

# User-driven Manipulation of Geospatial Data

## DIPLOMARBEIT

zur Erlangung des akademischen Grades

## Diplom-Ingenieur

im Rahmen des Studiums

## Medieninformatik

eingereicht von

**Florian Stabel**

Matrikelnummer 0306331

an der  
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller  
Mitwirkung: Privatdoz. Dipl.-Geograf. Dr. Annett Bartsch

Wien, 24. Oktober 2012

\_\_\_\_\_  
(Unterschrift Verfasser)

\_\_\_\_\_  
(Unterschrift Betreuung)



# User-driven Manipulation of Geospatial Data

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Media Informatics**

by

**Florian Stabel**

Registration Number 0306331

to the Faculty of Informatics  
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Assistance: Privatdoz. Dipl.-Geograf. Dr. Annett Bartsch

Vienna, 24. Oktober 2012

\_\_\_\_\_  
(Signature of Author)

\_\_\_\_\_  
(Signature of Advisor)





# Erklärung zur Verfassung der Arbeit

Florian Stabel  
Müllerviertel 23, 8051 Graz

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

---

(Ort, Datum)

---

(Unterschrift Verfasser)



# Danksagung

Als erstes möchte ich mich bei Herrn Prof. Eduard Gröller für seine Ratschläge und das Korrekturlesen meiner Arbeit bedanken. Zudem hat er mich motiviert, meine Arbeit in Englisch zu verfassen.

Mein größter Dank gilt meiner Familie, die mich während meines Studiums stets unterstützt und selten an mir gezweifelt hat. :)

Außerdem will ich an dieser Stelle meiner Verlobten Domenica sowie unserer gemeinsamen Tochter Zoe danken, die während meiner Diplomarbeit außerordentlich aufopfernd waren! Ich liebe euch!



# Abstract

A WebGIS is an application designed to store, analyze, manage and present geographical data. It provides tools which support users to explore and understand the underlying data. Depending on the datasource (raster or vector data), these tools are varying. Many more styling options for vector data than for raster data exist. This is because of the position information of vector data which allows the system to apply a wider range of styling features.

This thesis is about user-driven manipulation of geospatial data and is part of the ESA DUE Permafrost project. It should help users to navigate through remote sensed data and to explore relevant features of it. Therefore it introduces methods to change the visualization of the data within a WebGIS by assigning different style profiles at runtime. The reference implementation is based on a specific software arrangement. The data is managed by GeoServer, an open source software server with the capability to edit and share geospatial data. To get a visual representation, GeoServer applies a Styled Layer Descriptor (SLD) on the data. A SLD is an XML based language and describes the appearance of associated layers. Modifying the SLD enables to influence the layer appearance. Since the data of the ESA DUE Permafrost project is of type raster only, this work concentrates on tools for styling raster data.

As a result of this work, an interactive legend is introduced. It is a user interface and manipulates the SLD at runtime according to the user settings. This gives the user the capability to highlight areas of interest based on the underlying data. The user interface also acts as a color legend for the displayed data.



# Kurzfassung

Ein WebGIS ist eine Applikation, die dazu dient, geographische Daten zu speichern, zu analysieren, zu verwalten und zu präsentieren. Um die zugrunde liegenden Daten zu verstehen, bietet ein WebGIS unterschiedliche Hilfsmittel an. Diese unterscheiden sich zudem in der Art der Datenquelle (Raster- oder Vektordaten). Vektordaten haben zusätzlich Positionsinformationen gespeichert, die es dem System ermöglichen, eine größer Auswahl an Stiloptionen anzubieten.

Diese Arbeit handelt über benutzergesteuerte Manipulation von Permafrost-Daten und ist Teil des ESA DUE Permafrost-Projektes. Der Zweck dieser Arbeit ist es, Benutzer in der Navigation von Fernerkundungsdaten und deren Untersuchung von relevanten Merkmalen zu unterstützen. Dabei werden Methoden vorgestellt, die die Darstellung von Daten in einem WebGIS verändern können. Dies wird ermöglicht, indem Stilprofile zur Laufzeit auf die jeweiligen Daten angewendet werden. Die Referenz-Implementierung basiert auf einer vorgegeben Software-Architektur. Zur Verwaltung der geographischen Daten wird ein Open Source Software-Server namens GeoServer verwendet. Um eine visuelle Repräsentation der Daten zu erhalten, wendet GeoServer einen sogenannten Styled-Layer-Descriptor (SLD) an. Ein SLD ist eine auf XML basierte Sprache, mittels der das Aussehen von verknüpften Datenschichten beeinflusst werden kann. Da sämtliche Daten des ESA DUE Permafrost-Projektes Rasterdaten sind, konzentriert sich diese Arbeit auf diese.

Als Ergebnis dieser Arbeit wird eine interaktive Legende vorgestellt. Dieses Benutzer-Interface manipuliert einen SLD zur Laufzeit, abhängig von den Benutzereinstellungen und aktualisiert die Datenansicht. Somit ermöglicht die interaktive Legende dem Benutzer interessante Werte aus den Permafrost-Daten hervorzuheben.





# Contents

<b>List of Abbreviations</b>	<b>xi</b>
<b>List of Figures</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The ESA DUE Permafrost Project . . . . .	1
1.2 Problem Statement . . . . .	3
1.3 Aim of the Work . . . . .	3
1.4 Thesis Structure . . . . .	4
<b>2 Related Work</b>	<b>7</b>
<b>3 The Starting Position</b>	<b>11</b>
3.1 The Graphical-Communication Process . . . . .	11
3.2 The Permafrost WebGIS . . . . .	12
3.3 User-Control Elements . . . . .	15
<b>4 Extending the Permafrost WebGIS</b>	<b>21</b>
<b>5 Implementation</b>	<b>29</b>
5.1 The User Interface . . . . .	31
5.2 Style Files . . . . .	33
5.3 Color Legend . . . . .	37
5.4 Transforming Settings to User Styles . . . . .	41
<b>6 Results</b>	<b>47</b>
6.1 Visualization Type Range . . . . .	47
6.2 Visualization Type Classification . . . . .	48
6.3 Using different Colormaps . . . . .	52
6.4 Highlighting Areas without Data . . . . .	52
6.5 Overlapping Layers . . . . .	58
<b>7 Conclusion</b>	<b>61</b>
	ix

<b>A Remote-Sensing Products</b>	<b>63</b>
<b>Bibliography</b>	<b>67</b>

# List of Abbreviations

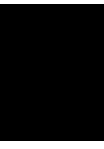
<b>AJAX</b>	Asynchronous JavaScript and XML
<b>API</b>	Application Programming Interface
<b>CLUT</b>	Color Look-up Table
<b>CSS</b>	Cascading Style Sheet
<b>CRS</b>	Coordinate Reference System
<b>DEM</b>	Digital Elevation Model
<b>DUE</b>	Data User Element
<b>DOY</b>	Day Of Year
<b>ECV</b>	Essential Climate Variable
<b>GIS</b>	Geographic Information System
<b>GML</b>	Geography Markup Language
<b>HEX</b>	Hexadecimal
<b>HTML</b>	Hypertext Markup Language
<b>JSON</b>	JavaScript Object Notation
<b>LST</b>	Land Surface Temperature
<b>MVC</b>	Model View Controller
<b>OGC</b>	Open Geospatial Consortium
<b>POI</b>	Point of Interest
<b>REST</b>	Representational State Transfer
<b>SLD</b>	Styled Layer Descriptor

<b>SSM</b>	Surface Soil Moisture
<b>SRS</b>	Spatial Reference System
<b>UI</b>	User Interface
<b>URL</b>	Uniform Resource Locator
<b>WebGIS</b>	Web Geographic Information System
<b>WMS</b>	Web Map Service
<b>XML</b>	Extensible Markup Language
<b>XSD</b>	XML Schema Definition

# List of Figures

1.1	Map of the Arctic and adjacent countries. . . . .	5
2.1	The WMS request contains the SLD assigned as a parameter. . . . .	8
3.1	The graphical-communication process. . . . .	11
3.2	Graphic variables after Jacques Bertin. . . . .	13
3.3	Software components of the Permafrost WebGIS. . . . .	14
3.4	Screenshot of the Permafrost WebGIS and its components. . . . .	15
3.5	Two different colormaps which can be applied to the data. . . . .	16
3.6	Two different colormaps applied to the same map section (around Mount McKinley, Alaska) create different visualizations. . . . .	17
3.7	Visualization of the same map section as in Figure 3.6 but of different visualization type. . . . .	18
3.8	Example of a color legend of the visualization type <i>range</i> . . . . .	20
3.9	Example of a color legend of the visualization type <i>classification</i> . . . . .	20
4.1	Communication between the software components of the Permafrost WebGIS. . . .	22
4.2	Different user styles create different visualizations without affecting each other. . .	24
4.3	CLUT created from a colormap image. . . . .	26
5.1	Schematic workflow of adding a layer to the viewport. . . . .	30
5.2	User interface of the reference implementation including the color legend. . . . .	31
5.3	The UI is integrated into the layer-styling process. . . . .	34
5.4	Color legend to SLD: outer classes . . . . .	42
5.5	Color legend to SLD: visualization type <i>range</i> . . . . .	43
5.6	Color legend to SLD: no stretching . . . . .	44
5.7	Color legend to SLD: visualization type <i>classification</i> . . . . .	44
5.8	Color legend to SLD: areas without data . . . . .	45
6.1	UI settings and output: visualization type <i>range</i> . . . . .	49
6.2	UI settings and output: outer classes. . . . .	50
6.3	UI settings and output: colormap stretching disabled. . . . .	51
6.4	UI settings and output: visualization type <i>classification</i> . . . . .	53
6.5	UI settings and output: different class colors. . . . .	54

6.6	UI settings and output: many classes. . . . .	55
6.7	UI settings and output: using a rainbow colormap. . . . .	56
6.8	UI settings and output: highlighting areas with no data. . . . .	57
6.9	Color legends of the Digital Elevation Model and the Surface Soil Moisture. . . . .	58
6.10	The layer of the SSM overlaps the layer of the DEM. The political border is on top of both layers. . . . .	59
6.11	The opacity of the layer SSM is set to the value 0.5. The layer of the DEM is shining through. . . . .	59



# Introduction

This thesis is about user-driven manipulation of geospatial data. To be more specific it covers a part of the ESA DUE Permafrost project. It should help users to navigate through remote sensed data and to explore relevant features of it. The work introduces methods to change the visualization of the data within a Web Geographic Information System (WebGIS) by assigning different style profiles. The reference implementation is based on a specific software arrangement which is described in Chapter 3.

## 1.1 The ESA DUE Permafrost Project

The ESA DUE Permafrost project was initially launched in June 2009 by the European Space Agency (ESA). The Institute of Photogrammetry and Remote Sensing (I.P.F), Vienna University of Technology, is the principal investigator and leads a consortium of four partners (Gamma Remote Sensing, University of Waterloo, Jena University, Alfred Wegener Institute for Polar and Marine Research) which support the project by preprocessing and providing satellite data among other things. The ESA DUE Permafrost project is funded within the ESA Data User Element program.

### Permafrost

Permafrost is defined as soil which is frozen and its temperature is continuously under 0 °C (32 °F) for at least two years. Its state is an Essential Climate Variable (ECV) [8] and it is worthwhile to be observed. The ESA DUE Permafrost project aims to install a monitoring system tracking permafrost soil characteristics over time. This work is supposed to support this aspect. A complete list of all objectives is listed next (quoted from the ESA DUE Permafrost project [3, 4]).

1. Define earth observation-based services for permafrost monitoring depending on the user requirements.

2. Integrate the latest earth-observation technology with state of the art ground based measurements.
3. Demonstrate and validate the services with the user organizations.
4. Develop mid to long term scenarios for boreal and arctic permafrost monitoring.
5. Contribute to new scientific results in the domain of climate-change detection, climate modeling and hydrological modeling.

### **Remote-Sensing Products**

Permafrost is a subsurface phenomenon. That is why satellites can not directly observe its state. The ESA DUE Permafrost project has defined relevant satellite-observable parameters which provide a reliable statement about permafrost soil. These parameters are called Remote-Sensing Products:

- Digital Elevation Model (DEM)
- Land Surface Temperature (LST)
- Surface Soil Moisture (SSM)
- Day of Freeze/Thaw
- Subsidence
- Coherence

These Remote-Sensing Products are very important for this work, since each has its own characteristic concerning its visual representation. Each Remote-Sensing Product has its own color schema according to the meaning of the data. For instance, Land Surface Temperature goes from blue (cold) to red (warm). Although “cold” and “warm” is relative since permafrost is frozen soil. The ESA DUE Permafrost project focuses on areas around the Arctic and the Arctic itself (see Figure 1.1):

- Alaska
- Central Yakutia
- Laptev Sea
- Mount McKenzie
- Ob Estuary



## 1.2 Problem Statement

In order to be able to explore and analyze the permafrost data of the different regions, a WebGIS is needed. It manages the permafrost data and enables the user to monitor climate changes and the exploration of the geospatial data. A large amount of geospatial data in form of raster data is produced by partners of the ESA DUE Permafrost project. Raster data is a matrix which consists of cells organized into rows and columns. Each cell contains geographic information in a numerical form. This requires reliable software which is able to manage the data and provide an interface to gain insight. GeoServer has been chosen, which provides a rich Web Geographical Information System (WebGIS). A WebGIS is a popular system due to the public accessibility and quick overview over geographical data. Adding layers, displaying points of interests, setting different views or overlapping layers are just some standard features. Deep data analysis is often not supported. Since the user should get insight into the permafrost data, this thesis provides a solution.

The ESA DUE Permafrost project is supposed to deliver meaningful visual representations of the permafrost data. GeoServer does not provide any features for explorative research of raster data. This thesis tries to accomplish this task. The main requirements in combination with the existing software are:

- The system does not support tools which allow the user to manipulate the layer presentation. The task of this work is to find a way to influence the layer representation in real-time.
- Introduce user styles so that each registered user can adapt the visualization of each layer, without affecting the visualization of other users.
- Develop useful tools for user-driven manipulation of the layer presentation of the permafrost data.
- Introduce a user interface for users to adapt the output and include identified tools.
- The system uses a built-in legend explaining the colored data. Depending on the visualization settings, a static legend is generated. Replacing GeoServer's built-in legend is another task.

## 1.3 Aim of the Work

This thesis proposes an approach for manipulating the visualization of geospatial data in real-time. The current implementation of GeoServer does not offer any method to change the appearance of raster data without the background knowledge of the styling language. Based on the Styled Layer Descriptor (SLD) specification [13] and the characteristics of the permafrost data, a set of tools is identified. With these tools the user should be able to manipulate the layer visualization in real-time. The visualizations of other users should not be affected.

A user interface (UI) for controlling the layer appearance is designed. The identified tools are transformed to proper UI control-elements. According to a traditional static color legend,

the new user interface should also act as a color legend. With the mixture of a color legend and tools to adapt the data presentation, the exploration process should be simplified.

## 1.4 Thesis Structure

Chapter 2 deals with the related work covering the same topic as this thesis. Past and ongoing work dealing with a similar problem are discussed, compared and summarized. Advantages and disadvantages of each approach are listed and discussed relating to the approach of the current work.

The starting position is explained in Chapter 3. Since this thesis is part of the ESA DUE Permafrost project, the prerequisites are explained. It lists the given software components where the reference implementation is embedded. Also the requirements of the user interface are explained in detail.

Chapter 4 explains the modifications of the Permafrost WebGIS. A common user workflow is illustrated and described. It explains the communication between the involved components. The process, how GeoServer uses styles, is explained next. It shows how layers are requested and different styles are applied. The structure of the XML-based style is also described here. After that, the role of the colormaps is explained.

The following Chapter 5 covers the challenging parts of the implementation. It describes why the reference implementation is developed as a GeoServer extension and how it can be invoked. Understanding the needs of the user styles and additional files is important. This Chapter explains how styles are maintained in GeoServer and how they can be manipulated. Based on the identified tools, the user interface is designed and implemented. Therefore JavaScript is needed extensively. Selected code fragments are explained in detail. The main tasks of each UI control-element are listed and the impact of the elements on the layer visualization are shown. Listed figures explain how settings are transformed to a valid XML-based user style.

Chapter 6 contains the results of the main scenarios and their resulting visual output. UI settings of the color legend are compared with the layer visualization and explained in detail.

In Chapter 7 the results and the knowledge of this work are explained.



Figure 1.1: This map shows the relevant regions of the ESA DUE Permafrost project. The red line contains the areas where the average temperature is below 10 °C. (Source: [1])



## Related Work

This Chapter gives an overview of the existing work related to the subject of this thesis. Since the reference implementation is about the styling functions of GeoServer, this Chapter lists similar approaches. Existing work which is related to the subject of this thesis is rare. This is because this work covers a given software and deals with continuous data of satellites. The Open Geospatial Consortium (OGC) defines open standards which are supported by GeoServer and many other geospatial applications. Styled Layer Descriptor (SLD) is the main instrument for adapting the visual representation of the geospatial data in OGC compliant applications. It is a XML schema and describes the appearance of a layer in a Geographic Information System (GIS). This allows user-defined symbolization and coloring of geographic features. A GIS has a wide field of possibilities to visualize geospatial data [12]. Points, lines, areas and alphanumeric characters are the basic elements and do provide graphic variables like size, intensity, form, texture, direction and color [6]. Depending on the datasource (raster- or vector data), different graphic variables can be applied on the data.

The master thesis of Albrecht Weiser gives a solution how to do automated generation of XML-based SLD files [23]. It aims to convert proprietary ESRI ArcGIS maps to the open OGC standard SLD. The work processes existing maps and generates a valid SLD. The algorithm runs through all symbols of the ArcGIS map and analyzes its features. In case the SLD specification provides a proper substitute for the analyzed symbol, the program generates the corresponding SLD code. This algorithm is not directly applicable to this work since the SLD generation is not user-driven and it converts vector data only. But it helps to understand the structure and possibilities of the SLD and the resulting images.

Another work also covers the topic of interactive layer generation [10]. It describes how a layer can be requested via the existing public services of OGC compliant applications. This approach uses the public Web Map Service (WMS) which is used to request geospatial data in the form of images. The WMS is invoked via URL containing several parameters describing the datasource, the styling and the output format. Unlike the previous work, this work uses an existing SLD to request the visualization of the data. Figure 2.1 explains this approach. Every layer has a default SLD. Step 1) shows the URL of a WMS invocation. *SERVICE*, *Version* and

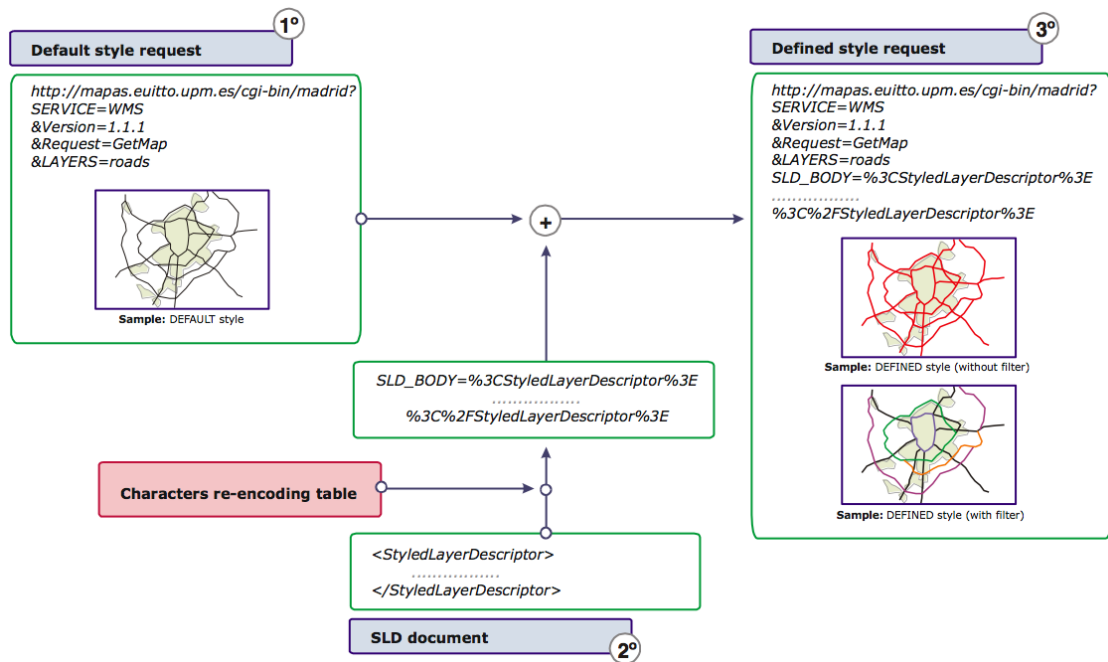


Figure 2.1: The SLD is encoded and assigned as the parameter `SLD_BODY` to the WMS request. It replaces the default SLD of the layer when the layer is rendered (source: [10]).

*Request* are parameters addressing the service. The parameter *LAYERS* addresses the ids of the requested layers. Without any styling parameter, the WMS will use the default SLD of each listed layer and renders the image which will be returned. The WMS provides parameters to add styling information to URL. The content of the parameter *SLD\_BODY* replaces the default-layer SLD if specified. In step 2) the content of an external SLD is encoded and assigned as the parameter *SLD\_BODY* to the request. The WMS applies the submitted SLD to the layer and returns the result (step 3). Since this approach appends the SLD to the URL, the length of the URL is increasing. If the URL is too long, the server has a problem in receiving the entire URL. This approach does not include a user-friendly interface to generate the SLD and assumes that the user builds the SLD manually. A lot of knowledge about the SLD specification is needed. Although this work concentrates on vector data, the WMS invocation is applicable on raster data too.

The most interesting approach is an implementation called *Styler* [15] which is part of the OpenGeo Suite. It provides an interface for styling layers by manipulating the corresponding SLD similar to the approach of this work. It is completely written in JavaScript and uses a standardized JavaScript library called GeoExt. The SLD (XML) is created by the client (JavaScript) and is sent to GeoServer afterwards. The work provides a user interface to manipulate the SLD. Similar to the previous work, the SLD is applied by sending an HTTP request (POST, GET, PUT or DELETE) to a GeoServer service. This has the advantage, that *Styler* can be used in

combination with any WMS compliant application. Styler can only handle vector data. This fact makes almost all features inapplicable to our requirements. Changing a style affects the visualization of all users. This means it does not support multi-user styles which is a fundamental requirement for this work.

All these implementations have different approaches how they request and submit styling information. They have two characteristics in common. They all support vector data only. This is regrettable since the ESA DUE Permafrost project provides raster data only. Furthermore, they are using GeoServer's public REST services and not its internal functions like an extension does. This work provides a user interface to adapt the visualization of raster data. It is implemented as a GeoServer extension and has access to its internal functions. All SLDs are managed by GeoServer and not externally.





## The Starting Position

At the beginning of this thesis not all requirements were clearly defined in detail. Primarily adapting the layer visualization in real-time by the user should be enabled. How and which tools should be provided was not defined at this point. The main software parts of the ESA DUE Permafrost project were already defined. The system was able to visualize the permafrost data. Because this thesis and its considerations are based on these software parts, it is important to understand their relationship and functionality.

### 3.1 The Graphical-Communication Process

Before a user can navigate through the geospatial data, the data is collected and preprocessed. A part of the reality is captured, transformed into a digital representation and transmitted via a medium. The result can be interpreted by the receiver. This process is called graphical communication (see Figure 3.1). Based on the real world, a *primary model* is created. Many simplifi-

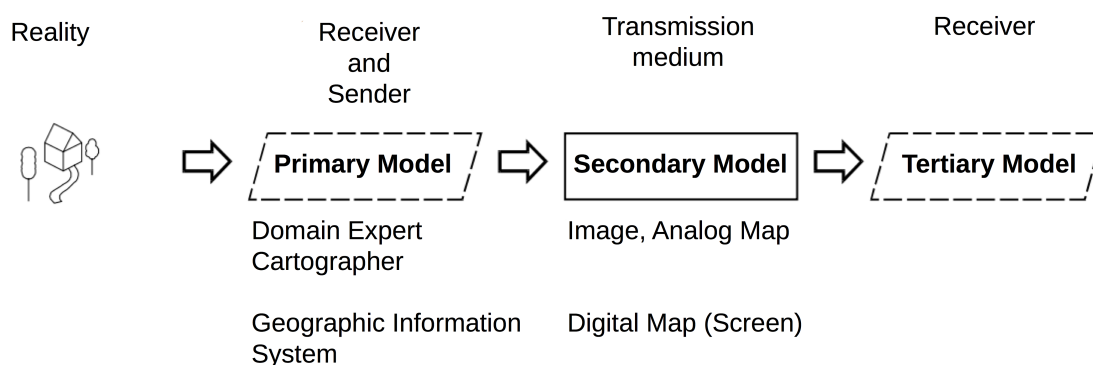


Figure 3.1: The graphical-communication process. (Source: [6], translated)

cations are done by omitting unimportant information. Which information is omitted depends on the purpose of the map. This process is done by domain experts or like in the ESA DUE Permafrost project, by data producers. The *secondary model* called graphical representation takes care of the visualization. The third step in the graphical-communication process is named *tertiary model* and is about the receiver. Based on the previous knowledge, the user interprets the visualization. Without knowing the meaning of the visualization, the user can not interpret the underlying information or misunderstands it. A legend helps the user to interpret the visualization.

Digital graphical representations have a big impact on the secondary and tertiary model. As symbols are information carriers, Jacques Bertin developed the *Graphical Semiology* [6] (see Figure 3.2). Bertin differentiates between four basic elements: points, lines, areas and alphanumeric characters. Each element has several graphic variables like size, brightness, form, texture, direction and color. Since the permafrost data lacks information about geometric objects, this thesis can deal with different colors and brightnesses only. On the following pages the term *layer* is used repeatedly. A layer is the visual representation of the underlying data. Layers are meant to be overlapping and displayed in the order they are stacked. The result of all merged layers is called map.

## 3.2 The Permafrost WebGIS

In order to make the permafrost data public, an application was set up named Permafrost Web Geographic Information System (WebGIS). A WebGIS is a system which allows to share and visualize geospatial data over the internet. People all over the world can access the data without special software. Simply a browser is needed to navigate through the data. Users can easily navigate over maps and can display additional information. This allows the user to explore the underlying data. To gain insight into the geospatial data, the following common features are very important for such systems.

- Zoom in/out to get details/overview.
- Switch between coordinate systems (WGS84 Lat/Long, EASE Grid North).
- Show/hide points of interest (POI).
- Overlap layers.
- Share maps with other users.
- Get additional information of features like sea level of a certain point.

The Permafrost WebGIS is about the *secondary model* according to the graphical-communication process. Graphic variables are applied to the permafrost data to get a visualization which is presented to the user. In order to ensure the requirements of the ESA DUE Permafrost project, the Permafrost WebGIS consists of several applications and libraries. Each software achieves different tasks and interacts with the other software components. Figure 3.3 shows the interactions between these components and acts as an overview image.

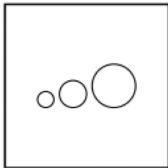

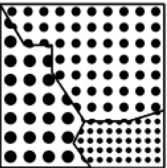
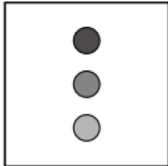


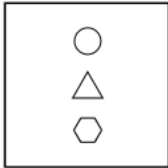
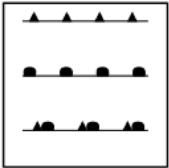

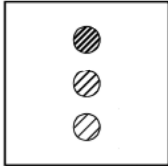




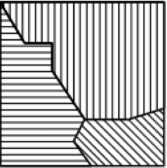
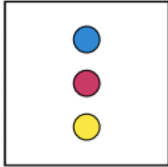


Graphic Variable	Point	Line	Area	Alphanumeric Characters
Size				small medium large
Intensity				bright medium dark
Shape				Arial Courier Lucida
Texture				Texture Texture Texture
Orientation				45° 0° 315°
Color				cyan magenta yellow

Figure 3.2: Graphic variables after Jacques Bertin. (Source: [6], translated)

## The Permafrost User-Interface

The Permafrost User-Interface (UI) is responsible for granting access to the permafrost data. It builds the framework of the Permafrost WebGIS and comprises all user-control elements. It controls which data should be displayed in the WebGIS and calls functions of other software components to adapt the visualization. Figure 3.4 shows a screenshot of the Permafrost WebGIS and its components. The Following operations can be controlled by the user directly via the UI:

- **Maps:** Switch between different regions with relevant permafrost data (Alaska, Central Yakutia, Laptev Sea, Mount McKenzie, Ob Estuary, Arctic).

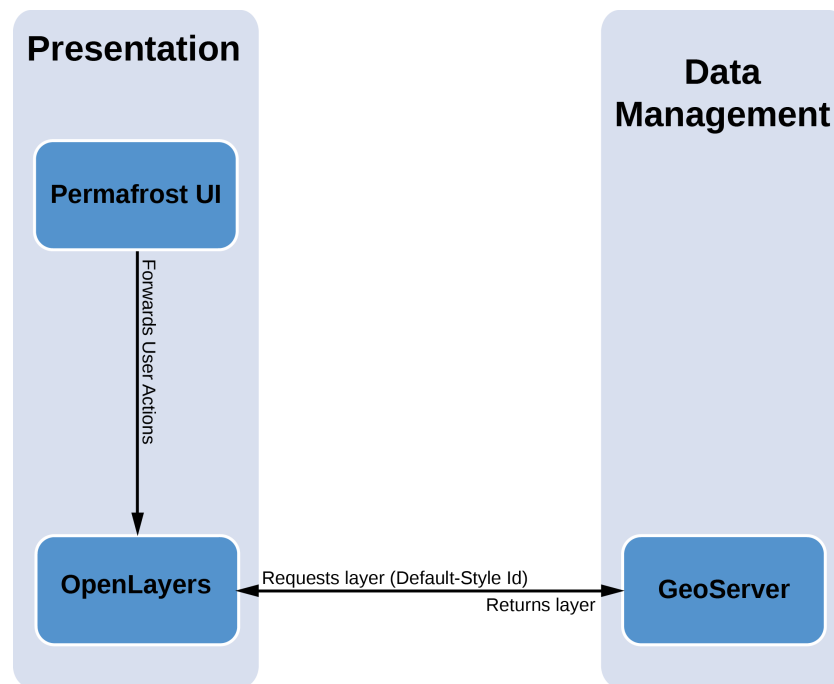


Figure 3.3: Software components of the Permafrost WebGIS.

- **Coordinate System:** Switch between two coordinate systems (WGS84 Lat/Long, EASE Grid North).
- **Active Layers:** Show or hide layers.
- **Year and Timescale:** Data of different time periods can be displayed.
- **Legend:** Show the color legend of a single layer.

## OpenLayers

OpenLayers is a JavaScript library which enables dynamic maps on any website. It provides the common control elements of a WebGIS as mentioned before. This library creates a viewport and arranges a digital map in form of an image inside. The map is the result of any number of superposed layers. A layer is the visual representation of the underlying data. The Permafrost UI invokes OpenLayers functions which affect the visualization inside the viewport. Every time the user changes settings via the provided Permafrost UI control-elements, OpenLayers requests a new layer visualization from the geospatial data server.

## GeoServer

GeoServer as the geospatial data server, manages the permafrost data of the ESA DUE Permafrost project. GeoServer is an open source software server with the capability to edit and

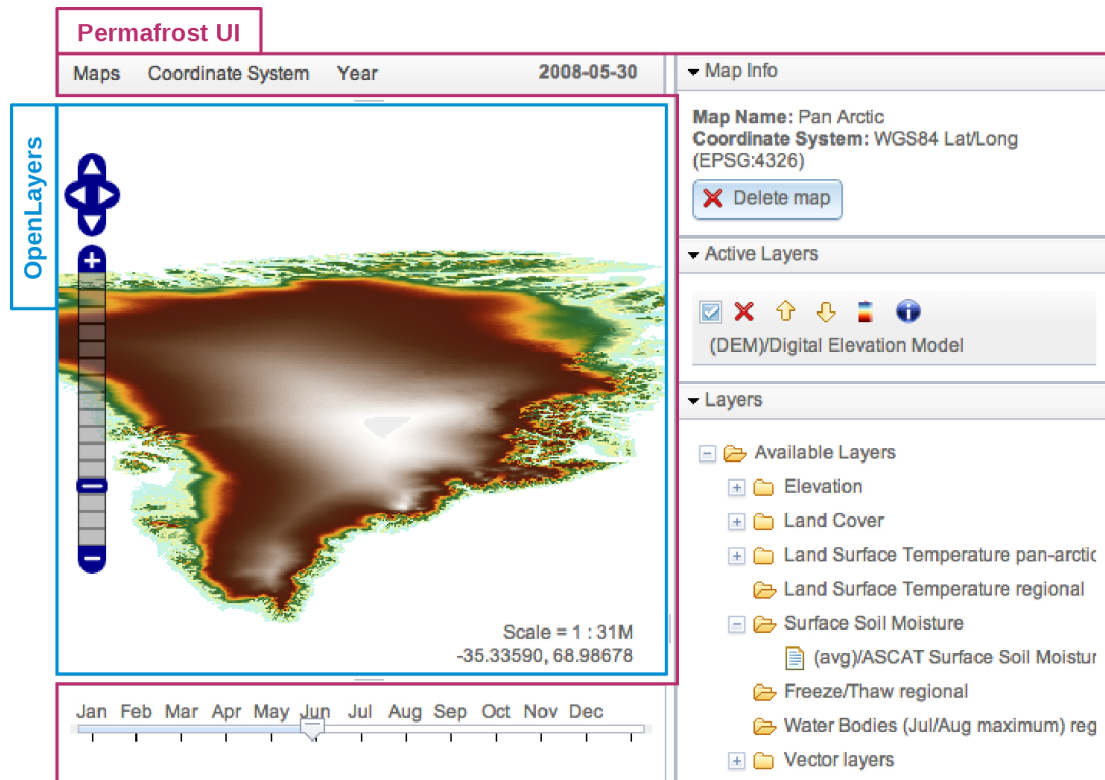


Figure 3.4: Screenshot of the Permafrost WebGIS and its components. GeoServer acts in the background and provides the data for the visualization.

share geospatial data. GeoServer can handle two types of data, vector and raster data. Vector data describes geometric objects with points, lines and polygons. The geometric objects correspond to map features such as borders of a country, streets, trails or buildings. Raster data describes a matrix of cells. These cells are also called pixels, squares, and grids and determine the detail that can be maintained in the dataset. Vector data are qualified for spatial details, while raster data are well suited for analyzing data that vary continuously from location to location such as elevation, temperature or soil moisture. Depending on the data format, GeoServer applies styles to the data to create an image of the requested permafrost data. A style is a set of rules which describe how to convert data features into a visual representation.

### 3.3 User-Control Elements

Now that the existing software components are explained, the requirements of this work can be described. This section covers the considerations based on the existing software components and introduces tools to adapt the layer visualization. Raster data limits the possibilities of the

visualization since it lacks information about geometric objects. Color and brightness as graphical variables, are the only way to visualize features of the permafrost data. This means that values are assigned to colors and rendered as an image afterwards. As means of expression, colors are essential in Geographical Information Systems and are characterized by the following properties.

- A color is a carrier of information by its own.
- A color simplifies and accelerates the transmission of information.
- The meaning of a certain color is associative.

If the color composition is done wrong, data can be misinterpreted or in the worst case not be interpretable. It is also important to provide an explanation of the color. This becomes even more important if people are suffering from abnormality concerning color perception. In general the meaning of colors differs in every culture and even more over time periods within the same culture. There is also a great diversity between domains. In every domain the intuitive and correct experience of colors should be created.

To customize the colors of the visualization, colormaps are introduced. Colormaps help the user to apply a certain color schema to the data. Figure 3.5 shows two colormaps which are used in this thesis. The colormap in Figure 3.5a is used for visualizing values of the Remote-Sensing Product Digital Elevation Model (DEM), while the colormap in Figure 3.5b is used for visualizing values of the Remote-Sensing Product Land Surface Temperature (LST). Figure 3.6 shows the results of the two colormaps using the area around Mount McKinley (Alaska).

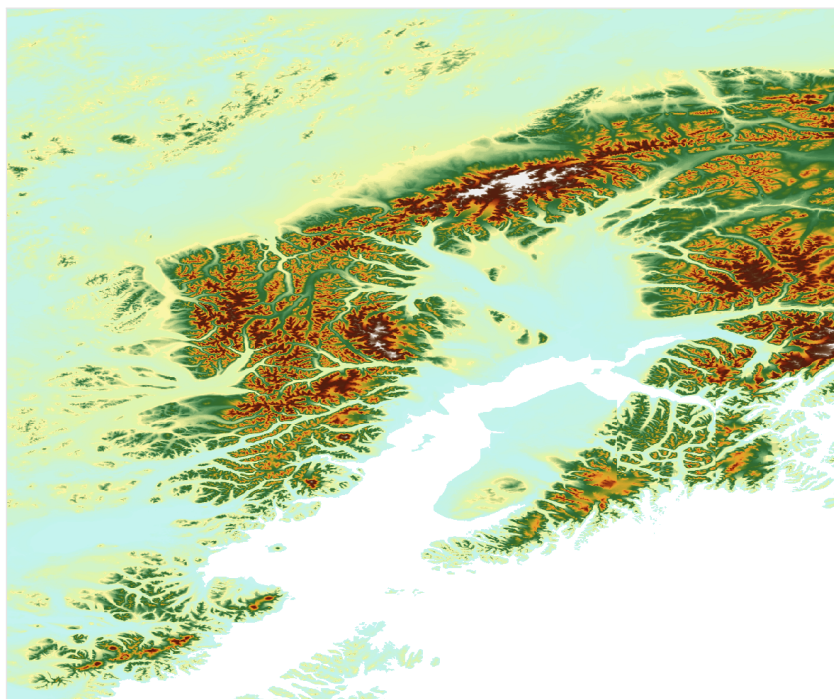


Figure 3.5: Two different colormaps which can be applied to the data.

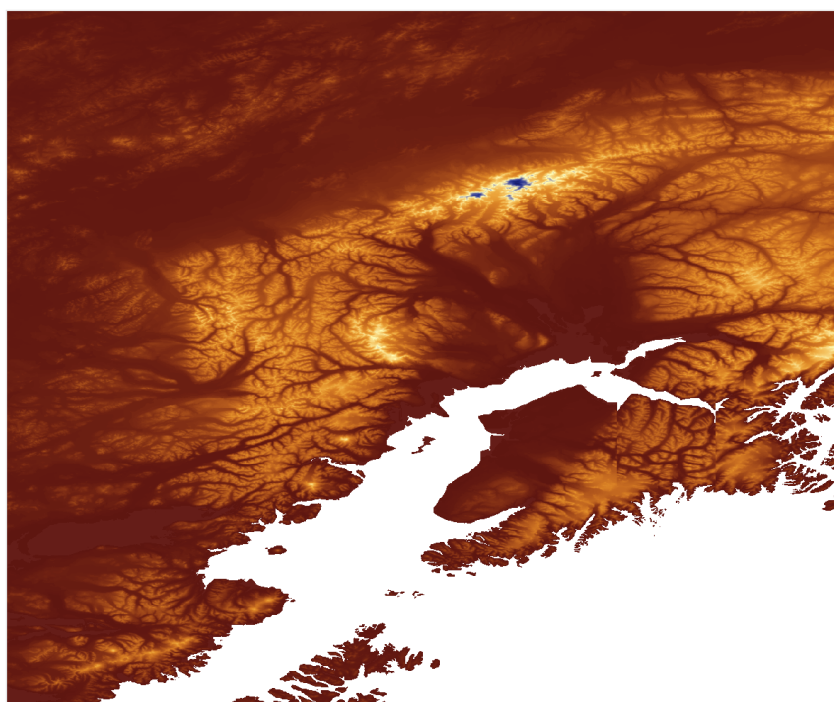
Colormaps help the user to interpret the visualization. In some cases it could be useful to apply a different colormap to the data which shows other details. To be able to select the values which are displayed, other tools are necessary. Therefore two main visualization types are introduced in order to select the values of the permafrost data which should be displayed:

**Range** is used for colorful images. Which means that a selected value range is applied on the selected colormap. The values within this range are mapped to the colormap. These values are visualized in the WebGIS only. This allows to specify a value range to be displayed. Values outside the selected value range are clipped or can be colored in a custom color.

**Classification** is used to get a better insight into the underlying data. This type allows the user to specify a number of value classes which are distributed over the whole value range. The user can determine which values are assigned to which class. Each class gets a different color depending on the colormap. All values of the value range are visualized in



(a) Visualization created with the colormap from Figure 3.5a.



(b) Visualization created with the colormap from Figure 3.5b.

Figure 3.6: Two different colormaps applied to the same map section (around Mount McKinley, Alaska) create different visualizations.



the WebGIS in the color of the assigned class. The color of each class can be changed manually to highlight specific areas.

Figure 3.6 shows an example created with the visualization type *range*. Figure 3.7 shows the visualization of the same map section but created with the visualization type *classification* using ten classes.

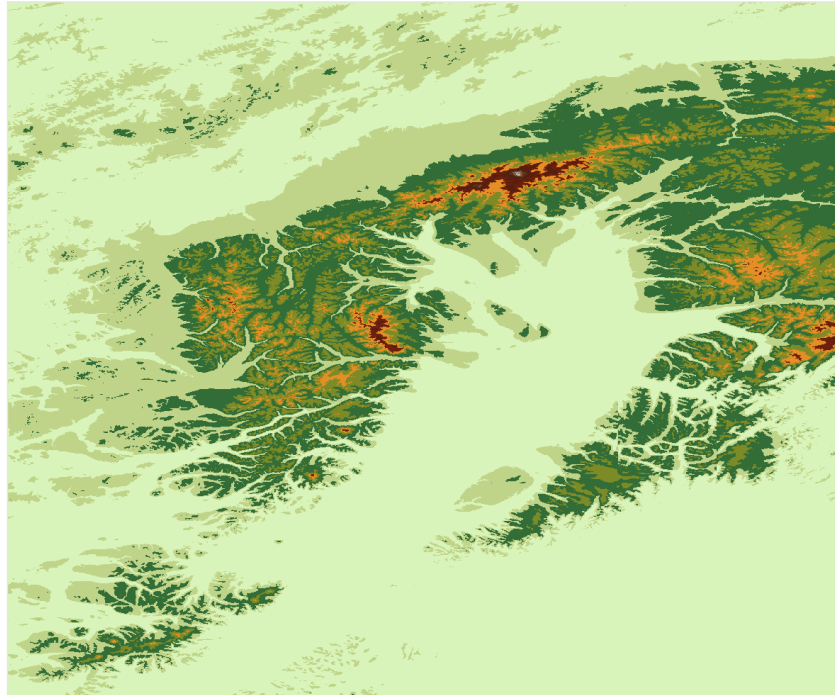


Figure 3.7: Visualization of the same map section and colormap as in Figure 3.6 but created with a *classification* of ten classes.

The user can choose whether to apply the visualization type *range* or *classification*. The terms *range* and *classification* are used in the sense of the visualization type only in this work. Besides the visualization types, more requirements are defined in Table 3.1. Some of them are applicable to both visualization types while some are just applicable to one visualization type.

### Color Legend

The color legend is the most interesting tool of this thesis. It replaces GeoServer's internal static color legend and is designed to be interactive. This means it contains requirements listed above which can be adjusted by the user. In order to be able to adjust the value range of each class, the color legend is implemented as a slider bar with multiple handles on it. Moving a handle, will reallocate the values to the adjacent classes. The bar represents the entire value range while the handles serve as class limiters. Handles can be set to any value within the value range, depending on a defined step width. This means, the user can not set a value outside the value range.



Range	Classification
Show values within a user set value range.	Define a custom number of classes by the user.
Highlight or hide data outside the selected value range.	Automatic colorization of selected classes.
Stretching or compressing resizes the selected colormap to the selected value range.	The user should be able to assign a custom color to any class.
The user should be able to apply different colormaps.	
Highlight areas where no data exists. This is very important to distinguish areas without data from areas with data.	
Unit conversion: examples are <i>Kelvin</i> $\Leftrightarrow$ <i>Fahrenheit</i> $\Leftrightarrow$ <i>Celsius Degree</i>	
Save settings for each user and style. Every time the user reopens the user interface, previous settings should be restored.	
Reset user modifications to presets. This restores the visualization if the user wants to reset his modifications.	

Table 3.1: Requirements on the reference implementation.

### Highlighting Areas of Interest

Users should be able to explore features of the geospatial data. The idea is that areas of interest can be examined by applying custom colors. A colorpicker is a simple way to let users choose a color without knowing color codes, e.g., hexadecimal codes. A colorpicker is a user interface to pick a color of a color schema [21].

Figure 3.8 shows an example of a color legend for the visualization type *range*. The selected value range is called visible class from now on. All values inside the visible class are mapped to the colors of the shown colormap. Sometimes values of the outer classes are of interest too. To highlight these outlying values in the visualization, the user can apply a custom color to values of *outer class 1* and *outer class 2*. Figure 3.9 shows an example of the color legend for the visualization type *classification*. For each class the user can apply a custom color to highlight the values in the visualization.

Remote sensing data producers of the ESA DUE Permafrost project do not always provide data for each area. This means that some areas are not considered in the remote sensing of the permafrost-soil data and defined no-data values are set. Depending on the project partner who was preprocessing the data, no-data values have different characteristics. To distinguish areas without data from the actual data, a custom color can also be defined for no-data values.

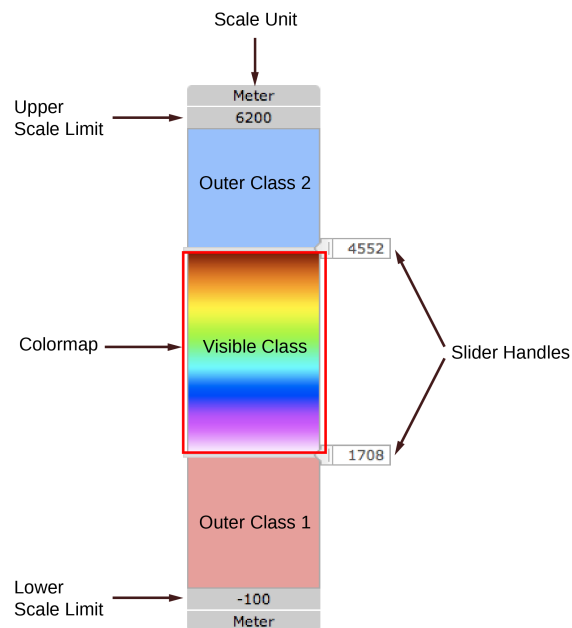


Figure 3.8: Example of a color legend of the visualization type *range*.

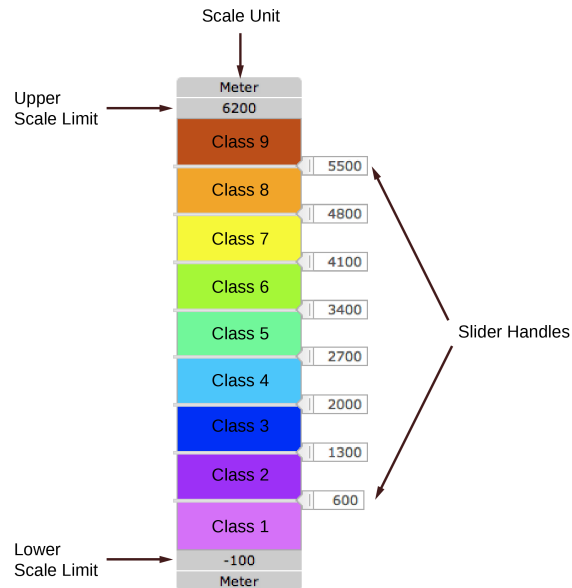
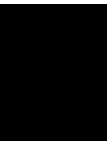


Figure 3.9: Example of a color legend of the visualization type *classification*.



## Extending the Permafrost WebGIS

In order to accomplish all requirements from Chapter 3, some modifications of the Permafrost WebGIS are necessary. These modifications are concerning the interaction between the software components and GeoServer. Figure 4.1 shows parts of the Permafrost WebGIS which are modified or implemented colored in red.

OpenLayers requests layer visualizations from GeoServer via the Web Map Service (WMS) and displays the visualization inside the viewport. The WMS is a service which is invoked over the internet. The response can be in various formats like image, text or html. Such services are called Representational State Transfer (REST). Depending on the parameters the WMS returns a visual representation of the requested data. The WMS as a part of GeoServer is important for this work, because it allows to request different appearances of the layers by changing the parameters.

The main part of the reference implementation is about extending GeoServer's functionality. The requirements of this work are implemented as a GeoServer extension and can be invoked from other components of the Permafrost WebGIS. GeoServer uses styles for creating visualizations of the data. The rules for mapping data values to graphic variables (points, lines, polygons or alphanumeric characters) are described in a style which is also called Styled Layer Descriptor (SLD). Styles are managed by GeoServer and have a unique id with which the style can be addressed by WMS. As an extension of GeoServer, the reference implementation has access to GeoServer's internal styling functions. Styles can be added, modified or deleted. All requirements identified in Chapter 3 are unified in one user interface (UI) which controls the layer visualization. Each UI control-element has an impact on the style. This means, if the state of the UI control-element changes, the style of the layer is modified accordingly. The id of the style is returned to the Permafrost UI.

Every time the Permafrost UI forwards user actions to OpenLayers, OpenLayers requests a new layer visualization. OpenLayers invokes the WMS and assigns parameters to address different properties like the style or width and height of the returning image. Since the WMS provides a lot of parameters, only the mandatory ones are listed in Table 4.1. A full list can be

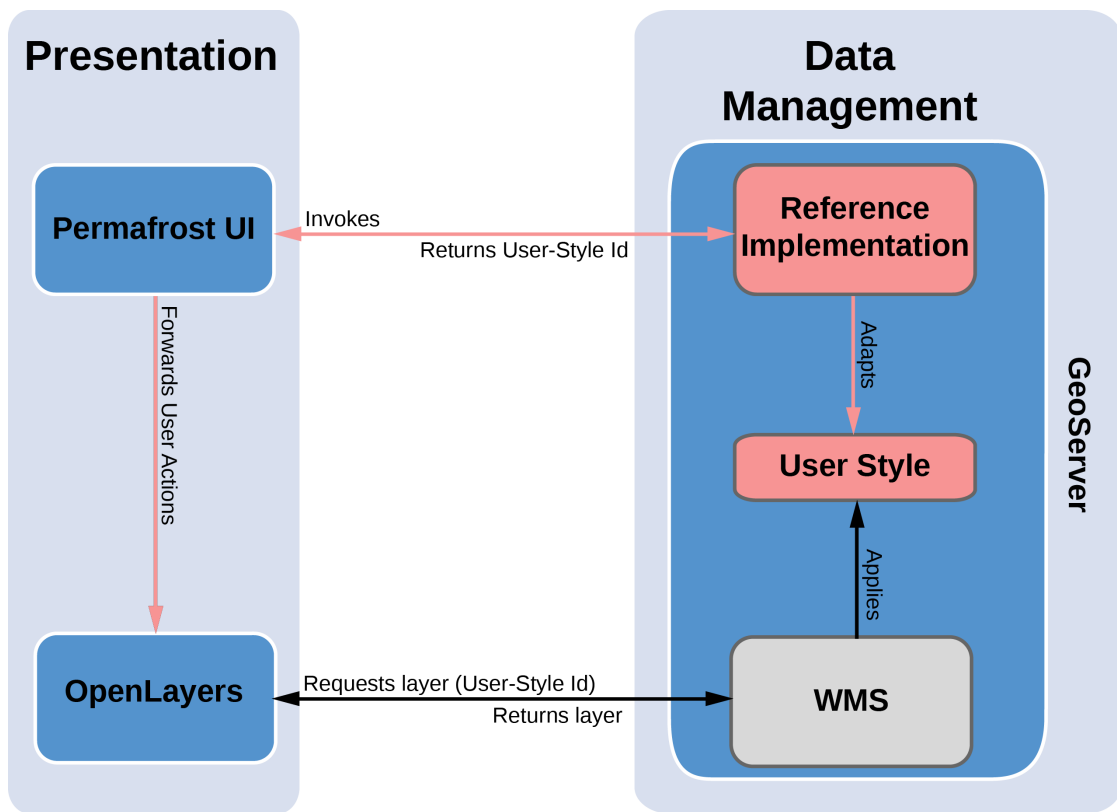


Figure 4.1: Communication between the software components of the Permafrost WebGIS. Modifications as a part of this work are colored in red.

looked up in the WMS specification [5]. The following listed parameters directly affect the layer visualization.

The parameter **STYLES** determines the styles which are applied to the layers specified by the parameter **LAYERS**. The number of assigned styles must comply with the number of assigned layers. The style plays a major role in this work, as it allows to manipulate the visual output. Thus the implementation focuses on methods to create and modify styles. The approach is to generate custom styles at runtime and assign them to the WMS to get the desired layer visualization. The parameter **STYLES** of the WMS request is changed on demand.

## User Styles

Each layer, stored in GeoServer, has a default style. Each style can be assigned as a default style to zero or more layers. If a style changes, the visualization of each linked layer changes too. One user can access the same layer and style at the same time. This means, if a user changes the style for his needs, the visualization of other users would change too.

This problem can be solved by introducing user styles. For every default style and user, a

Request Parameter	Description
VERSION=1.3.0	Determines the request version in respect to the service.
REQUEST=GetMap	Determines the type of the request. GetCapabilities, GetMap and GetFeatureInfo are available. GetMap determines that a visual representation of the given LAYERS is requested.
LAYERS=layer_list	Comma-separated list of the id of one or more requested map layers.
STYLES=style_list	Comma-separated list of the id of one or more styles which should be applied to the given LAYERS.
SRS=namespace:identifier	Identifies the Spatial Reference System (SRS). For example: EPSG:4326 (Lat/Long).
BBOX=minx,miny,maxx,maxy	Specifies the bounding box corners (lower left, upper right) in Coordinate Reference System (CRS) units.
WIDTH=output_width	Width in pixels of the returning layer visualization.
HEIGHT=output_height	Height in pixels of the returning layer visualization.
FORMAT=output_format	Output format of map (GIF, JPEG, PNG, SVG, TIFF, WebCGM).

Table 4.1: Mandatory parameters of a WMS GetMap request.

custom style is created and stored in GeoServer, called user style. The user style is a clone of the default style at first and can be changed without affecting the layer visualization of other users. The default style acts as a template for the user style. Every time OpenLayers requests a layer, the WMS parameter STYLES is set to the user-style id. This way the user style is applied to the layer and a custom output is created and displayed. Referring to Figure 4.2, user style 1, user style 2 and user style 3 are independent styles which create user defined visualizations of the same map section. It is indispensable that the user style exists before OpenLayers invokes the WMS. Otherwise the layer can not be displayed.

So far the purpose of user styles is clear. In the following the approach how to adapt the user style to get the layer visualization for the user's needs is explained.

## Generating the User Style

In fact geospatial data has no visual component. To get a visual representation, the data must be styled by applying graphic variables to the data. GeoServer uses styles to accomplish this task. The syntax of a style is based on XML. XML tags allow to define styling options of single data properties. The style is capable describing the rendering of vector and raster data.

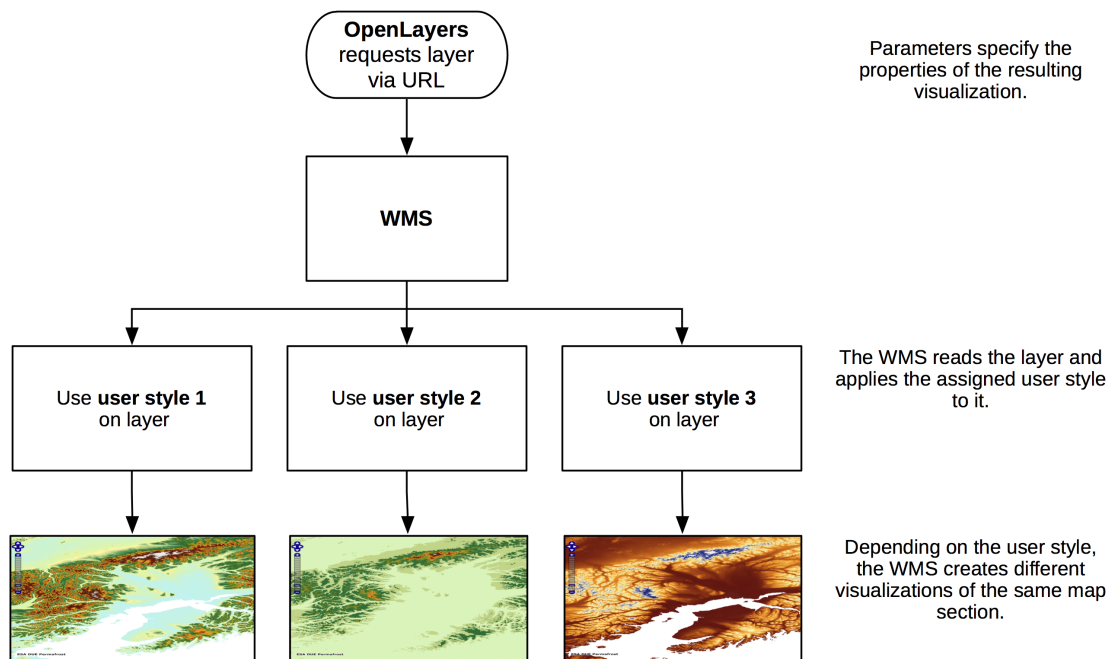


Figure 4.2: Different user styles create different visualizations without affecting each other.

The SLD specification contains many different styling options for graphic variables. The specification of the SLD uses symbolizer to define these styling options. All symbolizers, independent from the data format and their most interesting options, are listed below [9].

- The **PointSymbolizer** styles points, which are elements that contain only position information. It offers shape, opacity, size, rotation and external graphic as styling parameters.
- The **LineSymbolizer** styles lines that contain position and length. It provides parameters for stroke and fill. A pattern can also be applied.
- The **PolygonSymbolizer** styles two-dimensional geometry elements. The styling options are similar to the ones of the **LineSymbolizer**.
- The **TextSymbolizer** places text labels onto a certain area.
- The **RasterSymbolizer** applies color mapping to values. It does not support border or fill, since raster data does not provide position information.

In order to get a better understanding of the functionality of a style, a simple example is explained next. As the ESA DUE Permafrost project is providing raster data only, the visualization options are very limited. According to the permafrost data, the Listing 4.1 uses *RasterSymbolizer* to apply color mapping.

```

1  <FeatureTypeStyle>
2    <FeatureTypeName>Feature</FeatureTypeName>
3    <Rule>
4      <RasterSymbolizer>
5        <ColorMap type="intervals">
6          <ColorMapEntry color="#FF0000" opacity="1.0" quantity
7            ="1.0"/>
8          <ColorMapEntry color="#00FF00" opacity="1.0" quantity
9            ="2.0"/>
10         </ColorMap>
11       </RasterSymbolizer>
12     </Rule>
13   </FeatureTypeStyle>

```

Listing 4.1: Style example using *RasterSymbolizer*. The XLM tag *ColorMapEntry* defines the color mapping.

To gain control over the color assignment, the content of the tag *ColorMap* is important. Every *ColorMapEntry* tag assigns a *color* to a certain value or *quantity*. The attribute *type*=“*intervals*” of the *ColorMap* tag determines that values between two quantities, are assigned to the upper value and color. Without this attribute colors will be interpolated. According to this example, the attribute *type*=“*intervals*” means, that every data value with quantity less or equal 1.0 is colored in red (line 6) and every data value greater than 1.0 and less or equal 2.0 is colored in green (line 7). To make areas with a specific value transparent, the attribute *opacity* can be changed to 0. The value of the attribute *opacity* is set to 1.0 as default. A limitation of GeoServer is, that only 255 *ColorMapEntry* tags are supported. If more tags are defined, the layer can not be displayed.

Users can define their own styling options. In order to make the color mapping easier, the user can apply color schemas, also called colormaps. The user can choose between several colormaps. For fast color access, Color Look-up Tables (CLUT) are used. A CLUT is a table which stores information about colors and the corresponding values.

## Creating the Color Look-up Table from the Colormap

The concept of a Color Look-up Table (CLUT) is very simple but also efficient. The idea is, that all colormaps are created with the help of a graphical software and saved as images first. This way it is not necessary to set the color codes manually in the programming language and the colormap images can also be used for the user interface.

Every time GeoServer starts, all colormaps are loaded and the corresponding CLUTs are created. For each colormap a CLUT is created. Each color value of each pixel of a colormap is scanned in horizontal direction and saved in the CLUT. Figure 4.3 shows how the CLUT is created from a colormap.

Each colormap has a dimension of  $200\text{pixel} \times 20\text{pixel}$  irrespective of the screen resolution. 20 pixel in height are necessary for using the colormap in the user interface as an image. For creating the CLUT, one pixel in height would be enough. 200 pixel in width was chosen, because this allows to define distinguishable colors.

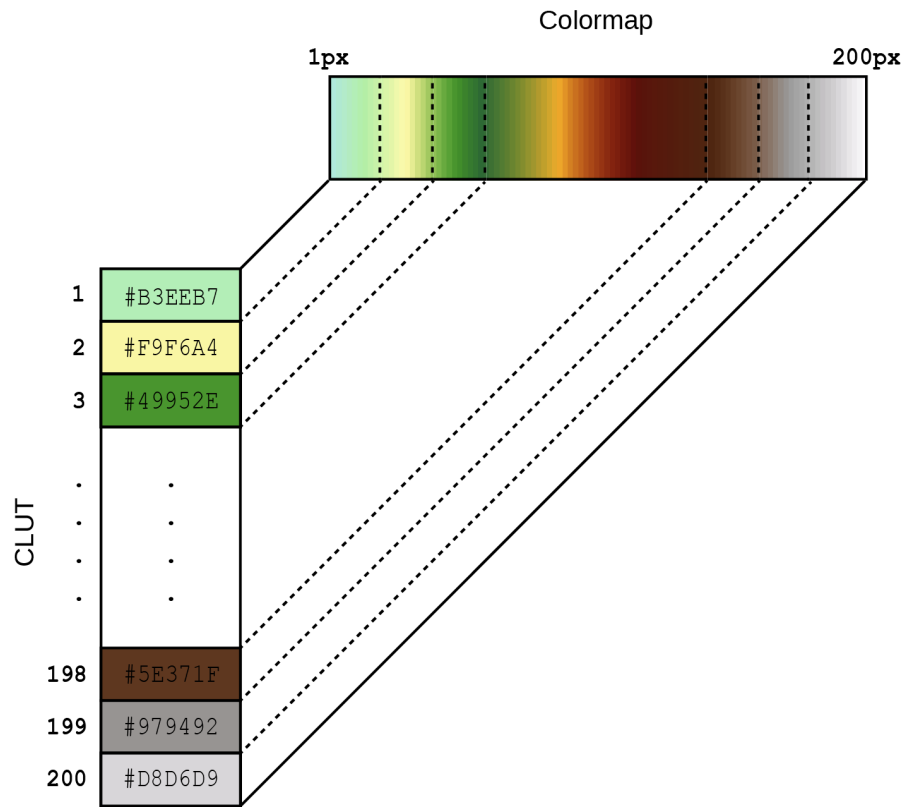


Figure 4.3: Each color value of each pixel in horizontal direction of the colormap is scanned and saved in the CLUT.

In every discipline colors do have a different meaning [2, 11]. According to the Remote-Sensing Products, Table 4.2 shows colormaps which are created to express the right meaning. Each layer in the Permafrost WebGIS expresses one permafrost soil relevant satellite-observable parameter and is type of one of the Remote-Sensing Products from Table 4.2. Each Remote-Sensing Product has a default colormap. The choices for the colormaps are done in cooperation with the project coordinator and are the common color schemas in geoinformatics for the listed Remote-Sensing Products.



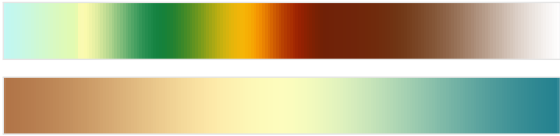



Remote-Sensing Product	Colormap
Digital Elevation Model	
Land Surface Temperature	
Surface Soil Moisture	
Day of Freeze/Thaw, Subsidence, Coherence	

Table 4.2: Remote-Sensing Products and their default colormaps.



## Implementation

The introduction of user styles has a major impact on the communication between the software components of the Permafrost WebGIS. According to Figure 5.1 the new communication is described as follows. Before a layer is added to the viewport, the Permafrost UI opens a dialog providing some information like layer name, server, opacity and also the id of the style which is applied. The Permafrost UI does not know the id of the user style yet, only the id of the layer's default style. A service is needed which provides the user-style id before the dialog is shown. This service is also part of this implementation and is named legend service.

The legend service returns the user-style id in a special format. Depending on two parameters, the default-style id and the user id, the service creates the user style derived from the default style and returns the id of the user style encoded as a JavaScript Object Notation (JSON) string. JSON is open standard and is designed for human-readable data interchange. Listing 5.1 shows an example of a JSON string using the user id 85 and default style `soil_moisture`.

```
1 {  
2   "result" : "success",  
3   "userStyle" : "_user_soil_moisture-85"  
4 }
```

Listing 5.1: Example for a JSON string after the user-style id was requested.

Every JSON string consists of a key-value pair which can also be nested. In this work it is kept simple.

- The value of the key **result** determines whether creating the user style was a success or not. The value can be *success* or *error*.
- **userStyle** determines the unique id of the user style. This id is used for the WMS request.

Before the legend service responses to the request, it checks if the according user style exists. If not, it is created. The Permafrost UI stores the returned user-style id in a database. The legend

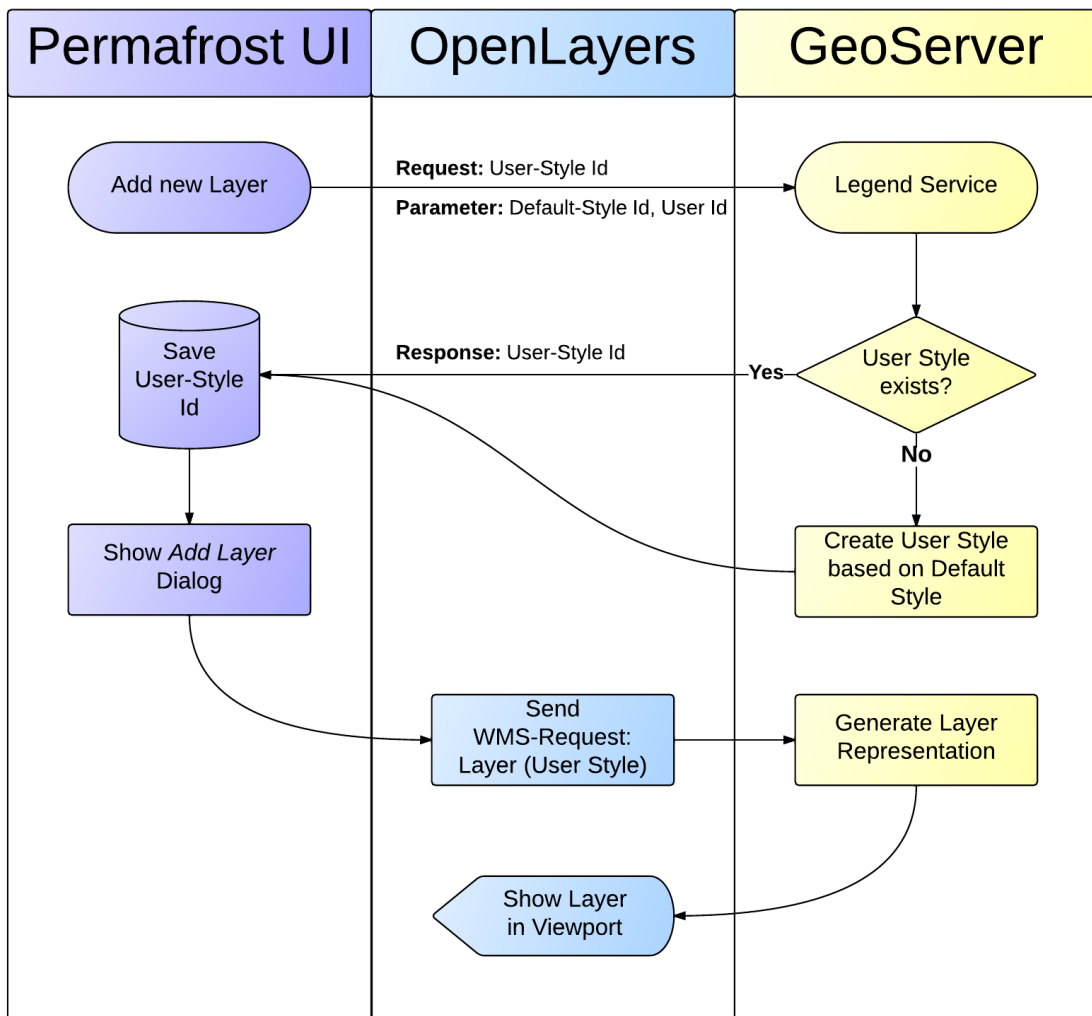


Figure 5.1: Schematic workflow of adding a layer to the viewport. The Permafrost UI requests the id of the user style. OpenLayers requests the layer image.

service guarantees that the layer can be displayed and the user-style id does not need to be created by the Permafrost UI.

From this point on, the Permafrost UI knows the user-style id of the current layer/user and can assign it to the WMS request. OpenLayers requests the layer and adds it to the viewport. It is indispensable that the user style exists before the WMS is invoked. Otherwise the layer can not be displayed. Now we can assume that the user style exists and the layer can be displayed. From now on, the Permafrost WebGIS has access to the user style.

## 5.1 The User Interface

The user interface (UI) unifies all identified requirements from Chapter 3. It plays a very important role in this thesis as it is the link between the user and the user style. The UI contains the control elements which affect the layer visualization. It is important to keep the user interface clean, simple and comprehensible. Since a lot of requirements are combined, the following components help to organize the UI.

- **Tabs** are used to group the settings of the visualization type *range* and *classification*.
- **Slider** is used for the color legend and to set the number of classes.
- **Colorpicker** lets the user choose a custom color.

The challenge is to bring all requirements together in one interface without overlapping functionalities. The requirements identified in Chapter 3 are converted into proper UI control elements and are arranged in the user interface. According to Figure 5.2, the user interface is

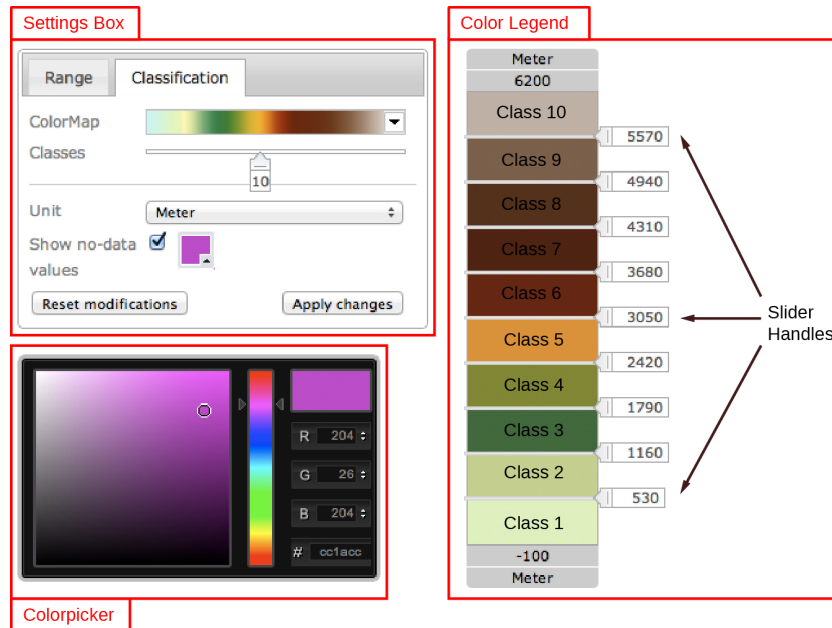


Figure 5.2: User interface of the reference implementation including the color legend.

divided into three blocks. The following list explains the elements of the UI.

- The upper left block (Settings Box) contains the main settings which affect the appearance of the color legend.
  - The user can easily toggle between the visualization types *range* and *classification* via tabs. Every time the visualization type changes, new settings are applied to the

color legend. Therefore the color legend is rebuilt by JavaScript. The previous color legend settings are preserved and restored if the visualization type changes again.

- The colormap determines the color schema used for the color legend. HTML does not support selection boxes with a graphical content. JavaScript is used to create a graphical selection box. If a different colormap is selected, it is applied to the color legend immediately.
- The number of classes in the color legend can be specified via a slider. If this value changes, the number of classes of the color legend changes too. New colors for each class are requested depending on the colormap.
- The unit can be changed for the Remote-Sensing Product Land Surface Temperature only. Other Remote-Sensing Products do not need to be converted, since no further measurement units are defined. For the Remote-Sensing Product Land Surface Temperature the following measurement units are available:

$$Kelvin \Leftrightarrow Fahrenheit \Leftrightarrow Celsius Degree$$

Changes are affecting the color legend. All values, including minimum and maximum of the color legend and handle values, are converted to the selected unit.

- Areas with no existing data can be highlighted in any color. This can be done with the colorpicker.
  - The checkbox for stretching the colormap is not included in the Figure, since this example shows elements of the visualization type *classification* only. If stretching the colormap is enabled, the selected colormap is stretched within the selected visible class. Otherwise the colormap fills the entire scale.
- The right block is the actual color legend. It is designed as a slider. It directly controls the mapping of the values to colors. It could be seen as a preview of the values of the Settings Box. The color legend should make interpreting the data and refining the visualization easier.
    - The current unit is located at the top and bottom of the scale.
    - The maximum and minimum of the entire scale are also located at the top and bottom of the scale.
    - All classes are arranged within the scale. Each class can be resized by moving the upper or lower handle. This allows to include more or less values to the class.
    - Each handle has a label box attached which shows the current value. If the handle is moving, the label changes too. This way the user can set precise limits for each class.
    - Clicking onto a certain class enables the colorpicker. This way values assigned to a class can be highlighted in the visualization.
  - Beneath the Settings Box, the colorpicker is located. The colorpicker is shown if the user wants to change the color of a certain class or of no-data values.

- By clicking the button *Apply changes*, the current settings are transformed into the user style. How this is done is explained in Section 5.4.
- Clicking the Button *Reset modifications* restores the default settings.

The UI can only be invoked for active layers. This means, the layer must be added to the viewport before the UI is invoked. Figure 5.3 shows how the UI is integrated into the layer-styling process. After a layer is added to the viewport, the UI can be invoked by the user. The default-style id and the user id are assigned as parameters. Values of the control elements are transformed into a valid user style after the settings of the UI are submitted. The Permafrost UI instructs OpenLayers to reload the layer visualization which applies the user style to the layer.

## 5.2 Style Files

GeoServer stores all styles as files in a common directory. Besides the style file another file is created called the legend meta. It is not a native file of GeoServer and contains the user settings of the UI. The legend meta is located in the same directory like the style file. It is supposed to save the user settings and permanent values of the user interface and does not affect the visualization of the layer. Every time the UI is invoked, the user settings are loaded and the UI state is recovered. If changes are submitted the legend meta will be updated.

The structure of the legend meta is kept very simple (see Listing 5.2). The tag *UserStyle* contains the following child tags: *Style* has only descriptive character and determines the id of the default style. *Scale* contains several tags that determine the permanent settings, like scale minimum and maximum of the color legend. *Settings* stores the user settings which are updated at runtime. First the tags of *Scale* are explained:

- **Min** is the minimum of the scale. This value is the label for the color legend's lower border.
- **Max** is the maximum of the scale. This value is the label for the color legend's upper border.
- **Classes** determines the maximum number of classes which the user can define for the visualization type *classification*.
- **Step** determines the step width of each handle, when it is moved by the user.
- **Units** defines all available units for conversion.

The tags of *Settings* are explained next:

- **Unit** specifies the current unit. This value must match one value of the available units for conversion.
- **Type** declares the visualization type (*range* or *classification*).
- **NoData** contains information about how areas without data are displayed.

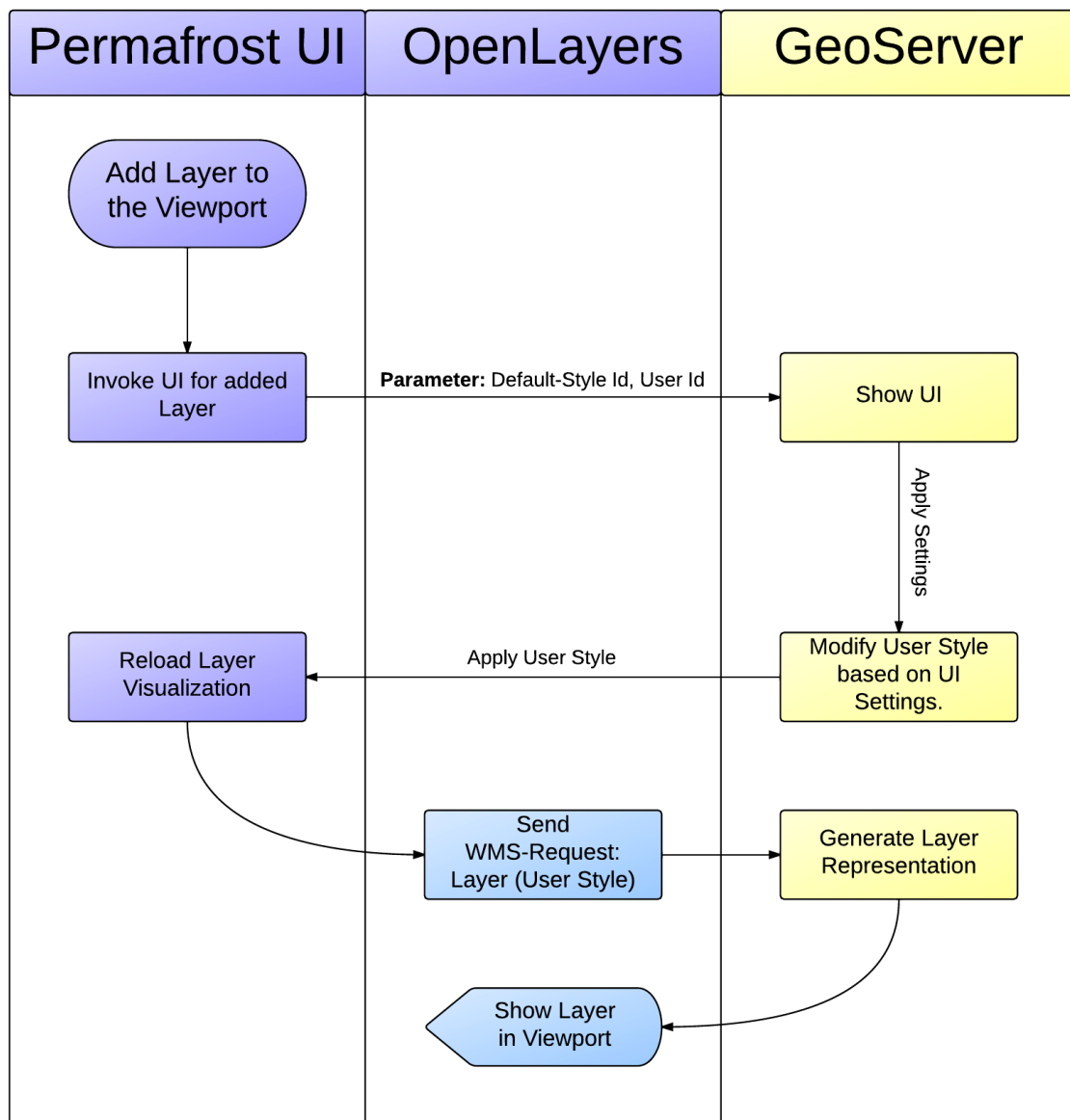


Figure 5.3: The UI is integrated into the layer-styling process. The user style, as a result of the UI, is applied to the layer and creates a custom visualization.

- **Values** determines the values of no data areas. These values can not be changed by the user.
- **Color** specifies the color used for no data areas.
- **Colormap** declares the current colormap.



- **Stretch** determines whether the colormap is stretched to the visible class or not.
- **Classes** specifies the values of the handles.
- **Colors** contains the colors of the classes. For visualization type *range*, the middle color tag is omitted.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <UserStyle>
3   <!-- Style Annotation -->
4   <Style>Topo</Style>
5   <!-- Fixed Scale Settings -->
6   <Scale>
7     <!-- Lower Scale End -->
8     <Min>-100</Min>
9     <!-- Upper Scale End -->
10    <Max>6200</Max>
11    <!-- Max Classes (Classification) -->
12    <Classes>20</Classes>
13    <!-- Handle Step Width -->
14    <Step>1.0</Step>
15    <!-- Defined Units for Conversion -->
16    <Units>
17      <Unit selected="True">Meter</Unit>
18    </Units>
19  </Scale>
20  <!-- User Settings -->
21  <Settings>
22    <!-- User-selected Scale Unit -->
23    <Unit>Meter</Unit>
24    <!-- Classified|Range -->
25    <Type>Range</Type>
26    <NoData>
27      <!-- Values used for no data -->
28      <Values>
29        <Value>-32768</Value>
30      </Values>
31      <!-- Color in HEX -->
32      <Color>#FF0066</Color>
33    </NoData>
34    <!-- Colormap Id -->
35    <Colormap>dem1</Colormap>
36    <Stretch>True</Stretch>
37    <!-- Values of Class Handles -->

```

```

38     <Classes>
39         <Class>1.0</Class>
40         <Class>3202.0</Class>
41     </Classes>
42     <!-- Colors of the Classes Between Handles -->
43     <Colors>
44         <Color>#ffffff</Color>
45         <Color/>
46         <Color>#eeeeee</Color>
47     </Colors>
48 </Settings>
49 </UserStyle>

```

Listing 5.2: Example of a legend meta. Default settings for the Remote-Sensing Product Digital Elevation Model.

Using the example on the style *topo*, which is the default style for the Remote-Sensing Product Digital Elevation Model, three files exists:

- **topo.xml** is the meta file and contains the GeoServer's internal style id and the corresponding filename. This file is managed by GeoServer and can be omitted for this implementation.
- **topo.sld** is the actual style und contains the visualization information. If the settings of the UI are changing, this file is updated.
- **topo\_meta.sld** contains the user settings of the user interface.

Each layer in GeoServer represents data of one specific Remote-Sensing Product. Each Remote-Sensing Product has a default style which is in further consequence the default style for all linked layers. All default styles are created once and will not be modified at runtime. A default style can be seen as a template for the user styles. The default-style id for each Remote-Sensing Product is listed in Table 5.1.

A user style is a clone of the default style first, but changes with the user settings of the UI. A user style is created for each user and default style. Changing the user style does not affect the visualization of other users but of the current user. The filenames of the user style and the legend meta follow certain criteria. The sign # marks a placeholder.

**User Style:**        \_user\_#DEFAULT-STYLE-ID#-#USER-ID#.sld

**Legend Meta:**     \_user\_#DEFAULT-STYLE-ID#-#USER-ID#\_meta.xml

Each filename consists of a prefix (\_user\_), the default-style id and the user id. This schema is also valid for the legend meta, although it has an additional suffix (\_meta). The prefix allows to identify all user files in the style directory. Table 5.2 shows an example how the user files are named.

Remote-Sensing Product	Default-Style Id
Digital Elevation Model	topo
Land Surface Temperature	lst_mean
Surface Soil Moisture	soil_moisture
Day of Freeze	freeze-rainbow
Day of Thaw	thaw-rainbow
Subsidence	subsidence-rainbow
Coherence	coherence-rainbow

Table 5.1: Remote-Sensing Products and their default-style ids.

	Style Name	Legend Meta
<b>Default</b>	topo.sld	topo_meta.sld
<b>User</b>	_user_topo-85.sld	_user_topo-85_meta.sld

Table 5.2: Examples for default and user files. For this example the default style is topo and user id is 85.

### 5.3 Color Legend

The most interesting UI control-element is the color legend as it controls the mapping from values to colors. This UI control-element replaces GeoServer's internal static color legend and can be adapted by the user interactively. Changing values of the Settings Box, causes the color legend to adapt to these changes. The color legend let the user define colors for certain values.

For the implementation of the color legend, jQuery is used. jQuery is a JavaScript library and provides a standardized API for manipulating HTML and event listening like mouse click on a specific HTML element. All common browsers are supported.

The jQuery slider plugin converts a selected HTML element into a slider. There are various options to set up the slider. In the following the main options are explained which are used in this work:

- **max** is the maximum value of the scale.
- **min** is the minimum value of the scale.
- **step** determines the step width of each handle.
- **values** determines the position of the handles within the scale.
- **orientation** specifies if the slider is vertical or horizontal.

These options are assigned every time the plugin is created and can also be changed at runtime. This work extends jQuery's slider plugin. In order to accomplish the previous mentioned requirements, the plugin needs to be extended. The color legend needs to access values from the Settings Box. Therefore the options of the slider plugin need to be extended:

- **type** defines the visualization type (*range* or *classification*).
- **unit** determines the unit which labels the scale. Also needed for unit conversion.
- **background** determines the relative path to the image which is used for the background of the scale. Only for visualization type *range*.
- **tooltips** can be *true* or *false*. If *true* every handle gets a label which determines the current value.
- **colors** allows to set custom colors for each class. The values must be in hexadecimal code.
- **stretch** can be *true* or *false*. If *true* the background image is stretched to the selected visible class. This option is applicable for visualization type *range* only.

Now these options can be accessed at runtime in the slider plugin. Every time values in the Settings Box are changing (e.g., colormap), these values are passed to the color legend immediately. Depending on the changes, the appearance and functionality of the color legend changes. The JavaScript Listing 5.3 shows how the slider is created and how the options are assigned.

```
1 // Create color legend slider
2 jQuery('#slider-legend').slider("destroy").slider({
3     type: set.type,
4     orientation: 'vertical',
5     min: set.scale.min,
6     max: set.scale.max,
7     unit: set.scale.unit,
8     values: set.classes,
9     background: set.colormapimage,
10    tooltips: true,
11    colors: set.colors,
12    stretch: set.stretch,
13    step: set.scale.step
14 });
```

Listing 5.3: This shows how the color legend is invoked. An empty HTML element is selected and the slider plugin is applied which adds custom HTML elements.

The jQuery slider plugin is attached to a certain element by a selector. A selector makes it possible to address any element in the HTML document. The element with the id *slider-legend* is addressed. The sign # marks the id of an element. After the element is selected, the slider plugin is invoked and the options are assigned.

Besides the adaption of the slider options, several adaptations in the code of the plugin are needed. Independent from the visualization type it is necessary to have multiple handles to define more than three value classes. The original plugin supports three classes only. The option *values* is supposed to handle two values which determine the position of the handles. Since the slider can theoretically handle any number of handles, the plugin is designed to support 20 handles. More handles would make the slider confusing. This means for every single value, assigned to the option *values*, a handle is created. A handle acts as limiter for two classes. Each handle can be moved between its neighbor handles or the border of the color legend. The space between two handles is called class and is implemented as an HTML element *div* which is supposed to be a container for other elements. This work does not explain all used HTML elements. The container fills the space between two handles or border. In this way each class can be colored by applying Cascading Style Sheet (CSS) rules. Every time a handle is moved, the adjacent container need to be recalculated in their height. Otherwise the container would slide over its borders (handles).

Listing 5.4 shows the code which recalculates the containers. Since the position of all containers is relative to its parent container, the positioning can be done in percent. This makes the rearrangement much easier. First the allocated height in percent of the adjacent container is calculated. Next the new height of the bottom container can be calculated since its height stops at the handle. After that the top container's height is calculated by subtracting the height of the bottom container of the allocated height. At the end it is necessary to reposition the top container.

```

1  // calculate allocated height or width in percent of adjacent
   container (range)
2  var allocatedPercent = parseFloat(this.ranges[index].css(
   widthOrHeight)) +
3    parseFloat(this.ranges[index+1].css(widthOrHeight));
4  var percentOffset = 0;
5
6  if (index > 0) {
7    percentOffset += (this.values(index-1) - this.options.min) /
8      (this.options.max - this.options.min) * 100;
9  }
10 var newHandlePercent = (this.values(index) - this.options.min)
   /
11   (this.options.max - this.options.min) * 100;
12
13 // calc width of container BEFORE handle
14 this.ranges[index].css(widthOrHeight, newHandlePercent -
15   percentOffset + "%");
16
17 // calc position and width of container AFTER handle
18 this.ranges[index+1].css(widthOrHeight, allocatedPercent -
19   parseFloat(this.ranges[index].css(widthOrHeight)) + "%");

```

```

20 this.ranges[index+1].css(leftOrBottom, newHandlePercent+"%");
21 \end{minted}
22 \label{}

```

Listing 5.4: Code fragment which recalculates the adjacent container of a handle.

Each handle has a label box attached. This box displays the current position of the corresponding label. Every time the handle changes in position, the label is updated.

Another important feature for highlighting areas is the colorpicker. This implementation also uses an existing plugin for jQuery [21], which is customized to fit the requirements. In order to attach the colorpicker to HTML elements, event listeners are used. First a CSS class (*colorpicker-activator*) is attached to certain div-containers. The jQuery slider plugin already uses a CSS class (*ui-slider-range*) for value classes which can be used for the event listener. Next the event listener *click* for these CSS classes is set up (see Listing 5.5). The event listener is a function which is invoked when the event fires. In this case the color of the calling element is converted into hexadecimal code. Afterwards the colorpicker is displayed and the color is passed. The colorpicker automatically updates the color of the element if another color was chosen. Clicking somewhere outside the colorpicker will let the colorpicker disappear. The colorpicker is also attached to the two outer classes of the visualization type *range*.

```

1  jQuery('.colorpicker-activator, .ui-slider-range').live('click'
    , function(event) {
2
3      event.stopPropagation();
4
5      var rgb = jQuery(this).css('background-color');
6      rgb = rgb.match(/^rgb\((\d+),\s*(\d+),\s*(\d+)\)$/);
7      function hex(x) {return ("0" + parseInt(x).toString(16)).
        slice(-2);}
8      hex = "#" + hex(rgb[1]) + hex(rgb[2]) + hex(rgb[3]);
9
10     var caller = jQuery(this);
11     jQuery('#colorpickerPlaceholder').ColorPicker({
12         singleInstance: true,
13         flat: true,
14         onChange: function (hsb, hex, rgb) {
15             caller.css('background-color', '#' + hex);
16
17             if (caller.hasClass('colorpicker-activator')) {
18                 jQuery('#styleMeta-user-noData-color').val('#' + hex)
19             }
20             if (caller.hasClass('ui-slider-range')) {
21                 preDefinedSettings[type].colors[caller.data('rangeId')]
22                     = '#' + hex;
23             }
24         }
25     });

```

```

23     }
24   });
25   jQuery('#colorpickerPlaceholder').ColorPickerSetColor(hex);
26   jQuery('#colorpickerPlaceholder').fadeIn(100);
27 });

```

Listing 5.5: The code fragment shows how the colorpicker is invoked and how the selected colors are applied to the element.

## 5.4 Transforming Settings to User Styles

This section is about transforming the user settings from the UI into a valid style definition. Since styles are XML documents with a known vocabulary (XSD) [14], all the control elements from the user interface can be converted into proper *ColorMapEntry* tags. The figures listed in this section, help to explain the relation between the color legend and the resulting style. It is important to know how the *ColorMapEntry* tags are sorted. The color legend has its lowest value at the bottom and its highest value at the top of the scale. All values are in descending order. This does not apply with the sorting of the *ColorMapEntry* tags. All *ColorMapEntry* tags are listed in ascending order in respect to the attribute *quantity*. If the *ColorMapEntry* tags are not sorted properly in the style file, GeoServer can not display the layer. For a easier understanding, the *ColorMapEntry* tags are sorted in descending order in the following figures.

After the UI settings are submitted, the color-mapping process is applied. This process differs depending on the user settings. The differences are explained in the following sections. The output of this process is a mapping list which contains a triple of *color*, *opacity* and *quantity*.

After all values of the scale are processed and inserted into the mapping list, all entries are sorted in ascending order according to *quantity*. Afterwards the *ColorMapEntry* tags are created and inserted into the user style inside the *ColorMap* tag. GeoServer does support 255 *ColorMapEntry* tags only. If more tags are listed, the visualization can not be displayed. But more tags are not required. Values without a corresponding *ColorMapEntry* tag are mapped to the next upper *ColorMapEntry* tag in respect to the *quantity*. This way it is necessary to define the upper border of a class only. Gaps can not occur.

### Visualization Type Range

The visualization type *range* is used to generate layer visualizations with color gradients. Color gradients make it hard to identify the corresponding value. According to Figure 5.4, the visualization type *range* consists of three classes: the visible class and two outer classes. The visible class defines the values which are visualized. All values between are mapped to a certain color depending on the colormap. The outer classes enclose the visible class and are used to highlight areas outside the visible class. Every time the height of the visible class changes by moving the lower or upper handle, the outer classes are changing too.

When the user submits the settings, the values of the visible class are determined and prepared for the style. For all values within the visible class, a color is requested from the CLUT.

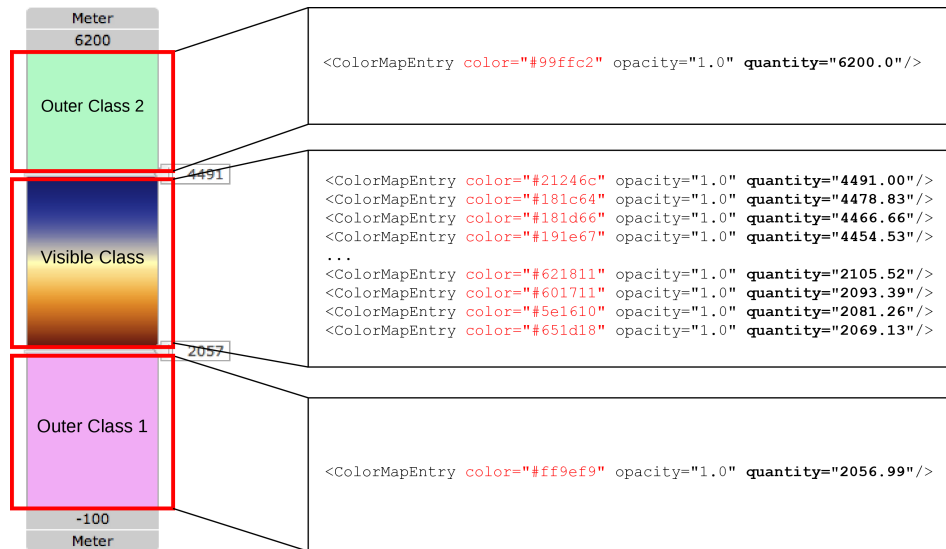


Figure 5.4: Two handles are dividing the scale into three classes. The outer classes can be colored to highlight or blank values outside the visible class.

Each color and corresponding value is saved in the mapping list. For the outer classes two class triples for lower and upper border are added to the mapping list. The colors for the outer classes are received from the color legend. Opacity has always the value 1.0 for values of the visible class and outer class. Afterward the mapping list is converted to the style.

Figure 5.5 shows the color legend and a part of the resulting style without outer classes. If one handle is at the scale's minimum/maximum, the corresponding outer class is not added since no outer class is needed.

Stretching the colormap is applicable to the visualization type *range* only since the selected colormap is applied to the visible class. Stretching is activated by default, which means that the colormap fits the visible class. If the value range of the visible class changes, the colormap adapts to the new size. The visible class shows always the full colormap. If stretching is deactivated, the colormap fills the full scale and the visible class shows only a part of the colormap. Only the visible part of the colormap is then extracted and saved in the mapping list. Figure 5.6 shows the color legend and the resulting style when stretching is deactivated. Compared to Figure 5.4, where stretching is activated, the resulting *ColorMapEntry* tags of the visible class differ in the value of the attribute color.

## Visualization Type Classification

While the algorithm for the visualization type *range* iterates over the scale and requests the colors from the CLUT, the visualization type *classification* receives the colors directly from the color legend. Since the colors of each class are already defined by the user, no mapping is needed. Each class is represented by one class triple for the upper border of the class. Figure 5.7 illustrates how classes and their colors are converted to *ColorMapEntry* tags. Lower and upper



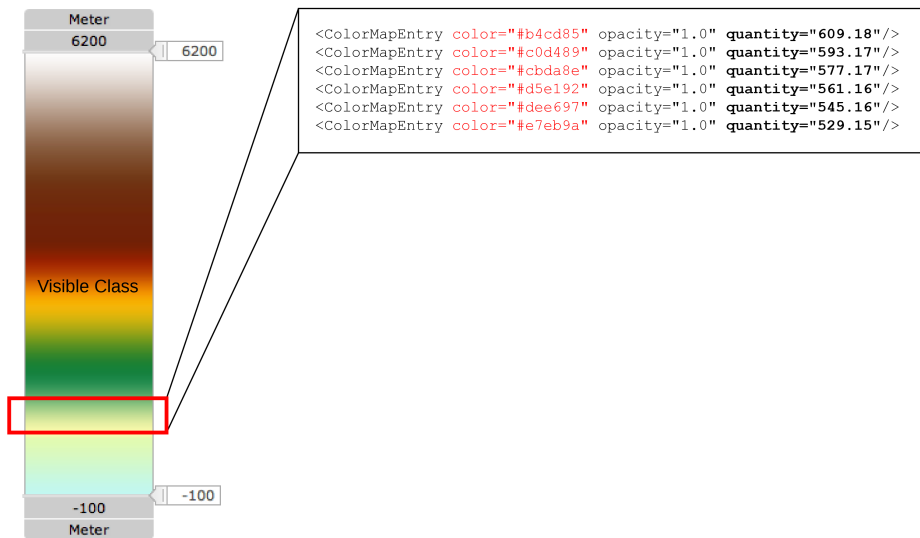


Figure 5.5: The visible class fills the whole scale. Outer classes are not visible since the handles are at the scale's minimum/maximum.

border as well as the color of each class are known. The entries can be created easily. Values between two class triples are automatically mapped to the upper border.

### Areas without Data

No-data areas are areas where the remote sensor has not scanned any data. Values of such areas are represented by a specific value, which varies for each Remote-Sensing Product since several independent partners are preprocessing the data. Depending on the Remote-Sensing Product, which the user style is based on, different no-data values are used. No-data areas exist for both visualization types. The user can set a custom color for no-data values with the use of the colorpicker. If the option *Show no-data values* in the UI is not selected, no-data values are set to transparent. Otherwise they are colored in the user set color. The transparency is specified with the XML attribute *opacity*. Figure 5.8 shows the corresponding XML code.

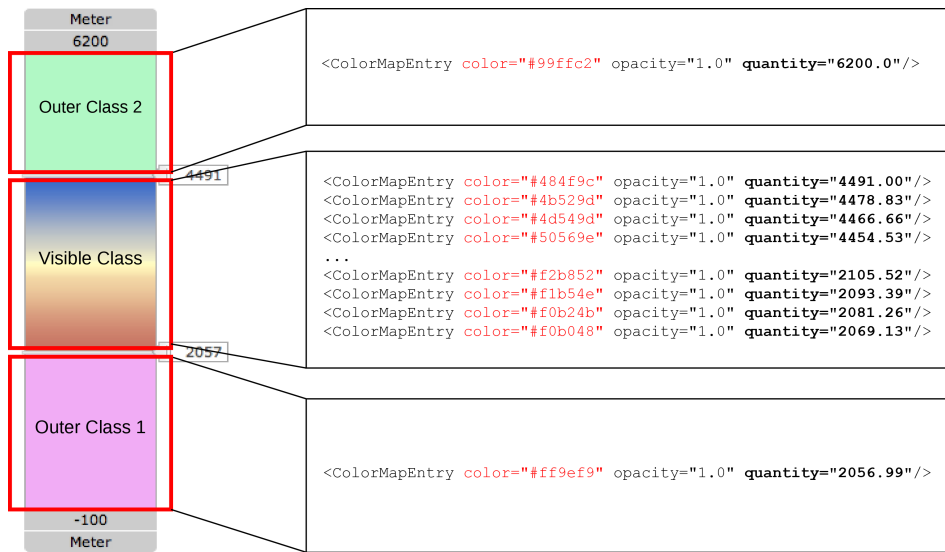


Figure 5.6: The colormap extends to the entire scale. The visible class shows only a part of the colormap.

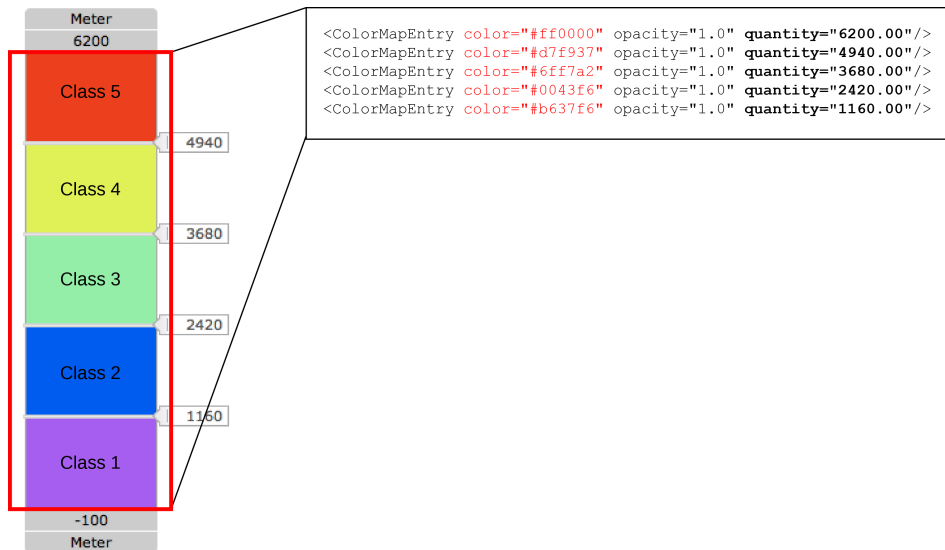


Figure 5.7: Five classes with different colors are distributed over the scale. Each class is defined as the upper border as well as a certain color.

```
1 <ColorMapEntry color="#0000FF" opacity="1.0" quantity="
  -32768.0"/>
```

(a) No-data values are colored in red.

```
1 <ColorMapEntry color="#0000FF" opacity="0.0" quantity="
  -32768.0"/>
```

(b) No-data values are transparent.

Figure 5.8: Listing (a) colors no-data values with the *quantity* -32768.0 in red (#0000FF). *Opacity* is set to 1.0. Listing (b) makes no-data areas transparent by setting the *opacity* to zero.



# CHAPTER 6

## Results

This Chapter shows the results of the implementation. Screenshots of the UI and the resulting layer visualization are compared. The results are compared to each other and the main differences are discussed. The color legend also acts as the description for the layer visualization. The following figures are created from the perspective of one user. Listing more examples for other users would make no sense, since user styles do not affect the visualization of other users. The current implementation was not tested by users of the ESA DUE Permafrost project since the Permafrost WebGIS was accessible for administrators and developers only at this time. As a proof of concept, the current implementation should be tested by the actual users of the ESA DUE Permafrost project.

All layer visualizations are created with the same map section (around Mount McKinley (Alaska)), map scale (1 : 904428) and coordinate system (EASE Grid North) and use data of the same Remote-Sensing Product (Surface Soil Moisture). Because the permafrost data is continuous, it is important to determine the date for the listed examples (2008-05-30). The low resolution ( $25\text{ km} \times 25\text{ km}$ ) of the Remote-Sensing Product Surface Soil Moisture causes pixelated images. A list of all parameters for all Remote-Sensing Products can be found in the Appendix A.

### 6.1 Visualization Type Range

*Range* is the simplest visualization type. It provides color gradients for values of the visible class. The output is always a colorful image. Areas of interest within a specific value range can hardly be identified due to smooth transitions. Data exploration is not served best. This visualization type is used for presentation purposes of the permafrost data only.

Figure 6.1 shows the result of the visualization type *range* including the entire scale. Since the unit of the product Surface Soil Moisture (SSM) is percent, 100% is the maximum. According to the corresponding colormap, areas with the highest moisture are displayed in dark blue (100%) while areas with low moisture are colored in dark brown (1%). This colormap

is the default colormap for SSM. Stretching the colormap is always enabled as default for the visualization type *range*. This does not play a role in this example since both handles are set to the upper and lower limit. No-data values are colored in white. In this example transparency is irrelevant since only one layer is displayed.

Figure 6.1b shows areas of brown and blue with smooth transitions. Areas of a different color can be determined easily. Areas with the same color but different brightness can hardly be specified. This can be improved by selecting a colormap with many distinguishable colors. Color gradients are not useful for visual data analysis.

## Outer Classes

Outer classes extend the functionality of the visualization type *range* with an important aspect of the visualization type *classification*. Outer classes highlight values outside the selected visible class. They are located at the lower and upper end of the color legend. Every time the height of the visible class changes, the adjacent outer class of the moved handle is adjusted.

Figure 6.2 illustrates the use of outer classes. The two outer classes are limited by the handles. At the following examples the variable  $x$  represents values of the permafrost data. The lower outer class includes all values with  $1 \leq x < 35$  while the upper outer class includes values with  $75 < x \leq 100$ . All values that fulfill  $35 \leq x \leq 75$  are lying inside the visible class. Because stretching is active, the colormap is applied on the visible class only. In Figure 6.2b the outer classes are clearly visible, colored in red and green. Comparing the Figure 6.2a to Figure 6.1a, the colormap of the visible class is more compressed in Figure 6.2a because of the smaller visible class. While intense colors are replaced by the outer classes, light colors turned more intense. This is because the positions of the handles and size of the visible class have changed. Outer classes allow to hide or highlight areas outside the visible class.

## Stretching the Colormap

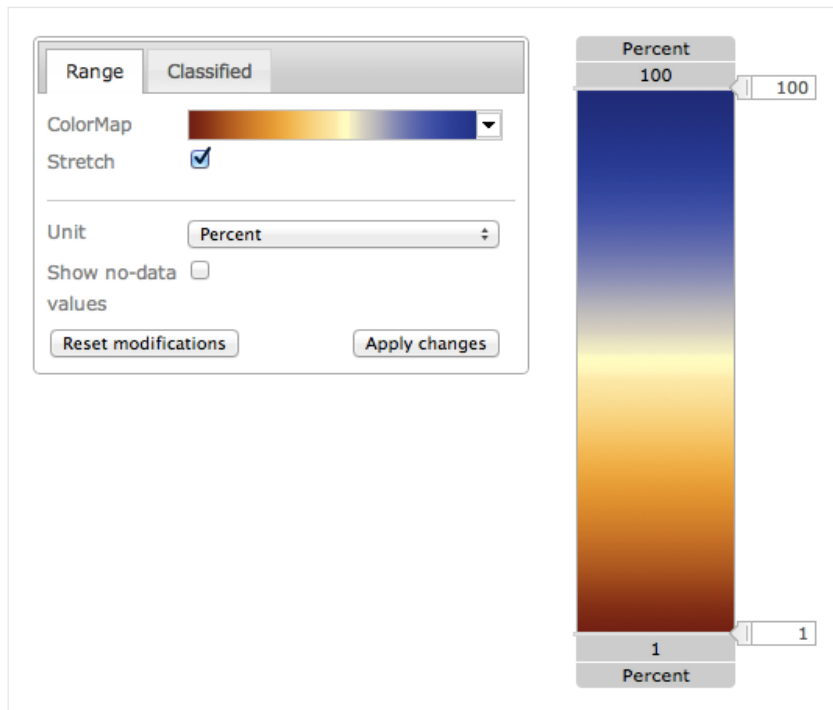
Since stretching is active as default, this section explains the result when it is deactivated. If it is active, the colormap is applied to the visible class only (see Figure 6.1a). If it is deactivated the colormap fills the entire scale and the colors outside the visible class are clipped. Compared to Figure 6.2b, colors are brighter in Figure 6.3b.

## 6.2 Visualization Type Classification

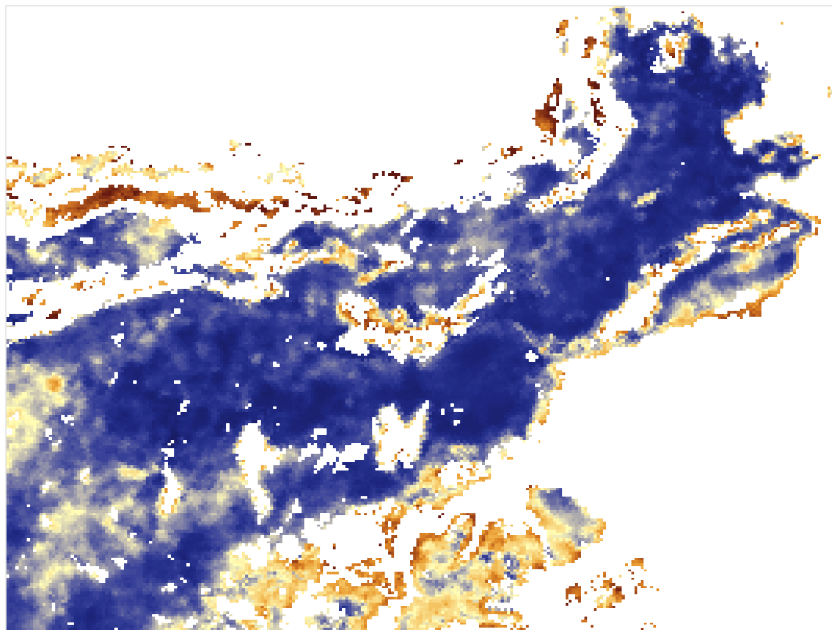
*Classification* is the second visualization type and designed for explorative research. The entire scale is divided into classes (at least two, at most twenty). The amount of classes is limited because of the user interface. More classes also means more handles on the color legend. This would make the color legend confusing.

*Classification* always visualizes all values of the scale. The scale is divided into classes which can be adjusted by the user. This allows to highlight areas of interest. Each class has a color which can be changed by the user.

Figure 6.4 allows a good comparison to Figure 6.1 because of the same selected colormap in the Settings Box. In this example *classification* provides two classes of blue tones, one gray

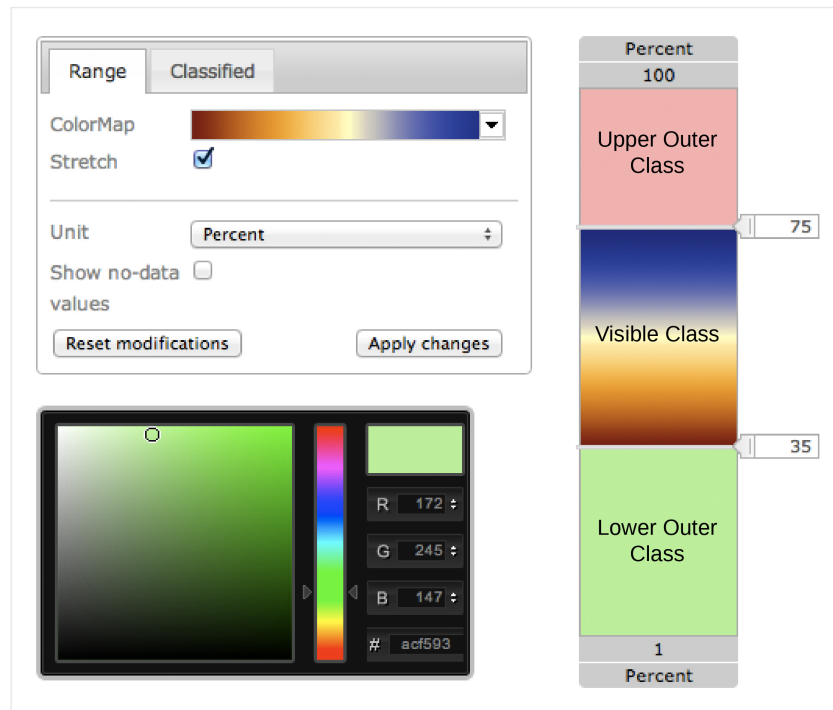


(a) Screenshot of the user interface.

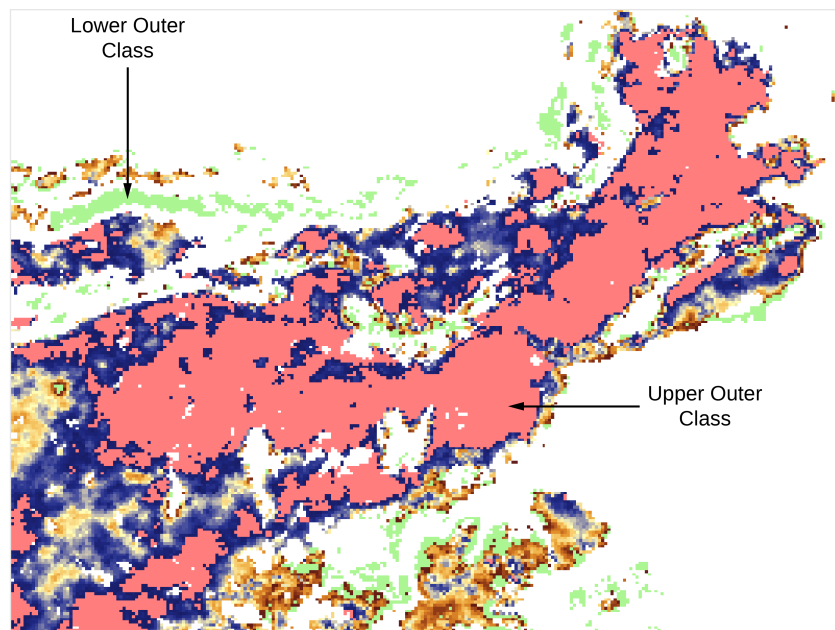


(b) Resulting layer visualization.

Figure 6.1: Areas of low moisture (brown) and high moisture (blue) can be identified easily in Figure 6.1b



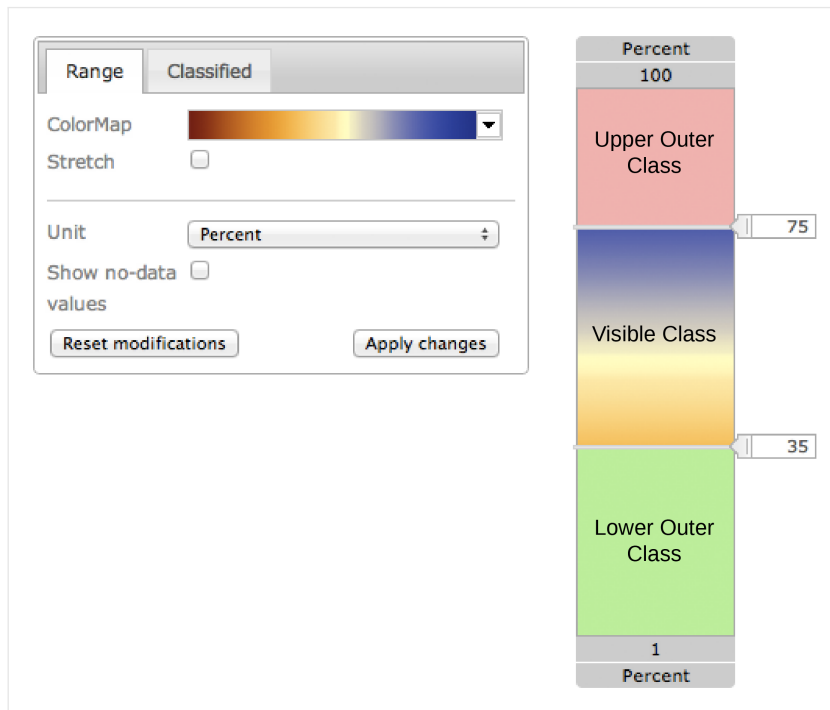
(a) Screenshot of the user interface.



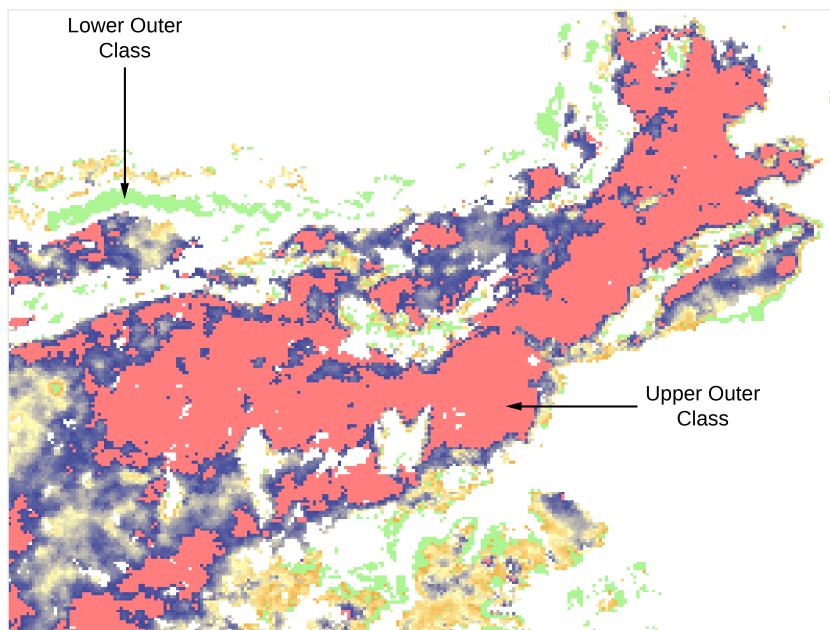
(b) Resulting layer visualization.

Figure 6.2: The outer classes, set to green and red, can be identified easily in the resulting layer visualization.





(a) Screenshot of the user interface.



(b) Resulting layer visualization.

Figure 6.3: (b) shows the output when colormap stretching is disabled. Compared to Figure 6.2b colors are brighter.

class and three more colors. Classes are equally distributed. All class colors are discretized from the selected colormap. Because each class has a unique color, values of these classes can be identified in the visualization. Because the visualization type *range* provides color gradients, the determination of monochrome areas is difficult. This reflects that the visualization type *classification* is better suited to accomplish explorative research than the visualization type *range*.

## User-defined Classes

*Classification* allows to highlight classes of specific value ranges. Every class and values within the class can be colored by the user. Figure 6.5 shows how this is done. The lowest and highest classes are colored in green and red. By clicking on the class, the colorpicker appears and the current color is preset. Figure 6.5b clearly highlights values within the selected classes.

Hiding areas is accomplished by coloring classes in white. But this does not make these areas transparent to other layers. Transparency is not implemented in this work, because the Permafrost UI already supports transparency of single layers.

The number of classes of the visualization type *classification* are limited to twenty and can be decreased to two. Figure 6.6 shows *classification* by means of fourteen classes. A different colormap than in the other figures is used to see the differences in the output. The settings result in an image with fourteen discrete colors. Areas can be distinguished despite the many colors.

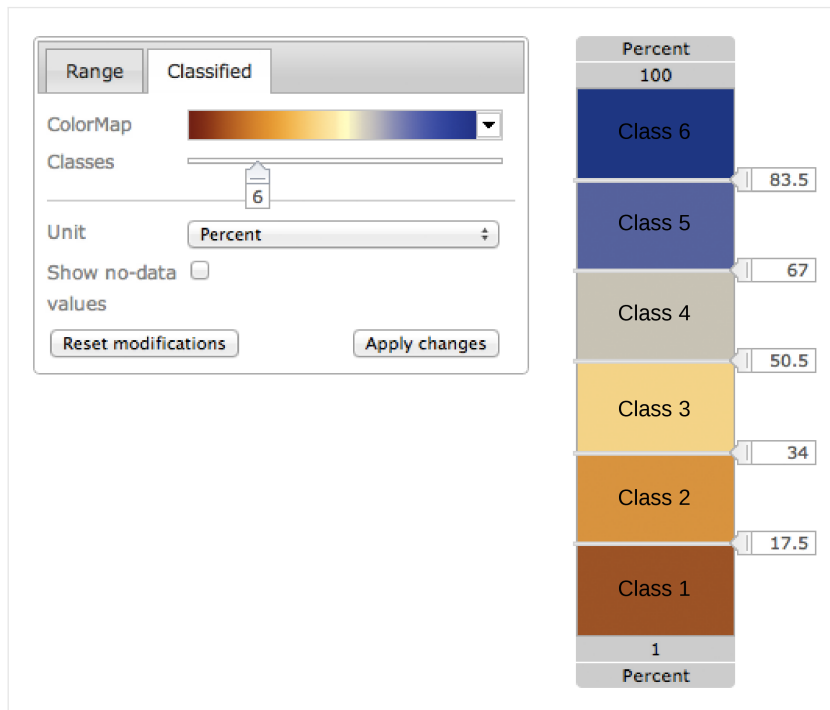
## 6.3 Using different Colormaps

Figure 6.7 illustrates the use of another colormap on the visualization type *range*. Although the rainbow colormap does not provide meaningful colors for SSM, it contains a lot more colors than the other colormaps. That is why it is used for this example. This allows to better distinguish areas. In Figure 6.6 the same colormap is used but for the visualization type *classification*.

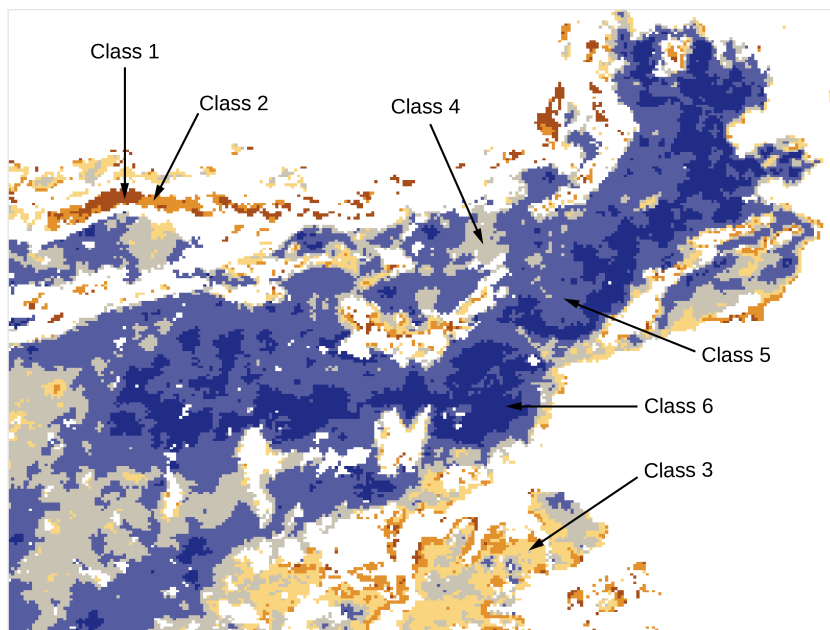
## 6.4 Highlighting Areas without Data

Highlighting areas without data is very important to distinguish them from actual data. Highlighting is applicable to both visualization types. Since no-data areas are transparent by default, they can not be distinguished from the white background. The color white can also be applied to values, which makes it impossible to determine whether the areas are without data or colored.

Depending on the Remote-Sensing Product, the values for areas without data are varying. On default these values are transparent. Only if a color is selected, the opacity is set to the maximum value 1.0. Figure 6.8 shows an example. For SSM no-data values are defined as  $x = 0, x > 100$ . The variable  $x$  represents values of the permafrost data. All values for which these conditions apply, are colored in green. The result can be seen in Figure 6.8b.

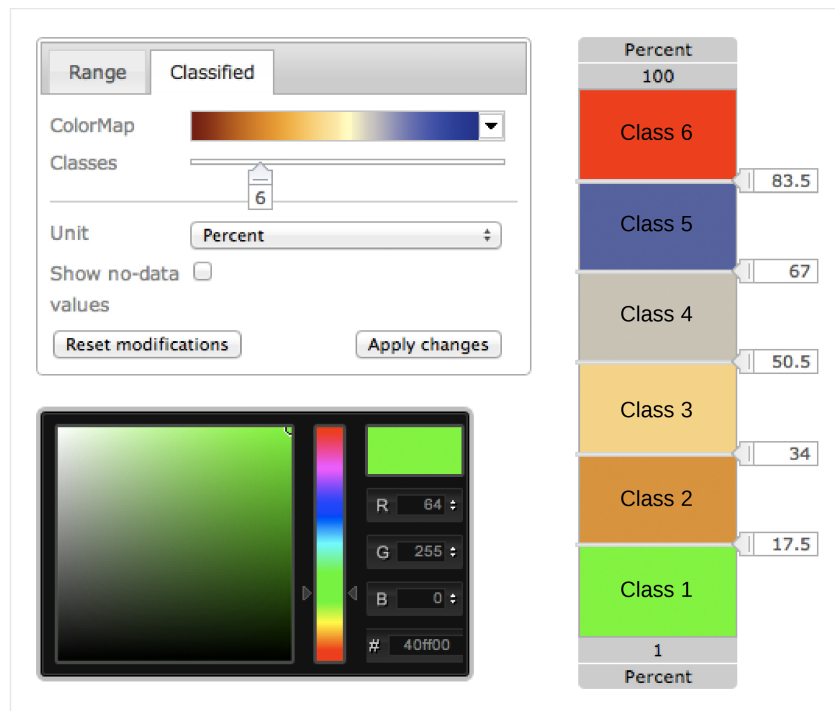


(a) Screenshot of the user interface.

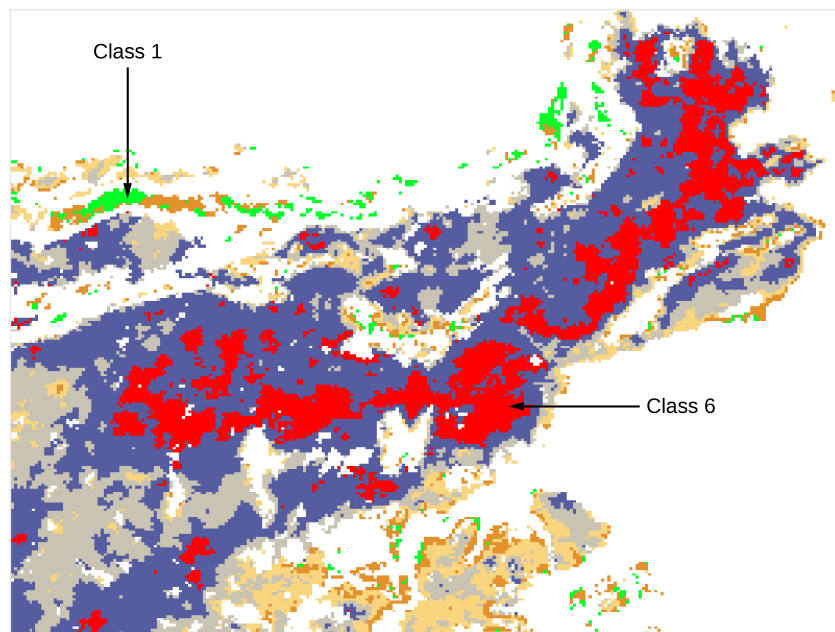


(b) Resulting layer visualization.

Figure 6.4: (a) The entire scale is divided into six classes of the same height. (b) The output clearly visualizes these classes.

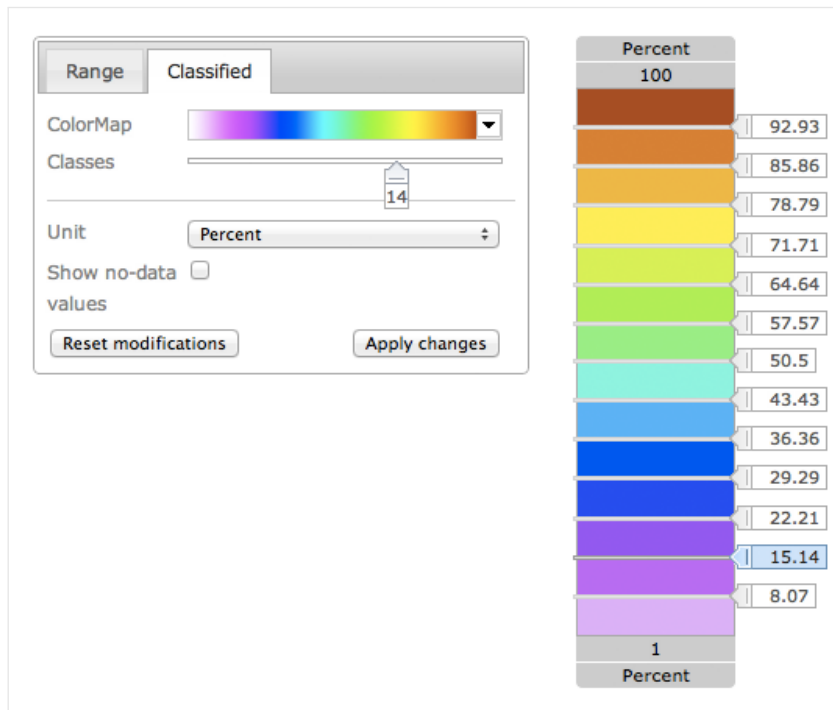


(a) Screenshot of the user interface.

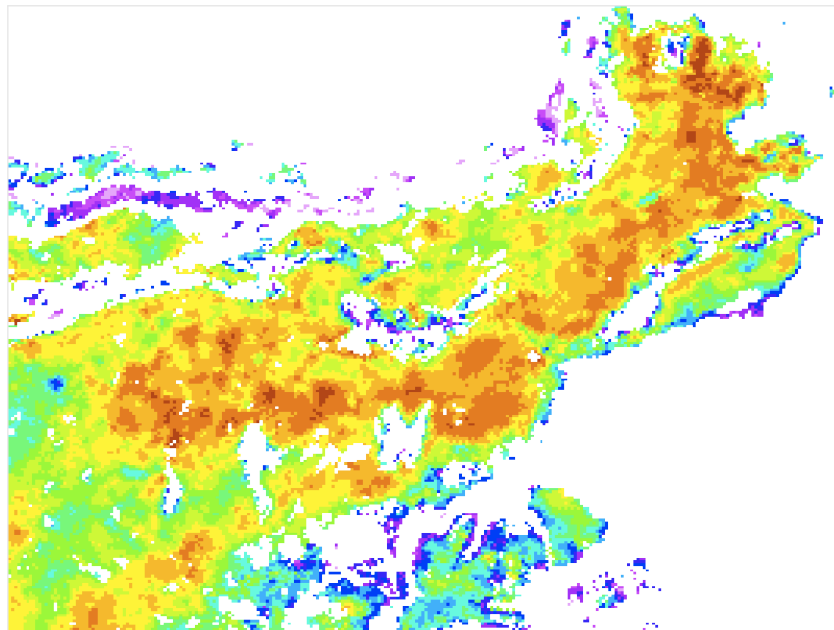


(b) Resulting layer visualization.

Figure 6.5: The lowest and highest class are colored in green and red.

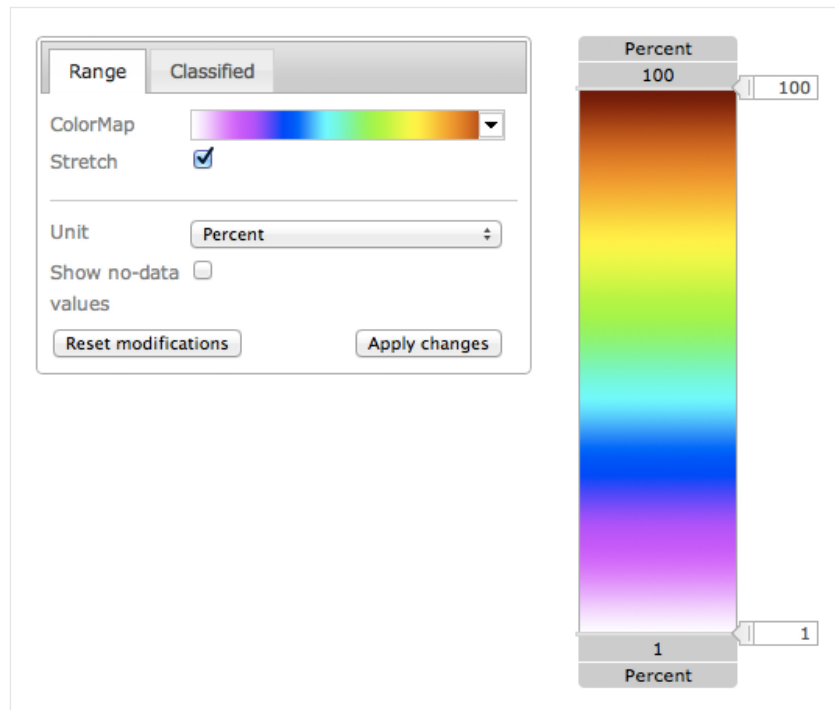


(a) Screenshot of the user interface.

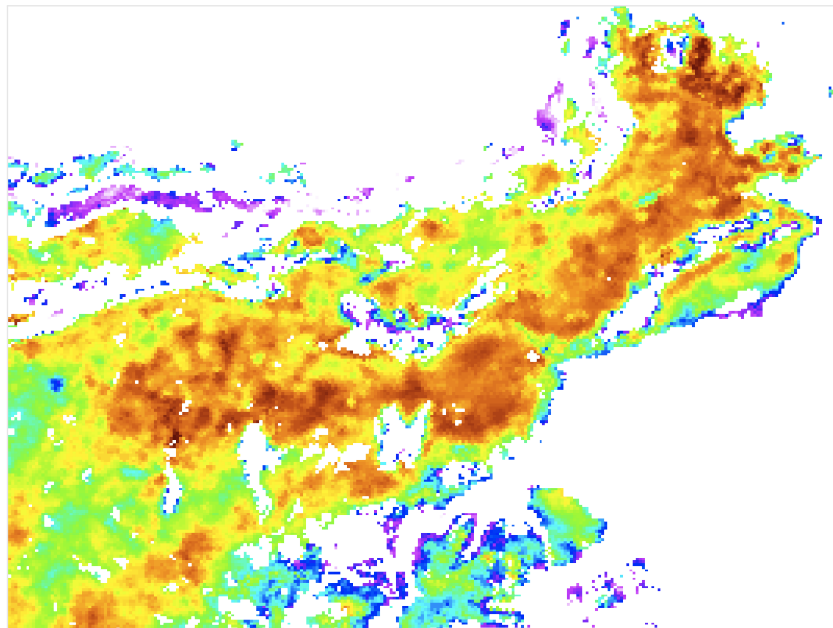


(b) Resulting layer visualization.

Figure 6.6: Fourteen classes are applied on a rainbow colormap which can be distinguished despite the many colors.

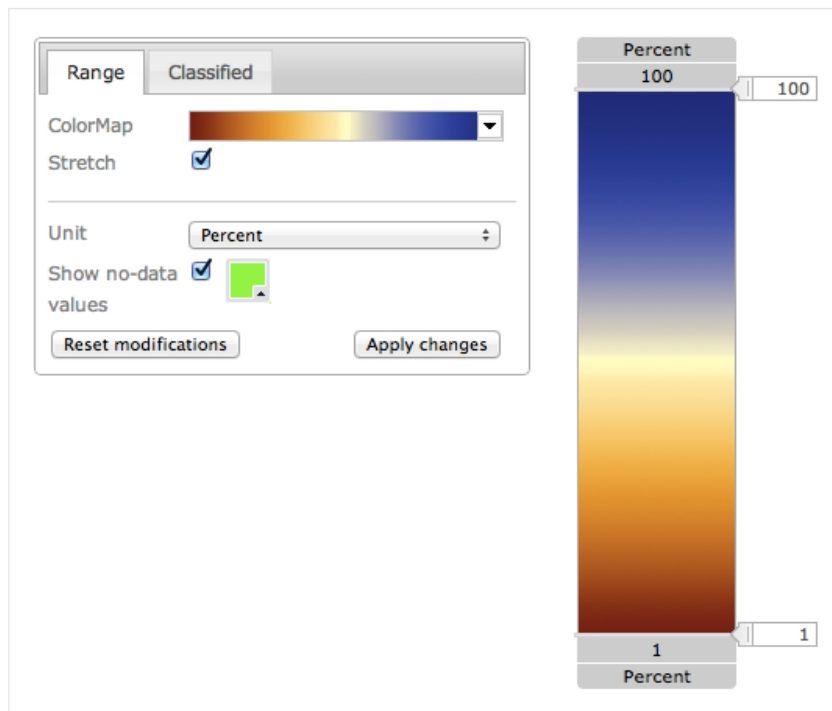


(a) Screenshot of the user interface.

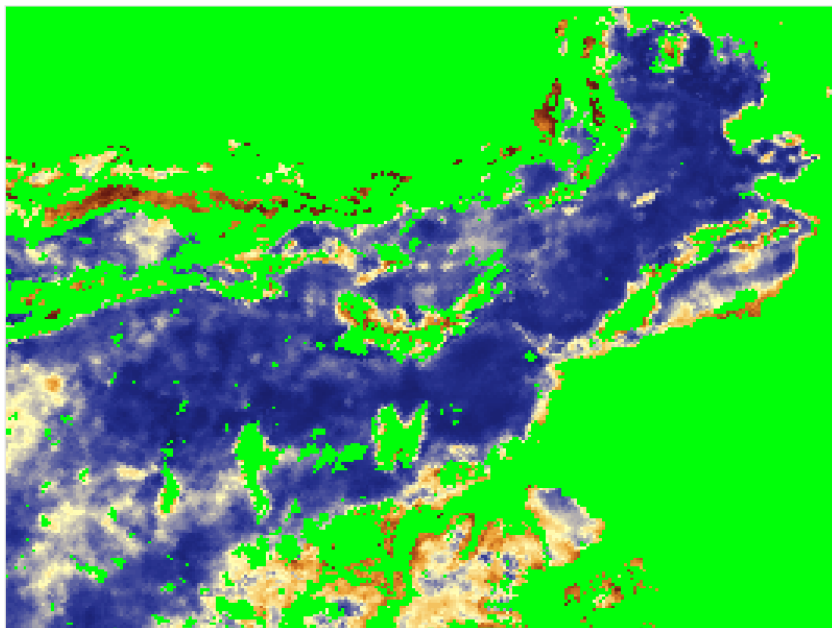


(b) Resulting layer visualization.

Figure 6.7: The selected colormap in Figure (b) contains a lot of different colors compared to the other provided colormaps.



(a) Screenshot of the user interface.



(b) Resulting layer visualization.

Figure 6.8: Areas where no data exists are highlighted in green.

## 6.5 Overlapping Layers

The Permafrost WebGIS is capable to visualize overlapping layers. Overlapping layers can help to orientate the permafrost data on reference points like the geographic borders or water bodies. The following Remote-Sensing Products and additional geographic information are used to demonstrate overlapping layers:

- Digital Elevation Model (DEM)
- Surface Soil Moisture (SSM)
- Political border (additional geographic information)

Figure 6.9 provides the color legend for DEM and SSM. Figure 6.10 shows the result of the visualization. The data of the used Remote-Sensing Products are of different resolution which can be seen in the result.

The layer of the SSM overlaps the layer of the DEM. This work supports transparency for no-data areas only. Areas without data of the DEM, are shining through the layer of the SSM. Transparency for layers is supported by the Permafrost WebGIS. This means, the opacity can be adjusted for the entire layer. Figure 6.11 shows the result of the transparent layer SSM. In this case interpreting the data is more difficult, because the colors of the layer SSM are brighter and do not match with the color legend. Also the colors of the DEM covered by the SSM can not be interpreted.

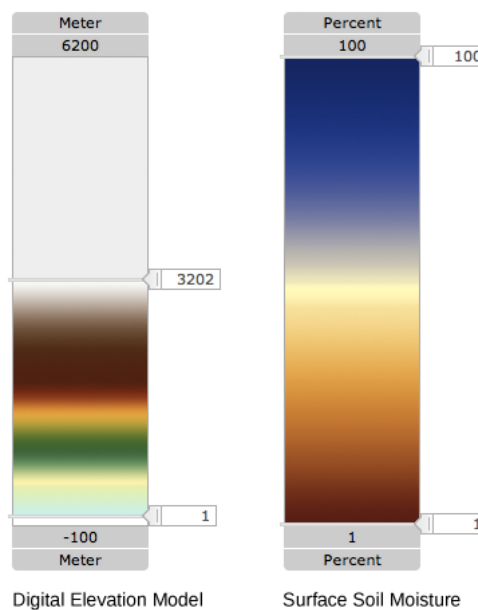


Figure 6.9: Color legends of the Digital Elevation Model (left) and the Surface Soil Moisture (right).



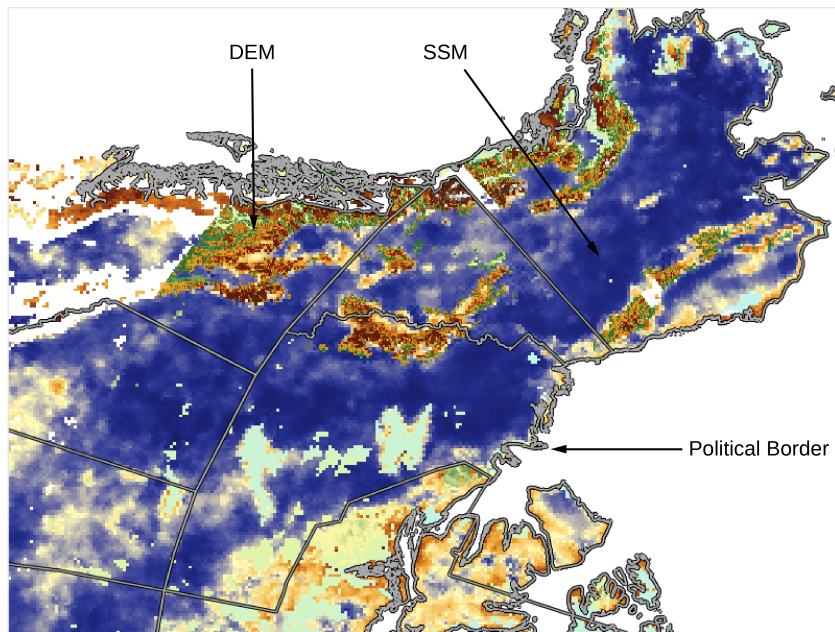


Figure 6.10: The layer of the SSM overlaps the layer of the DEM. The political border is on top of both layers.

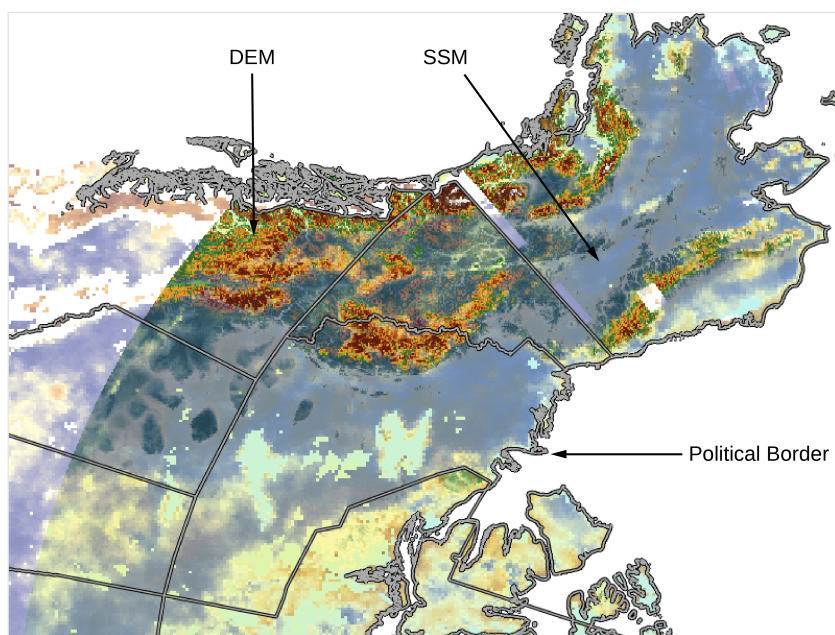


Figure 6.11: The opacity of the layer SSM is set to the value 0.5. The layer of the DEM is shining through.



## Conclusion

In OGC compliant applications, the style is essential for adapting the layer visualization. The style defines the styling information and declares how the mapping of values to graphical variables is done. GeoServer also implements this method to render a layer visualization. This means that the rendering process is limited to the styling options provided by the SLD specification.

Despite this limitation, the SLD specification provides a lot of styling options. Depending on the data format, graphical variables can be applied. Since the permafrost data is of type raster data, the data lacks of information about geometric objects. This fact has limited the styling possibilities of this work. The specification of the SLD does provide color and brightness as graphical variables for raster data only. So this work has concentrated on tools to enable data exploration by applying color mapping.

Although color mapping is a simple method, all requirements on this thesis could be implemented. It was an aim of this work to define a set of UI control-elements which allow exact examination of features of the raster data. This task could be accomplished, as the results in Chapter 6 show. Value ranges of interest can be highlighted in the layer visualization. Transparency of class colors was not implemented, assuming that transparent areas in combination with overlapping layers would make the visualization confusing for the user. With decreasing opacity, interpreting a color becomes more difficult.

To enable user-driven manipulation of the layer visualization without affecting the visualization of other users, user styles were introduced. As a result of the introduction of the user styles, the communication within the Permafrost WebGIS becomes more complex. It is necessary to provide the user with a style and its id before the user views a layer. A service was implemented which assures that the user style exists. This service creates three user-style files for each user and a default style. This fact accrues a lot of data garbage over the time. The ESA DUE Permafrost project includes six Remote-Sensing Products. For each Remote-Sensing Product and user, a user style is created. Assuming that the Permafrost WebGIS is expected to manage about 100 users, 1800 user-style files are created. If the permafrost data is updated and makes it necessary to update the layer visualization, all user styles need to be deleted or updated

based on the new default style. Styles can only be administrated by using GeoServer which is time-consuming. GeoServer manages a virtual copy of all styles. If the user-style file is deleted and the virtual copy still exists, GeoServer needs to be restarted. In the testing, all user-style files were deleted manually several times. Afterwards GeoServer was restarted. All styles were updated and the virtual copy of the deleted files were removed. Restarting GeoServer is not an option for the real operation.

In the testing, GeoServer had sometimes a problem with the rendering of specific layers. After the user style was updated, the layer could not be displayed immediately. Especially layers based on the Remote-Sensing Product Land Surface Temperature had some latency before they were correctly displayed. This problem could not always be reproduced and occurred randomly. Analyzing the HTTP response of the WMS showed that GeoServer produced the latency. In this case, the WMS returned a blank image. The exact source of the problem could not be identified in this work but it could be related to GeoServer's caching methods.

Although the data can be examined with the implementation of this work, styling options for vector data would serve the explorative research better. For example, if the user could apply different patterns to value ranges, areas of interest would be more distinguishable from other areas. Therefore the permafrost data is needed to be converted into vector data. Since this process is complex, this would need further research.

## Remote-Sensing Products

Remote-Sensing Products in context of the ESA DUE Permafrost project are relevant satellite-observable parameters. Each product has different parameters. The following Tables list all products of the ESA DUE Permafrost project and their relevant parameters. These parameters are very important for this work since they determine basic information for the legend.

A Remote-Sensing Product represents several permafrost soil characteristics. Based on these characteristics any number of data in form of layers can exist. This means various layers of the same Remote-Sensing Product but different settings, can exist side by side. The main parameters this work is based on, can be found in the following Tables. It is necessary to explain the meaning of the parameters before listing the characteristics of each Remote-Sensing Product.

Several Remote-Sensing Products are created by more than one remote sensor. Parameters of the same Remote-Sensing Product but different sensors are listed in the same table. The remote sensor is not necessary for this work and is therefore omitted.

- **Remote-Sensing Product** determines the name of the product.
- **Unit** determines the unit of the data values.
- **Maximum value** specifies the maximum value of the product dataset.
- **Minimum value** specifies the minimum value of the product dataset.
- **No-data values** determine values which represent cells without information.
- **Spatial resolution** determines the resolution of the remotely sensed data.
- **Temporal frequency of input data** determines which interval the data has been aggregated. This allows to visualize different dates of the product.
- **Maximum time period** specifies the period in which the data has been acquired.

<b>Parameter</b>	<b>Specification</b>
Remote-Sensing Product	Land Surface Temperature
Unit	Kelvin
Maximum value	400
Minimum value	1
No-data values	0
Spatial resolution	$1\text{ km} \times 1\text{ km}$ $25\text{ km} \times 25\text{ km}$
Temporal frequency of input data	Weekly, Monthly
Maximum time period	2000 – 2010

Table A.1: Characteristics of the product **Land Surface Temperature** [7].

<b>Parameter</b>	<b>Specification</b>
Remote-Sensing Product	Digital Elevation Model
Unit	Meter
Maximum value	6200
Minimum value	-100
No-data values	-32768
Spatial resolution	90 m
Temporal frequency of input data	Once
Maximum time period	–

Table A.2: Characteristics of the product **Digital Elevation Model** [18, 19].

<b>Parameter</b>	<b>Specification</b>
Remote-Sensing Product	Day of Freeze/Thaw
Unit	Day of year (DOY)
Maximum value	366
Minimum value	1
No-data values	-1
Spatial resolution	$1\text{ km} \times 1\text{ km}$
Temporal frequency of input data	Yearly
Maximum time period	2005 – 2010

Table A.3: Characteristics of the products **Day of Freeze/Thaw** [16, 17].

<b>Parameter</b>	<b>Specification</b>
Remote-Sensing Product	Surface Soil Moisture
Unit	Percent
Maximum value	100
Minimum value	1
No-data values	0, $x > 100$
Spatial resolution	$1\text{ km} \times 1\text{ km}$ $25\text{ km} \times 25\text{ km}$
Temporal frequency of input data	Weekly
Maximum time period	2007 – September 2010

Table A.4: Characteristics of the product **Surface Soil Moisture** [16, 17].

<b>Parameter</b>	<b>Specification</b>
Remote-Sensing Product	Water Bodies
Unit	Value
Maximum value	1 (water body exists)
Minimum value	0 (no water body exists)
No-data values	-32768
Spatial resolution	$150\text{ m} \times 150\text{ m}$
Temporal frequency of input data	Yearly
Maximum time period	2007 – 2011

Table A.5: Characteristics of the product **Water Bodies** [20].

<b>Parameter</b>	<b>Specification</b>
Remote-Sensing Product	Subsidence
Unit	Meter
Maximum value	0.25
Minimum value	-0.25
No-data values	0
Spatial resolution	$\sim 20\text{ m}$
Temporal frequency of input data	Varying, as available
Maximum time period	2007 – 2010

Table A.6: Characteristics of the product **Subsidence** [22].

<b>Parameter</b>	<b>Specification</b>
Remote-Sensing Product	Coherence
Unit	Value
Maximum value	1
Minimum value	0.0001
No-data values	0
Spatial resolution	$\sim 20\text{ m}$
Temporal frequency of input data	Varying, as available
Maximum time period	2007 – 2010

Table A.7: Characteristics of the product **Coherence** [22].



# Bibliography

- [1] The Arctic, 2009. URL <http://de.wikipedia.org/w/index.php?title=Datei:Arctic.svg&filetimestamp=20090809182959>. Accessed: 2012-08-29.
- [2] S. Bartel. *Farben Im Webdesign: Symbolik, Farbpsychologie, Gestaltung*. X.media.press / publishing. Springer, 2003. ISBN 9783540439240.
- [3] A. Bartsch. ESA DUE Permafrost Project, 2012. URL <http://www.ipf.tuwien.ac.at/permafrost/index.php/project-description>. Accessed: 2012-04-18.
- [4] A. Bartsch, A. Wiesmann, T. Strozzi, C. Schmullius, S. Hese, C. Duguay, B. Heim, J. Boike, and M. Herold. Implementation of a satellite data based permafrost information system - the DUE Permafrost Project. In *Proceedings ESA Living Planet Symposium, ESA Special Publication SP-686, Bergen, Norway, 26 June- 02 July 2010*, GRS, Laboratorium voor Geo-informatiekunde en remote sensing, Laboratory of Geo-information Science and Remote Sensing, 2010. ESA.
- [5] J. de la Beaujardiere. OpenGIS Web Feature Service 2.0 Interface Standard Web Map Server Implementation Specification, 03 2006. URL <http://www.opengeospatial.org/standards/wms>. Accessed: 2012-04-05.
- [6] N. De Lange. *Geoinformatik in Theorie und Praxis*. Springer, 2005. ISBN 9783540282914.
- [7] C. Duguay, A. Soliman, S. Hachem, and W. Saunders. Circumpolar and regional land surface temperature (1st) with links to geotiff images and netcdf files. 2012. doi:10.1594/PANGAEA.775962. URL <http://dx.doi.org/10.1594/PANGAEA.775962>. Accessed: 2012-06-15.
- [8] GCOS. GCOS Essential Climate Variables, 2012. URL <http://www.wmo.int/pages/prog/gcos/index.php?name=EssentialClimateVariables>. Accessed: 2012-04-24.
- [9] GeoServer. Styling - GeoServer 2.1.x User Manual, 2012. URL <http://docs.geoserver.org/stable/en/user/styling/index.html>. Accessed: 2012-05-08.

- [10] A. M. Ibanez, J. M. Honduvilla, M. Callejo, and M. Angel. Interactive style generation for layer visualization through a WMS, 2005.
- [11] R. Jackson, L.W. MacDonald, and K. Freeman. *Computer generated colour: a practical guide to presentation and display*. Wiley professional computing. J. Wiley, 1994. ISBN 9780471935995.
- [12] Kraak, M. The role of the map in a Web-GIS environment. *Journal of Geographic Systems*, 6:83–93, 2004.
- [13] M. Lupp. Styled Layer Descriptor profile of the Web Map Service Implementation Specification, 2007. URL <http://www.opengeospatial.org/standards/sld>. Accessed: 2011-09-04.
- [14] OGC. Styled Layer Descriptor: Specification, 2010. URL <http://schemas.opengis.net/sld/1.0.0/StyledLayerDescriptor.xsd>. Accessed: 2011-09-04.
- [15] OpenGeo. OpenGeo Website, 2012. URL <http://opengeo.org/>. Accessed: 2012-05-31.
- [16] C. Paulik, T. Melzer, S. Hahn, A. Bartsch, B. Heim, K. Elger, and W. Wagner. Circumpolar surface soil moisture and freeze/thaw surface status remote sensing products with links to geotiff images and netCDF files. 2012. doi:10.1594/PANGAEA.775959. URL <http://doi.pangaea.de/10.1594/PANGAEA.775959>. Accessed: 2012-06-15.
- [17] D. Sabel, S. Park, A. Bartsch, S. Schlaffer, J. Klein, and W. Wagner. Regional surface soil moisture and freeze/thaw timing remote sensing products with links to geotiff images. 2012. doi:10.1594/PANGAEA.779658. URL <http://dx.doi.org/10.1594/PANGAEA.779658>. Accessed: 2012-06-15.
- [18] M. Santoro and T. Strozzi. Circumpolar digital elevation models > 55 degree N with links to geotiff images. 2012. doi:10.1594/PANGAEA.779748. URL <http://dx.doi.org/10.1594/PANGAEA.779748>. Accessed: 2012-06-15.
- [19] M. Santoro and T. Strozzi. ALOS Digital Elevation Models with links to geotiff files. 2012. doi:10.1594/PANGAEA.783306. URL <http://dx.doi.org/10.1594/PANGAEA.783306>. Accessed: 2012-06-15.
- [20] S. Schlaffer, D. Sabel, A. Bartsch, and W. Wagner. Regional water bodies remote sensing products with links to geotiff images. 2012. doi:10.1594/PANGAEA.779754. URL <http://dx.doi.org/10.1594/PANGAEA.779754>. Accessed: 2012-06-15.
- [21] Stefan Petre. ColorPicker - jQuery Plugin, 2012. URL <http://www.eyecon.ro/colorpicker/>. Accessed: 2012-05-10.

- [22] T. Strozzi. InSAR Digital Elevation Models for subsidence with link to geotiff files. 2012. doi:10.1594/PANGAEA.783307. URL <http://dx.doi.org/10.1594/PANGAEA.783307>. Accessed: 2012-06-15.
- [23] A. Weiser. Automatisierte Generierung von Styled Layer Descriptor-Dateien aus ESRI ArcGIS-Projekten zur Publikation mit OGC-konformen Mapservern. Master's thesis, University of Applied Science Mainz, 2005.