0" maxOccurs="1">
  <ows:Identifier>size</ows:Identifier>
  <ows:Title>Neighborhood size</ows:Title>
  <LiteralData>
    <ows:DataType ows:reference="xs:integer">integer</ows:DataType>
    <ows:AnyValue/>
    <DefaultValue>3</DefaultValue>
  </LiteralData>
</Input>
```

WPS Modulbeschreibung

Fließkommazahlen als WPS LiteralData

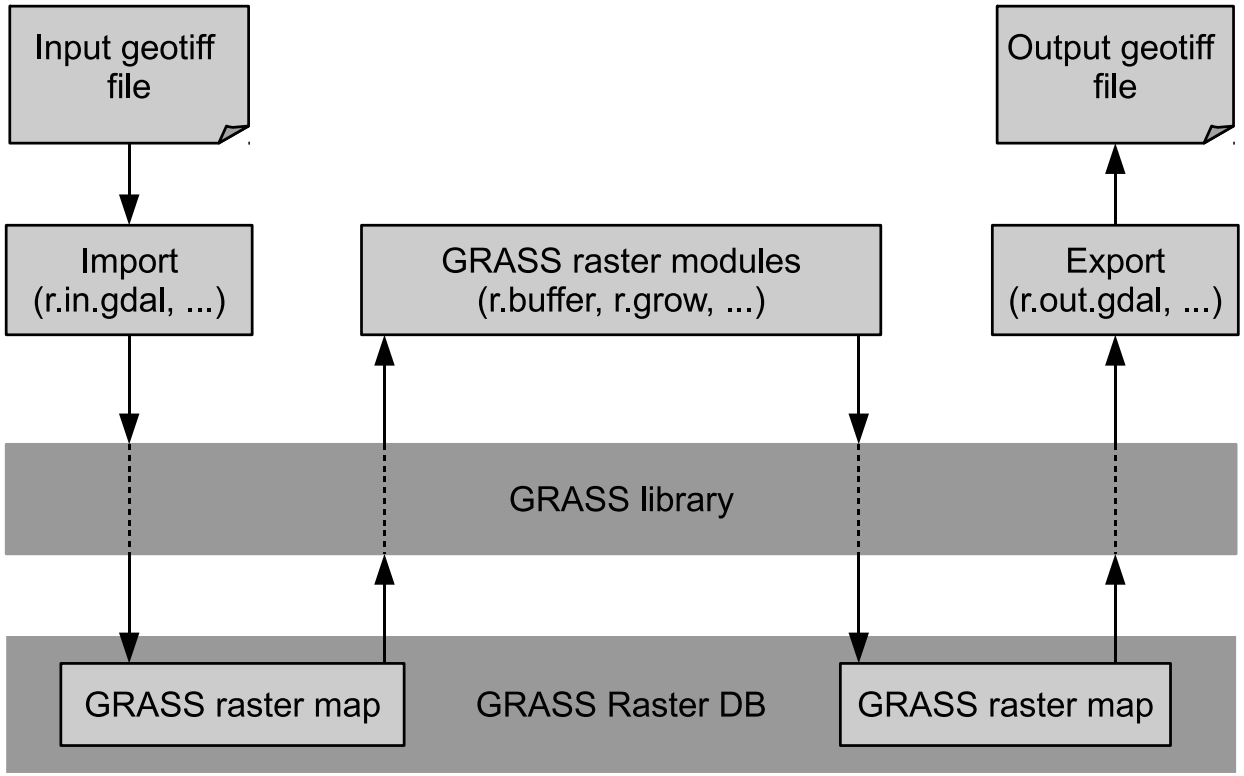
```
<Input minOccurs="0" maxOccurs="1">
  <ows:Identifier>gauss</ows:Identifier>
  <ows:Title>Sigma (in cells) for Gaussian filter</ows:Title>
  <LiteralData>
    <ows:DataType ows:reference="xs:float">float</ows:DataType>
    <ows:AnyValue/>
  </LiteralData>
</Input>
```

WPS Modulbeschreibung

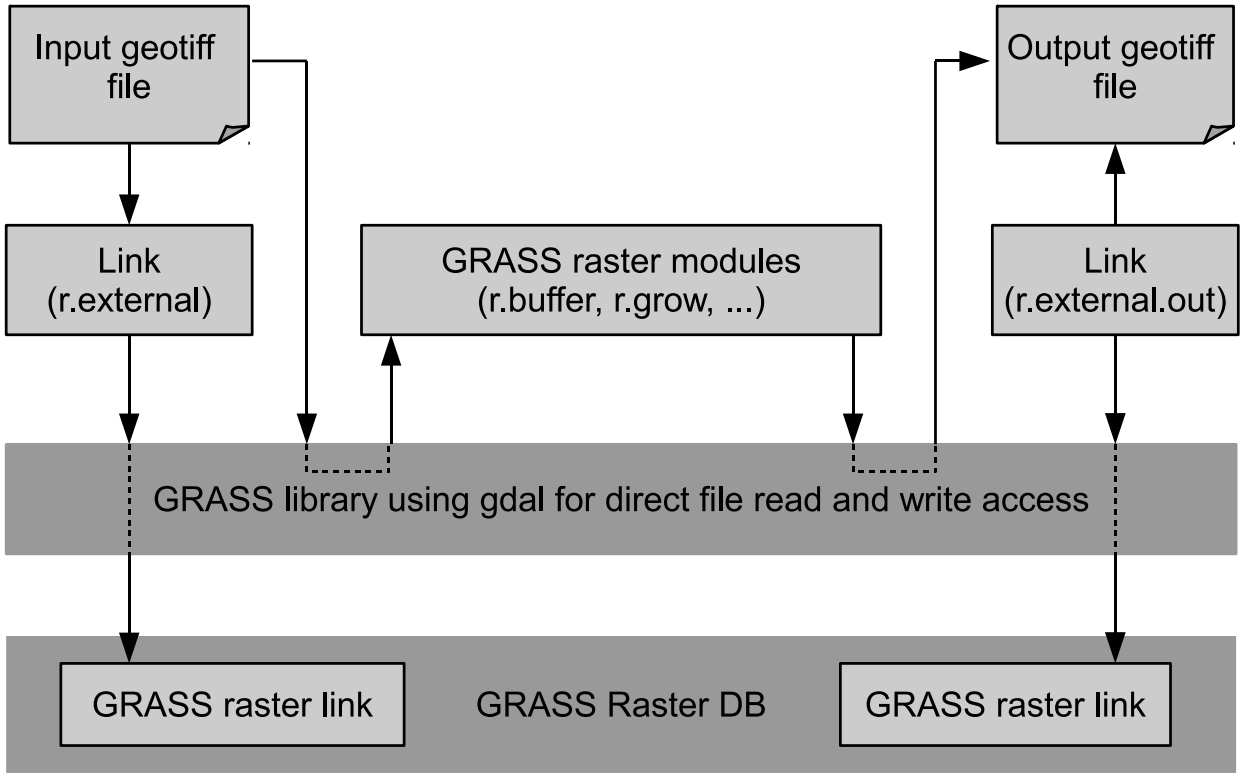
Zeichenketten als WPS LiteralData

```
<Input minOccurs="0" maxOccurs="1">
  <ows:Identifier>title</ows:Identifier>
  <ows:Title>Title of the output raster map</ows:Title>
  <LiteralData>
    <ows:DataType ows:reference="xs:string">string</ows:DataType>
    <ows:AnyValue/>
  </LiteralData>
</Input>
```

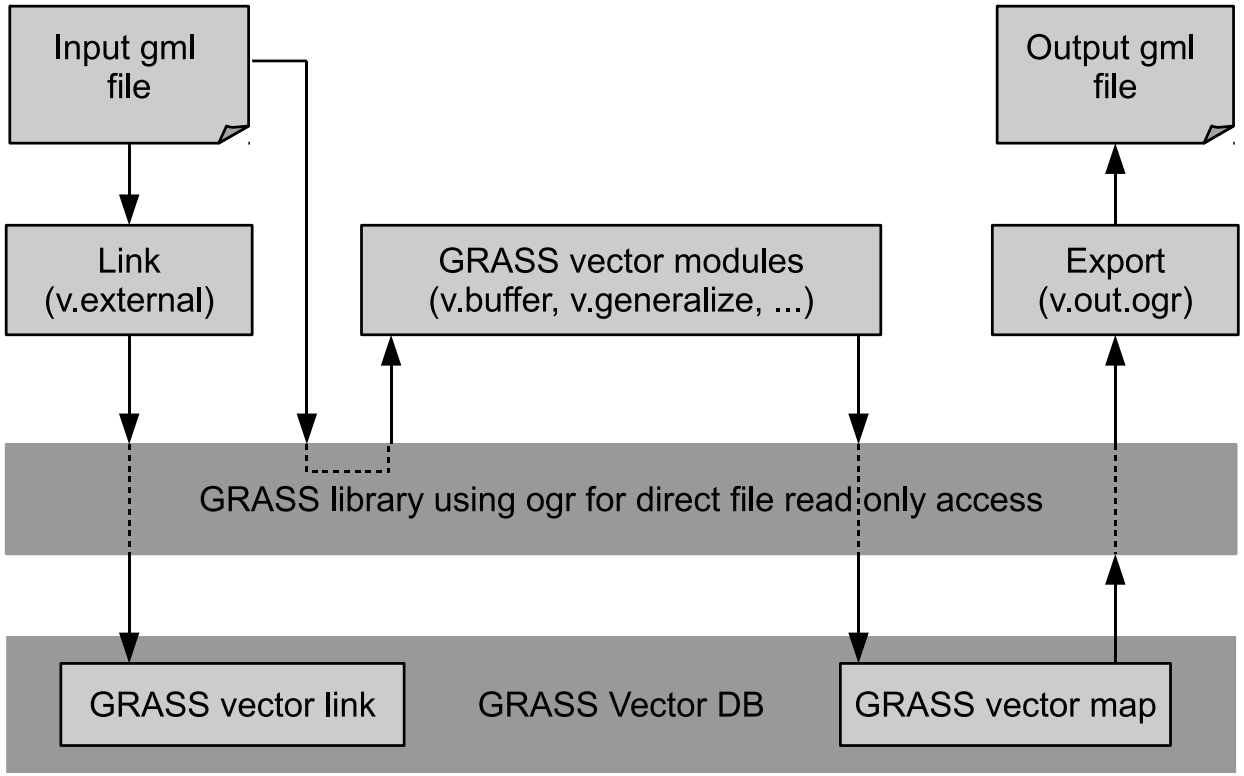
Import und Export von Rasterdaten



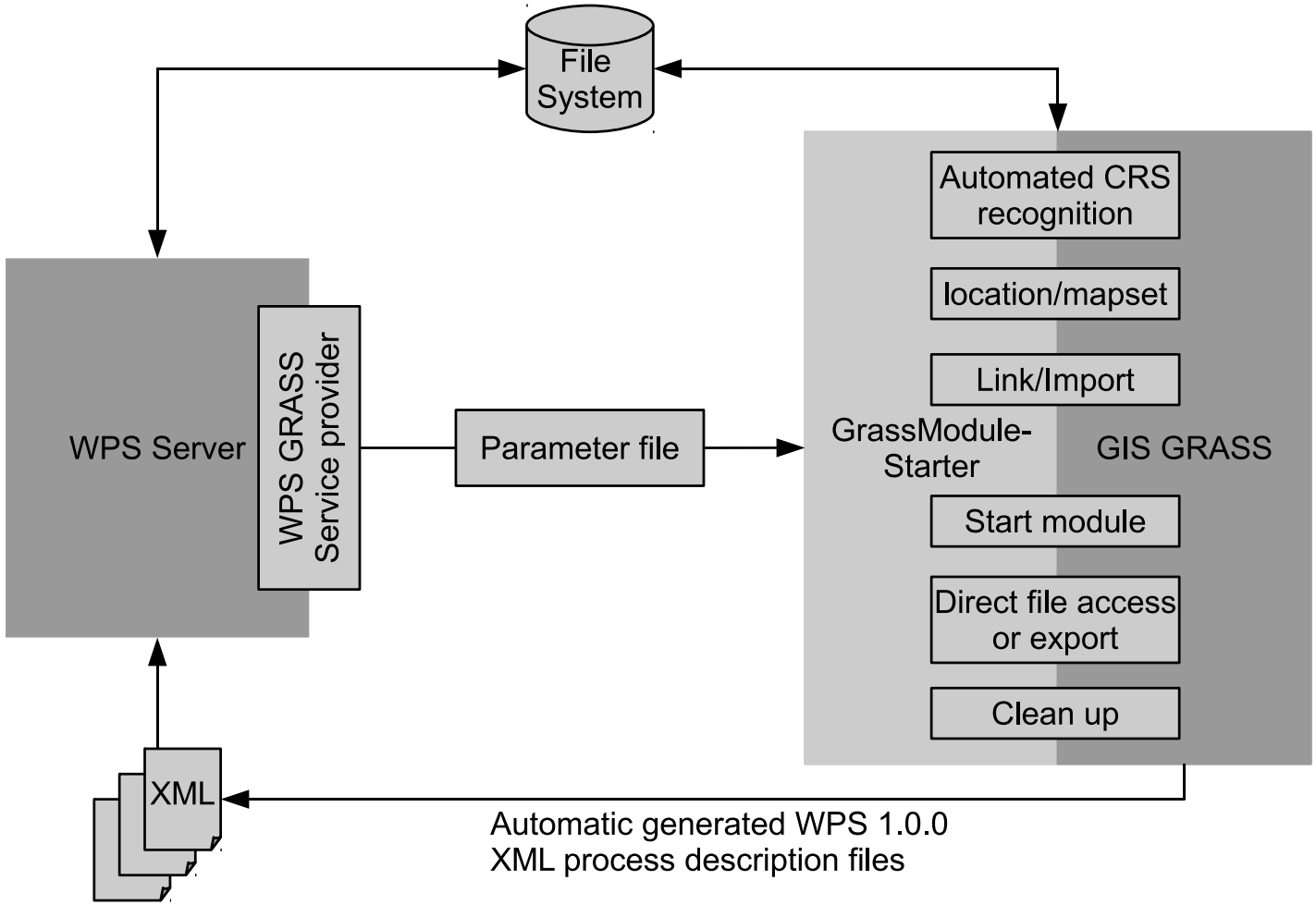
Direkter Dateizugriff auf Rasterdaten



Direkter Zugriff auf Vektordaten



Vorschlag neues Integrationskonzept



GRASS-Integration mittels GrassModuleStarter

Programmkonzept GrassModuleStarter

- Pythonmodul für Kommandozeilenaufruf oder Import in Pythonprogramme
- Einfache Key-Value Parameterdatei als Eingabe
- Lesen der Eingabedaten vom Dateisystem
- Rechenergebnisse werden im Dateisystem abgelegt
- Verarbeitet allgemeine GRASS Parameter aus der WPS Prozessbeschreibung (Rasterauflösung, BoundingBox)
- Aufruf aller GRASS Module in temporärer Location

GRASS-Integration mittels GrassModuleStarter

Programmablauf GrassModuleStarter

- Setzen aller benötigten GRASS Umgebungsvariablen
- Generierung einer temporären Location auf Basis Referenzkoordinatensystems des ersten Raster/Vektor Inputs
- Linken oder Import der Eingangsdaten
- Aufruf des GRASS Moduls
- Export der Ergebnisse in vorgegebenen Verzeichnispfad
- Löschen aller temporären Daten
- Generierung Fehlerreport

GRASS-Integration mittels GrassModuleStarter

Systemeinstellungen der GrassModulestarter Eingabedatei

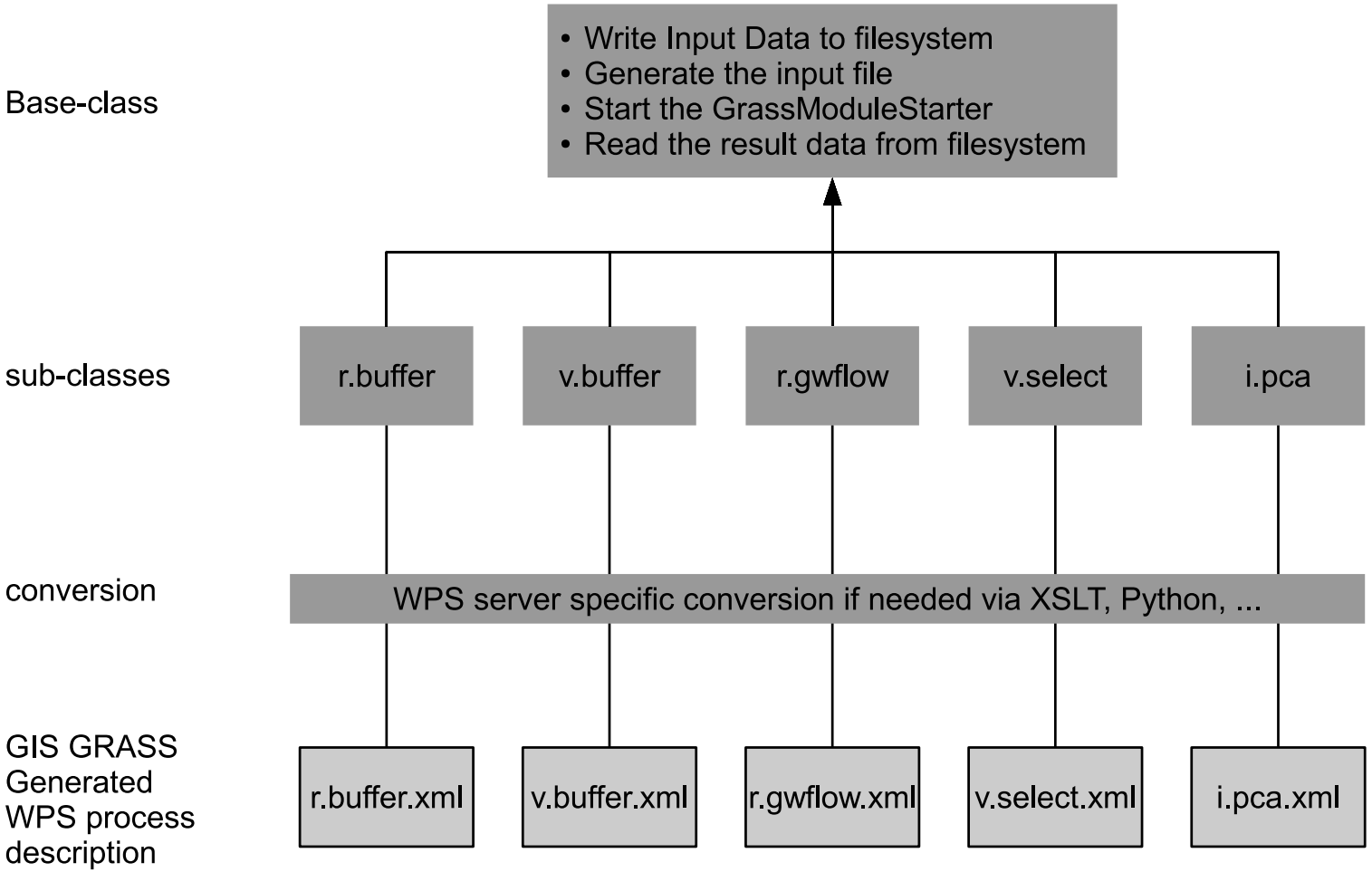
```
[System]
 WorkDir=/tmp
 OutputDir=/tmp

[GRASS]
 GISBASE=/home/soeren/src/grass7.0/grass_trunk/dist.i686-pc-linux-gnu
 GRASS_ADDON_PATH=
 GRASS_VERSION=7.0.svn
 Module=r.contour
 LOCATION=
 LinkInput=TRUE
 IgnoreProjection=FALSE
 UseXYLocation=FALSE
```

GRASS-Integration mittels GrassModuleStarter

Ein- und Ausgabedaten in der GrassModulestarter Eingabedatei

```
[ComplexData]
 Identifier=input
 PathToFile=/tmp/srtm90.tiff
 MimeType=image/tiff

[LiteralData]
 Identifier=levels
 DataType=double
 Value=50

[ComplexOutput]
 Identifier=output
 PathToFile=/tmp/srtm90contour.gml
 MimeType=text/xml
 Encoding=UTF-8
 Schema=http://schemas.opengis.net/gml/3.1.0/polygon.xsd
```

Implementierungsvorschlag



Implementierungsvorschlag

52North und ZOO-Project

- 52North WPS Server:
  - Basisklasse von *AbstractAlgorithm* ableiten, gegebenfalls Implementierung von Parser und Generatoren
  - GRASS WPS Prozessbeschreibungen können ohne Konvertierung übernommen werden (gegebenfalls Anpassungen notwendig)
- ZOO-Project WPS Server
  - Basisklasse in Python, Java oder C++ implementieren
  - Konvertierung der GRASS WPS Prozessbeschreibungen in zcfg Format mittels Python (PyXB)

Software

GIS GRASS Version 7

http://grass.osgeo.org

GrassModuleStarter aus vtk-grass-bridge

code.google.com/p/vtk-grass-bridge

52North

http://www.52north.org

ZOO-Project

www.zoo-project.org

Danke

Vielen Dank für Ihre Aufmerksamkeit

Ende

GIS CODER

www.giscoder.de