

gvSIG CE (Community Edition) Schnittstellen zur Automatisierung und Programmierung

Benjamin Ducke
GIS-Fachberater und -Entwickler
benducke@fastmail.fm

1. März 2013

Überblick der Schnittstellen



Zahlreiche Schnittstellen:

- gvSIG (CE): Java-API, Python-Skripte (Jython)
- SEXTANTE: Java API, Skripte, *Modeler*
- GRASS-Backend: C, *Unix Shell Scripting*
- SAGA-Backend: C++
- R-Backend: R (Skripte)
- (weitere Backends für SEXTANTE)

Überblick der Schnittstellen



Zahlreiche Schnittstellen:

- gvSIG (CE): Java-API, Python-Skripte (Jython)
- SEXTANTE: Java API, Skripte, *Modeler*
- GRASS-Backend: C, *Unix Shell Scripting*
- SAGA-Backend: C++
- R-Backend: R (Skripte)
- (weitere Backends für SEXTANTE)

Überblick der Schnittstellen



Zahlreiche Schnittstellen:

- gvSIG (CE): Java-API, Python-Skripte (Jython)
- SEXTANTE: Java API, Skripte, *Modeler*
- GRASS-Backend: C, *Unix Shell Scripting*
- SAGA-Backend: C++
- R-Backend: R (Skripte)
- (weitere Backends für SEXTANTE)

Überblick der Schnittstellen



Zahlreiche Schnittstellen:

- gvSIG (CE): Java-API, Python-Skripte (Jython)
- SEXTANTE: Java API, Skripte, *Modeler*
- GRASS-Backend: C, *Unix Shell Scripting*
- SAGA-Backend: C++
- R-Backend: R (Skripte)
- (weitere Backends für SEXTANTE)

Überblick der Schnittstellen



Zahlreiche Schnittstellen:

- gvSIG (CE): Java-API, Python-Skripte (Jython)
- SEXTANTE: Java API, Skripte, *Modeler*
- GRASS-Backend: C, *Unix Shell Scripting*
- SAGA-Backend: C++
- R-Backend: R (Skripte)
- (weitere Backends für SEXTANTE)

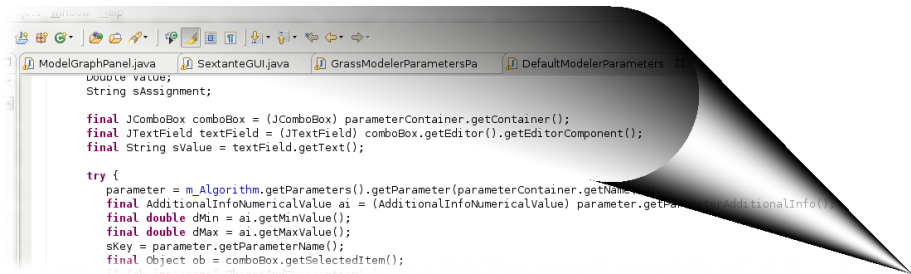
Überblick der Schnittstellen



Zahlreiche Schnittstellen:

- gvSIG (CE): Java-API, Python-Skripte (Jython)
- SEXTANTE: Java API, Skripte, *Modeler*
- GRASS-Backend: C, *Unix Shell Scripting*
- SAGA-Backend: C++
- R-Backend: R (Skripte)
- (weitere Backends für SEXTANTE)

Programmierung von gvSIG CE (1/2)



```
ModelGraphPanel.java | SextanteGUI.java | GrassModelerParameters | DefaultModelerParameters
double value;
String sAssignment;

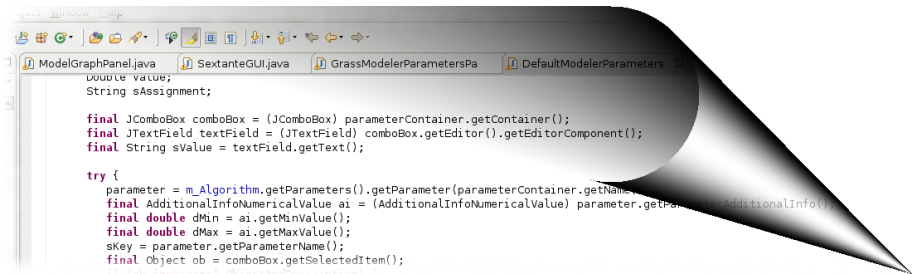
final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();
final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();
final String sValue = textField.getText();

try {
    parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getName());
    final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getParameterAdditionalInfo();
    final double dMin = ai.getMinValue();
    final double dMax = ai.getMaxValue();
    sKey = parameter.getParameterName();
    final Object ob = comboBox.getSelectedItem();
}
```

- optimiert für Arbeit mit Eclipse

- Quellcode im SVN: <http://sourceforge.net/projects/gvsigce/>
- *sehr* umfangreiche API, teils in Spanisch kommentiert
- schwierig zu erlernen, viele „Baustellen“
- http://gvsigce.sourceforge.net/wiki/index.php/Development_and_releases/

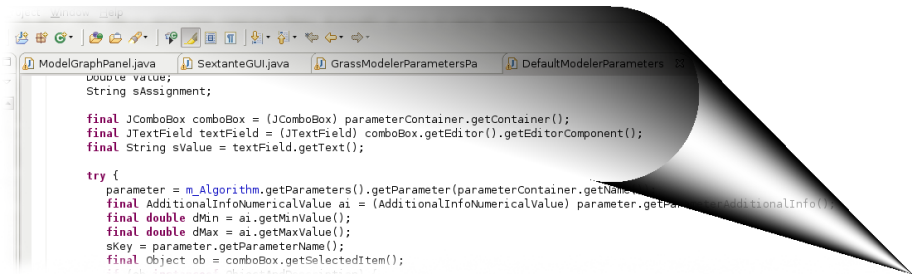
Programmierung von gvSIG CE (1/2)



```
Double value;  
String sAssignment;  
  
final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();  
final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();  
final String sValue = textField.getText();  
  
try {  
    parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getName());  
    final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getParameterAdditionalInfo();  
    final double dMin = ai.getMinValue();  
    final double dMax = ai.getMaxValue();  
    sKey = parameter.getParameterName();  
    final Object ob = comboBox.getSelectedItem();  
}
```

- optimiert für Arbeit mit Eclipse
- Quellcode im SVN: <http://sourceforge.net/projects/gvsigce/>
 - *sehr* umfangreiche API, teils in Spanisch kommentiert
 - schwierig zu erlernen, viele „Baustellen“
 - http://gvsigce.sourceforge.net/wiki/index.php/Development_and_releases/

Programmierung von gvSIG CE (1/2)



```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import org.gvsig.ce.*;
import org.gvsig.ce.parameters.*;
import org.gvsig.ce.parameters.additionalinfo.*;

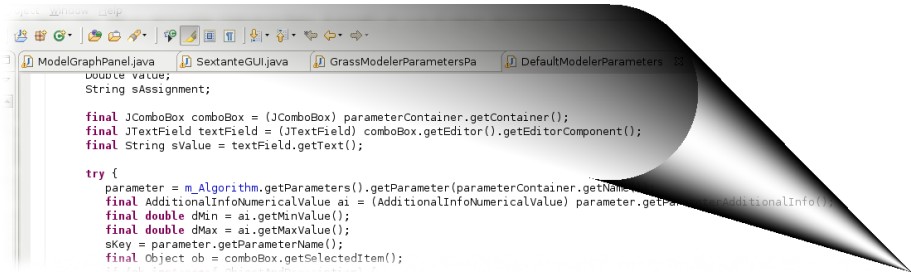
public class ModelGraphPanel {
    Double value;
    String sAssignment;

    final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();
    final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();
    final String sValue = textField.getText();

    try {
        parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getParameterName());
        final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getAdditionalInfo();
        final double dMin = ai.getMinValue();
        final double dMax = ai.getMaxValue();
        sKey = parameter.getParameterName();
        final Object ob = comboBox.getSelectedItem();
    }
}
```

- optimiert für Arbeit mit Eclipse
- Quellcode im SVN: <http://sourceforge.net/projects/gvsigce/>
- *sehr* umfangreiche API, teils in Spanisch kommentiert
- schwierig zu erlernen, viele „Baustellen“
- http://gvsigce.sourceforge.net/wiki/index.php/Development_and_releases/

Programmierung von gvSIG CE (1/2)



```
import java.util.*;
import java.awt.*;
import javax.swing.*;
import org.gvsig.ce.*;
import org.gvsig.ce.parameters.*;
import org.gvsig.ce.parameters.additionalinfo.*;

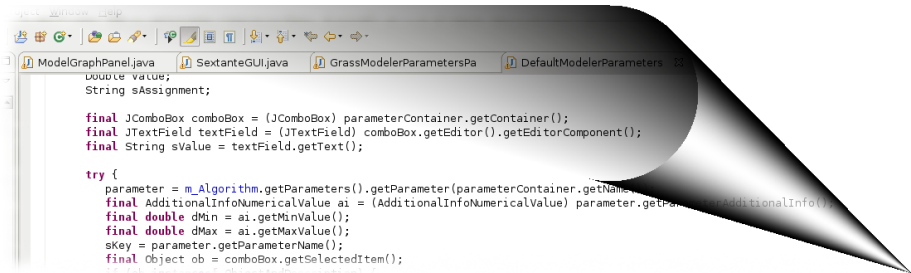
public class ModelGraphPanel {
    Double value;
    String sAssignment;

    final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();
    final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();
    final String sValue = textField.getText();

    try {
        parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getParameterName());
        final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getAdditionalInfo();
        final double dMin = ai.getMinValue();
        final double dMax = ai.getMaxValue();
        sKey = parameter.getParameterName();
        final Object ob = comboBox.getSelectedItem();
    }
}
```

- optimiert für Arbeit mit Eclipse
- Quellcode im SVN: <http://sourceforge.net/projects/gvsigce/>
- *sehr* umfangreiche API, teils in Spanisch kommentiert
- schwierig zu erlernen, viele „Baustellen“
- http://gvsigce.sourceforge.net/wiki/index.php/Development_and_releases/

Programmierung von gvSIG CE (1/2)



```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import org.gvsig.ce.*;
import org.gvsig.ce.parameters.*;
import org.gvsig.ce.parameters.additionalinfo.*;

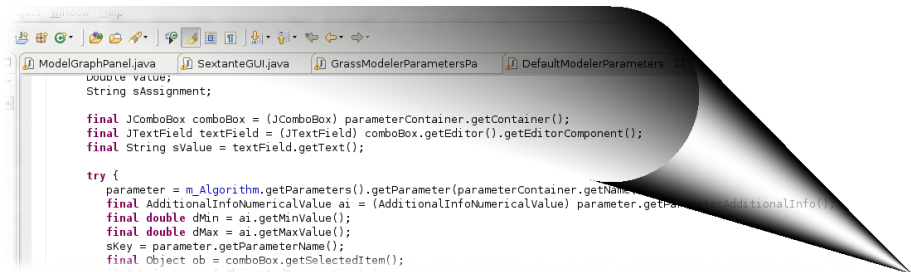
public class ModelGraphPanel {
    Double value;
    String sAssignment;

    final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();
    final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();
    final String sValue = textField.getText();

    try {
        parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getParameterName());
        final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getAdditionalInfo();
        final double dMin = ai.getMinValue();
        final double dMax = ai.getMaxValue();
        sKey = parameter.getParameterName();
        final Object ob = comboBox.getSelectedItem();
    }
}
```

- optimiert für Arbeit mit Eclipse
- Quellcode im SVN: <http://sourceforge.net/projects/gvsigce/>
- *sehr* umfangreiche API, teils in Spanisch kommentiert
- schwierig zu erlernen, viele „Baustellen“
- http://gvsigce.sourceforge.net/wiki/index.php/Development_and_releases/

Programmierung von gvSIG CE (2/2)



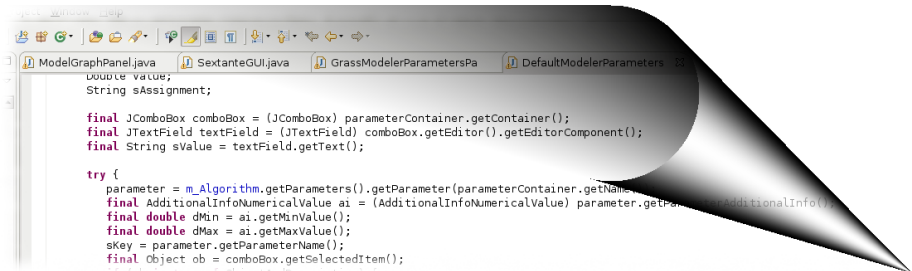
```
ModelGraphPanel.java | SextanteGUI.java | GrassModelerParametersPa | DefaultModelerParameters
Double value;
String sAssignment;

final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();
final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();
final String sValue = textField.getText();

try {
    parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getName());
    final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getAdditionalInfo();
    final double dMin = ai.getMinValue();
    final double dMax = ai.getMaxValue();
    sKey = parameter.getParameterName();
    final Object ob = comboBox.getSelectedItem();
}
```

- Python-Schnittstelle bildet Java-API ab; kaum dokumentiert
- Rasterdatentreiber in C/C++ (<http://www.gdal.org/>); per Java Native Interface (JNI)
- modular: Erweiterungen in separaten Ordnern unter „extensiones“
- Benutzeroberfläche über XML-Dateien anpassbar

Programmierung von gvSIG CE (2/2)



```
ModelGraphPanel.java | SextanteGUI.java | GrassModelerParametersPa | DefaultModelerParameters

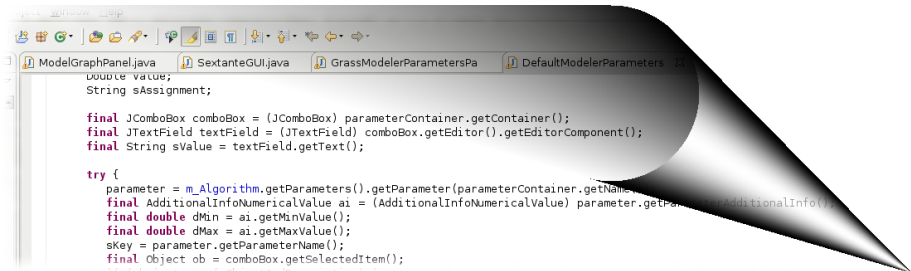
Double value;
String sAssignment;

final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();
final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();
final String sValue = textField.getText();

try {
    parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getName());
    final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getAdditionalInfo();
    final double dMin = ai.getMinValue();
    final double dMax = ai.getMaxValue();
    sKey = parameter.getParameterName();
    final Object ob = comboBox.getSelectedItem();
}
```

- Python-Schnittstelle bildet Java-API ab; kaum dokumentiert
- Rasterdatentreiber in C/C++ (<http://www.gdal.org/>); per Java Native Interface (JNI)
- modular: Erweiterungen in separaten Ordnern unter „extensiones“
- Benutzeroberfläche über XML-Dateien anpassbar

Programmierung von gvSIG CE (2/2)



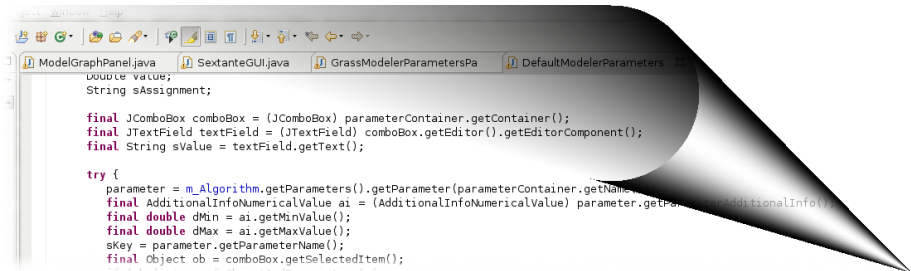
```
ModelGraphPanel.java | SextanteGUI.java | GrassModelerParametersPa | DefaultModelerParameters
Double value;
String sAssignment;

final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();
final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();
final String sValue = textField.getText();

try {
    parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getName());
    final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getAdditionalInfo();
    final double dMin = ai.getMinValue();
    final double dMax = ai.getMaxValue();
    sKey = parameter.getParameterName();
    final Object ob = comboBox.getSelectedItem();
}
```

- Python-Schnittstelle bildet Java-API ab; kaum dokumentiert
- Rasterdatentreiber in C/C++ (<http://www.gdal.org/>); per Java Native Interface (JNI)
- modular: Erweiterungen in separaten Ordnern unter „extensiones“
- Benutzeroberfläche über XML-Dateien anpassbar

Programmierung von gvSIG CE (2/2)



```
ModelGraphPanel.java | SextanteGUI.java | GrassModelerParametersPa | DefaultModelerParameters
Double value;
String sAssignment;

final JComboBox comboBox = (JComboBox) parameterContainer.getContainer();
final JTextField textField = (JTextField) comboBox.getEditor().getEditorComponent();
final String sValue = textField.getText();

try {
    parameter = m_Algorithm.getParameters().getParameter(parameterContainer.getName());
    final AdditionalInfoNumericalValue ai = (AdditionalInfoNumericalValue) parameter.getParameterAdditionalInfo();
    final double dMin = ai.getMinValue();
    final double dMax = ai.getMaxValue();
    sKey = parameter.getParameterName();
    final Object ob = comboBox.getSelectedItem();
}
```

- Python-Schnittstelle bildet Java-API ab; kaum dokumentiert
- Rasterdatentreiber in C/C++ (<http://www.gdal.org/>); per Java Native Interface (JNI)
- modular: Erweiterungen in separaten Ordnern unter „extensiones“
- Benutzeroberfläche über XML-Dateien anpassbar



- sehr einfache, gut dokumentierte Java-API
- einfache Automatisierung: Scripts (?) und *Modeler*
- mitgelieferte Module als Anschauungsmaterial
- Problem: Speichermanagement bei großen Eingabedaten



- sehr einfache, gut dokumentierte Java-API
- einfache Automatisierung: Scripts (?) und *Modeler*
- mitgelieferte Module als Anschauungsmaterial
- Problem: Speichermanagement bei großen Eingabedaten



- sehr einfache, gut dokumentierte Java-API
- einfache Automatisierung: Scripts (?) und *Modeler*
- mitgelieferte Module als Anschauungsmaterial
- Problem: Speichermanagement bei großen Eingabedaten



- sehr einfache, gut dokumentierte Java-API
- einfache Automatisierung: Scripts (?) und *Modeler*
- mitgelieferte Module als Anschauungsmaterial
- Problem: Speichermanagement bei großen Eingabedaten



GRASS GIS

The world's leading Free GIS software



- effiziente, gut dokumentierte C-API (derzeit GRASS 6.4.3)
- stabil; ideal für sehr große Rasterdatenmengen
- C-Quellcode muss *passend* kompiliert werden (Betriebssystem, GRASS Version, weitere Bibliotheken, etc.)
- Alternative: direkte Automatisierung mit Unix Shell Scripts/MSYS
- *Achtung: topologisches Vektordatenmodell*

Programmierung für GRASS-Backend



GRASS GIS

The world's leading Free GIS software



- effiziente, gut dokumentierte C-API (derzeit GRASS 6.4.3)
- stabil; ideal für sehr große Rasterdatenmengen
- C-Quellcode muss *passend* kompiliert werden (Betriebssystem, GRASS Version, weitere Bibliotheken, etc.)
- Alternative: direkte Automatisierung mit Unix Shell Scripts/MSYS
- *Achtung: topologisches Vektordatenmodell*

Programmierung für GRASS-Backend



GRASS GIS

The world's leading Free GIS software



- effiziente, gut dokumentierte C-API (derzeit GRASS 6.4.3)
- stabil; ideal für sehr große Rasterdatenmengen
- C-Quellcode muss *passend* kompiliert werden (Betriebssystem, GRASS Version, weitere Bibliotheken, etc.)
- Alternative: direkte Automatisierung mit Unix Shell Scripts/MSYS
- *Achtung: topologisches Vektordatenmodell*

Programmierung für GRASS-Backend



GRASS GIS

The world's leading Free GIS software



- effiziente, gut dokumentierte C-API (derzeit GRASS 6.4.3)
- stabil; ideal für sehr große Rasterdatenmengen
- C-Quellcode muss *passend* kompiliert werden (Betriebssystem, GRASS Version, weitere Bibliotheken, etc.)
- Alternative: direkte Automatisierung mit Unix Shell Scripts/MSYS
- *Achtung: topologisches Vektordatenmodell*

Programmierung für GRASS-Backend



GRASS GIS

The world's leading Free GIS software



- effiziente, gut dokumentierte C-API (derzeit GRASS 6.4.3)
- stabil; ideal für sehr große Rasterdatenmengen
- C-Quellcode muss *passend* kompiliert werden (Betriebssystem, GRASS Version, weitere Bibliotheken, etc.)
- Alternative: direkte Automatisierung mit Unix Shell Scripts/MSYS
- *Achtung: topologisches Vektordatenmodell*

Implementation weiterer Backends in SEXTANTE



- alles, was sich von der Kommandozeile steuern lässt, ist geeignet
- derzeit: GRASS, SAGA (C++), R (Skripte)
- A: neues Backend in SEXTANTE implementieren oder
- B: als GRASS-Skript verpacken (Overhead beachten)

Implementation weiterer Backends in SEXTANTE



- alles, was sich von der Kommandozeile steuern lässt, ist geeignet
- derzeit: GRASS, SAGA (C++), R (Skripte)
- A: neues Backend in SEXTANTE implementieren oder
- B: als GRASS-Skript verpacken (Overhead beachten)

Implementation weiterer Backends in SEXTANTE



- alles, was sich von der Kommandozeile steuern lässt, ist geeignet
- derzeit: GRASS, SAGA (C++), R (Skripte)
- A: neues Backend in SEXTANTE implementieren oder
- B: als GRASS-Skript verpacken (Overhead beachten)

Implementation weiterer Backends in SEXTANTE



- alles, was sich von der Kommandozeile steuern lässt, ist geeignet
- derzeit: GRASS, SAGA (C++), R (Skripte)
- A: neues Backend in SEXTANTE implementieren oder
- B: als GRASS-Skript verpacken (Overhead beachten)

Welche Schnittstelle(n) wählen?



- interaktive Werkzeuge: gvSIG (Java API)
- nicht-topologische Vektoroperationen: SEXTANTE (Java API)
- große Rasterdatenmengen: GRASS (C) oder SAGA (C++)
- Prototypen, Automatisierung: GRASS (Shell Scripts), SEXTANTE (Modeler)
- (statistische Funktionen: R-Skripte)
- andere Anforderungen (Lizenzen): neues SEXTANTE-Backend

Welche Schnittstelle(n) wählen?



- interaktive Werkzeuge: gvSIG (Java API)
- nicht-topologische Vektoroperationen: SEXTANTE (Java API)
- große Rasterdatenmengen: GRASS (C) oder SAGA (C++)
- Prototypen, Automatisierung: GRASS (Shell Scripts), SEXTANTE (Modeler)
- (statistische Funktionen: R-Skripte)
- andere Anforderungen (Lizenzen): neues SEXTANTE-Backend

Welche Schnittstelle(n) wählen?



- interaktive Werkzeuge: gvSIG (Java API)
- nicht-topologische Vektoroperationen: SEXTANTE (Java API)
- große Rasterdatenmengen: GRASS (C) oder SAGA (C++)
- Prototypen, Automatisierung: GRASS (Shell Scripts), SEXTANTE (Modeler)
- (statistische Funktionen: R-Skripte)
- andere Anforderungen (Lizenzen): neues SEXTANTE-Backend

Welche Schnittstelle(n) wählen?



- interaktive Werkzeuge: gvSIG (Java API)
- nicht-topologische Vektoroperationen: SEXTANTE (Java API)
- große Rasterdatenmengen: GRASS (C) oder SAGA (C++)
- Prototypen, Automatisierung: GRASS (Shell Scripts), SEXTANTE (Modeler)
- (statistische Funktionen: R-Skripte)
- andere Anforderungen (Lizenzen): neues SEXTANTE-Backend

Welche Schnittstelle(n) wählen?



- interaktive Werkzeuge: gvSIG (Java API)
- nicht-topologische Vektoroperationen: SEXTANTE (Java API)
- große Rasterdatenmengen: GRASS (C) oder SAGA (C++)
- Prototypen, Automatisierung: GRASS (Shell Scripts), SEXTANTE (Modeler)
- (statistische Funktionen: R-Skripte)
- andere Anforderungen (Lizenzen): neues SEXTANTE-Backend

Welche Schnittstelle(n) wählen?



- interaktive Werkzeuge: gvSIG (Java API)
- nicht-topologische Vektoroperationen: SEXTANTE (Java API)
- große Rasterdatenmengen: GRASS (C) oder SAGA (C++)
- Prototypen, Automatisierung: GRASS (Shell Scripts), SEXTANTE (Modeler)
- (statistische Funktionen: R-Skripte)
- andere Anforderungen (Lizenzen): neues SEXTANTE-Backend

„Wachstum durch gemeinsame Investition.“



<http://www.gvsigce.org/>