COGEAR

MODULE 1:

Description of the available data and the database structure

Del. No.: 1b.1.1

Author: Schnabel, O.

Institute of Cartography, ETH Zürich

October 26, 2009

Institute of Cartography (IKA)
ETH Zurich
CH-8093 Zurich
www.karto.ethz.ch

COGEAR Report

Description of the available data and the database structure

Task 1b.1: Development and management database
Deliverable 1b.1.1 and 1b.1.2: Database implementation prototype and data

October 2009

Table of Contents

Problem Statement

The COGEAR project aims to provide a platform for the deployment of sensor networks and data retrieval in the field of seismology and its related fields.

The achieve this goal, different systems and instruments will be used for the data request and data exchange from the sensors in the field. To unify the data and metadata from different sources and visualize them in a common way, the Institute of Cartography (IKA) at ETH Zurich developed a "distributed database" as GIS and service platform for COGEAR (task of COGEAR module 1b).

Concept of the Distributed Database

Monolithic databases have three disadvantages: First, they tend to become very huge since various data providers insert their data. Second, they have a fixed structure. This is a problem since each data provider stores his data in a specific way which makes sense for his specific scientific task. Therefore, he has to adapt the data to the needs of the database which means usually a huge additional effort. As third disadvantage, the data provider has to store the same data twice (once locally for his own scientific purpose, once for the monolithic database).

Therefore, the IKA (ETHZ) developed the concept of a distributed database. The main idea of this concept is the distributed storage of the data in different databases. Each of these databases are located in the domain of a single data provider and is maintained by this provider. This helps to keep the maintenance costs low and simplifies the workflow of the data provider since he has to store his data only once. As additional benefit, the data of other providers are not effected if one data source fails.

Implementation of the Distributed Database

To implement the above mentioned concept, IKA proposed the open source database PostGreSQL with spatial extension PostGIS as database software. According to the COGEAR workplan, IKA provided support for the generation of database schemas (in PostgreSQL) of the project partners and developed an own database schema for its data. The collected database schemas are listed under "Available Database Schemas" and exemplarily in the annexes. Until now, they are still in development and might be modified. The IKA is in contact with the other project partners to increase the available data sources progressively. Therefore, a higher number of accessible data can be expected in the future, depending on the available data of the project partners.

Moreover, IKA has created a geodatabase for the swisstopo pixel maps and the geological atlas to support the seismological research activities of the COGEAR project. For this task, IKA tested and used the latest commercial products such as the database IBM DB2 9.1 with spatial extension Spatial Extender, the spatial engine ESRI ArcSDE 9.3.1 and the mapserver ESRI ArcGIS Server 9.3.1 in order to prove the low-level database technology independence and the flexibility of the distributed database testbed and concept. The resulting geodatabase and WMS (see chapter "COGEAR Workflow") could be smoothly

integrated in the COGEAR infrastructure. In this context, IKA provided the following datasets for entire Switzerland:

Dataset	Data Type	approx. Size (MB)
pixel map 1:25.000, combined	GeoTIFF	6000
pixel map 1:50.000, combined	GeoTIFF	2000
pixel map 1:100.000, combined	GeoTIFF	500
pixel map 1:200.000, combined	GeoTIFF	200
pixel map 1:500.000, combined	GeoTIFF	70
pixel map 1:1.000.000, combined	GeoTIFF	40
Geological Atlas 1:25.000	GeoTIFF	12500

As additional service, IKA integrated adaptive zooming functionality in the WMS to provide scale-dependent background maps for the project partners.

Available Database Schemas

The following database schemas could be created and collected (Figure 2 shows an UML-based view of the PRS database schema):

Project Partner	Type of Data
SED (ETHZ)	seismic stations, event data
IKA (ETHZ)	swisstopo pixel maps (all scales), Geological Atlas
IGT (ETHZ)	borehole data
EngGeol (ETHZ)	cracks, soil temperature sensors at Randa
PRS (ETHZ)	LiDAR data (aerial and helicopter based), aerial images, UAV images of Randa rockfall, DTM-AV, DEM25
LMS (EPFL)	borehole data
IMAC (EPFL)	vibration records, survey of Visp

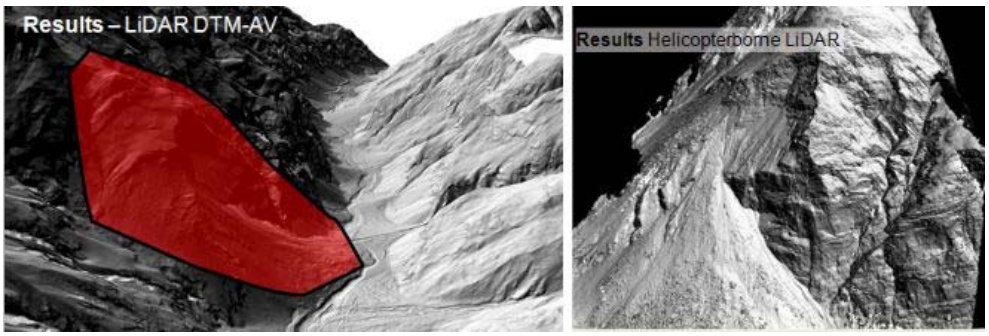

Figure 1: Airborne LiDAR, Helicopterborne LiDAR

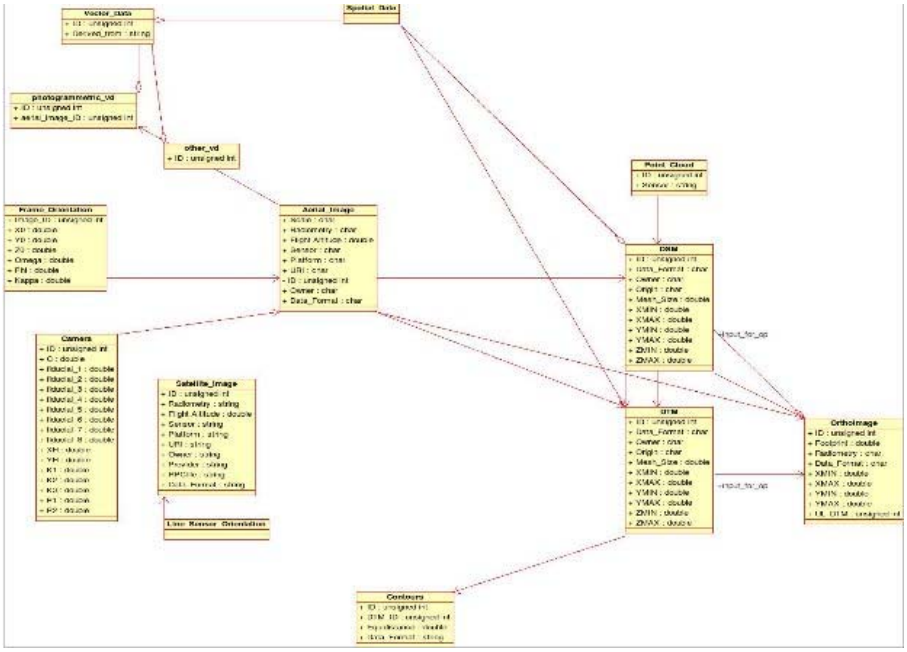

Figure 2: PRS database schema

COGEAR Workflow

Based on the distributed databases as data sources, a 2-step-visualization service workflow was designed by IKA. This new workflow for COGEAR (and the umbrella project SwissExperiment) is shown in Figure 3.

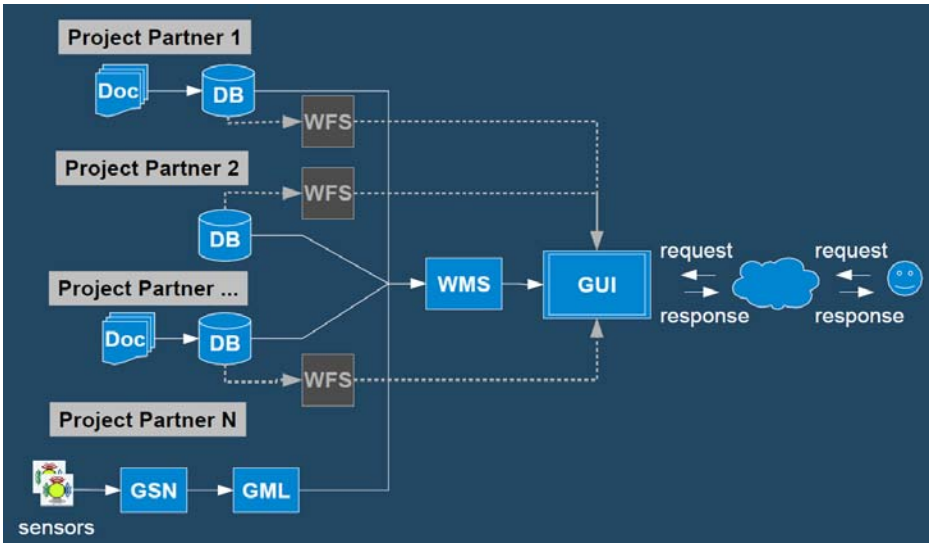

Figure 3: Workflow of the COGEAR project

In a first step, Web Map Services (WMS) will be build on top of the distributed databases of the project partners. These WMS can whether be located in the domain of the single project partners or in the domain of IKA. WMS are standardized interfaces to provide data and metadata as maps (GetMap request) and tables (GetCapabilities request). These WMS can be combined in a single WMS and integrated in a Graphical User Interface

(GUI). Via GUI elements such as buttons and sliders the user can navigate in the data of all project partners. Additionally, interactions with the data are possible. In the first step, this is the request of metadata from single map elements.

In a second step, Web Feature Services (WFS) will be build on top of the distributed databases of the project partners. WFS are standardized interfaces to share the data as vectors. Also these WFS should be made available via the GUI.

Implementation of the COGEAR Workflow

To implement the above mentioned workflow, IKA used and extended the software QGIS Mapserver. QGIS Mapserver is an open source Web Map Services 1.3 implementation. In addition, it implements advanced cartographic features as specified in the Map and Diagram Service specifications. In short, the content of vector and raster datasources (e.g. Shapefiles, GML, PostGIS, WFS, GeoTIFF) is visualized according to cartographic rules (specified as request parameters). The generated map is sent back to the client over the internet.

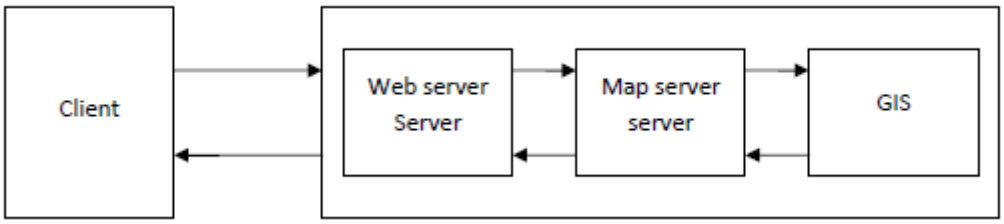

Figure 4: Client-Server-Architecture

Until now, QGIS Mapserver supports the following features:

- WMS (web map service) via HTTP GET. Supports GetCapabilities, GetMap, GetFeatureInfo and custom styling with Styled Layer Descriptor (SLD)
- SOAP via HTTP POST. Compatible with the ORCHESTRA and SANY Service Oriented Architecture (both are 3rd party funding EU projects)
- Native configuration with SLD. User friendly map symbolisation with QGIS Desktop and PublishtoWeb plugin
- Cartographic extensions to SLD (diagrams, patterns and custom symbols with Scalable Vector Graphics). Exchange of cartographic rules with the GetStyle operation.

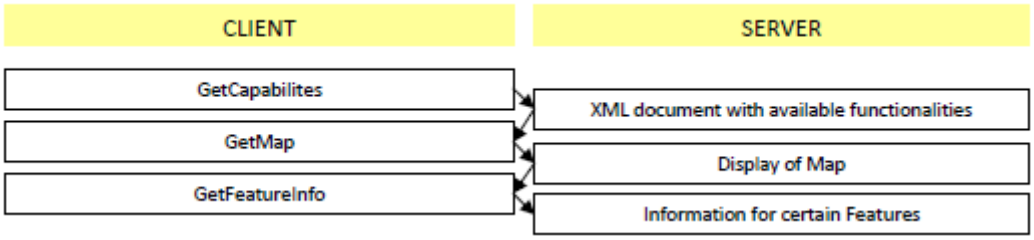

Figure 5: Structure of a WMS request

In a first step, QGIS Mapserver was used to produce WMS from the available databases of the project partners. These WMS were visualized separately in testbeds with a simple OpenLayers GUI. These testbeds were made available to the project partners. They can use it to navigate in their data and even request metadata of single objects.

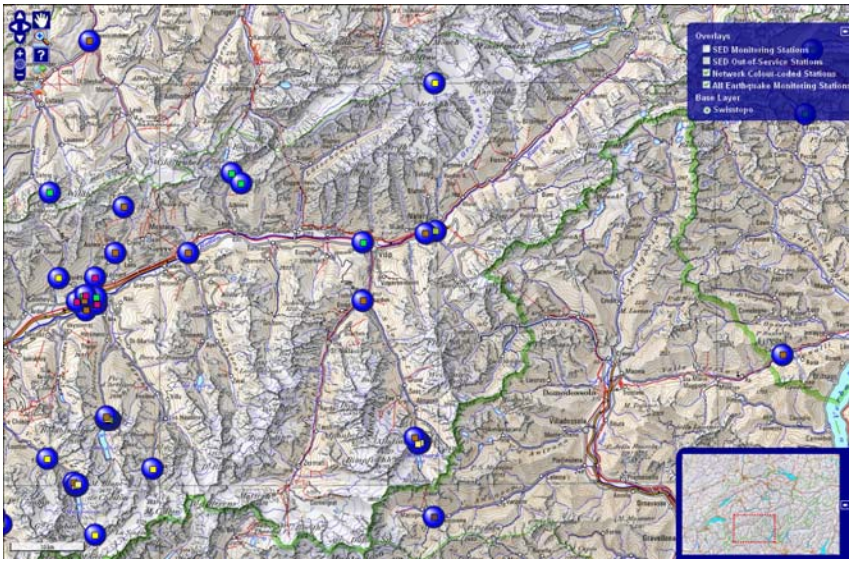

Figure 6: SED data combined with swisstopo pixel map (provided by IKA)

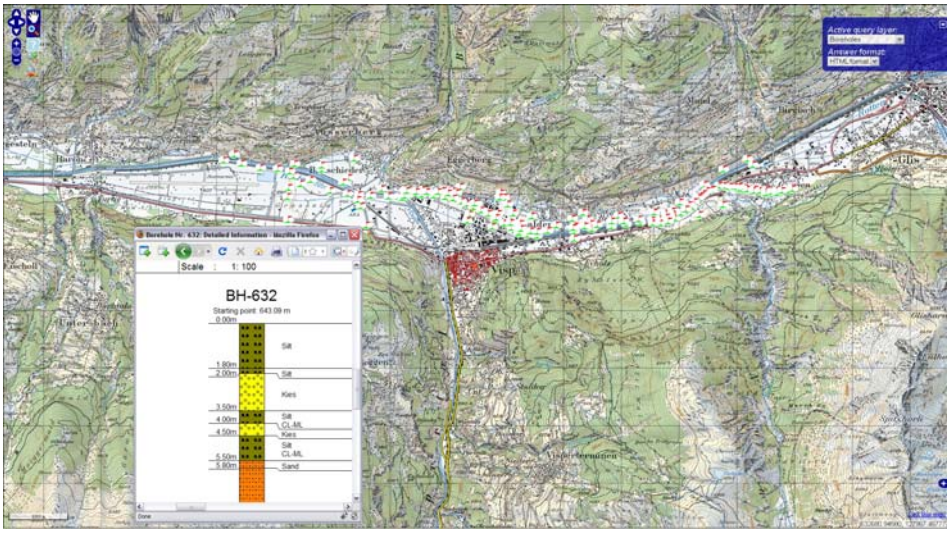

Figure 7: IGT data combined with swisstopo pixel map (provided by IKA)

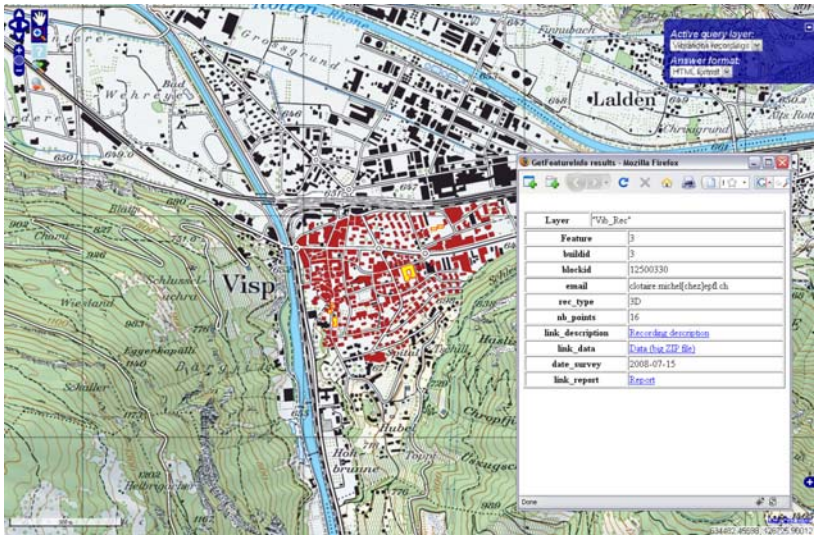

Figure 8: IMAC data combined with swisstopo pixel map (provided by IKA)

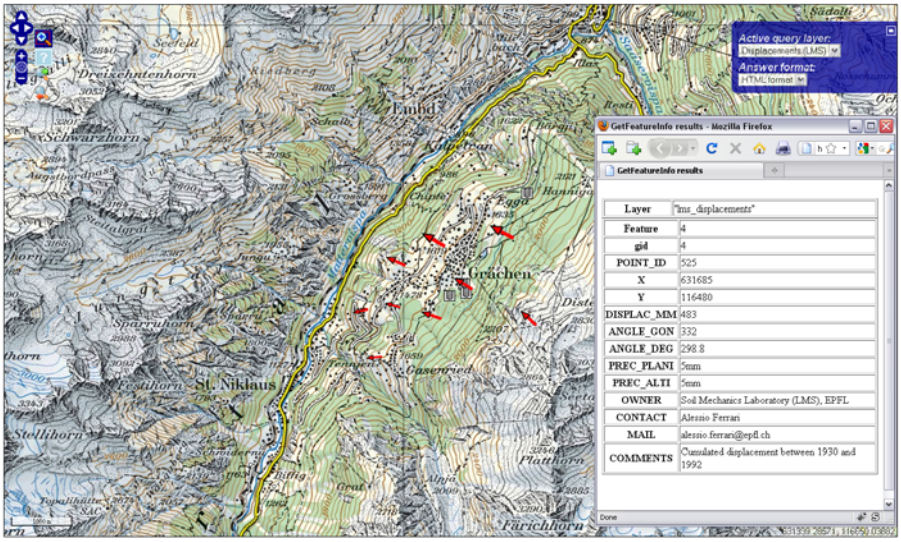

Figure 9: LMS data combined with swisstopo pixel map (provided by IKA)

Furthermore, IKA combined the testbeds in an additional testbed as proof of concept for the interinstitutional interoperability (see Figure 10). The unification of the schemas and database domains is performed visually in this testbed, and allows project partners to manage and evolve their schemas according to their needs.

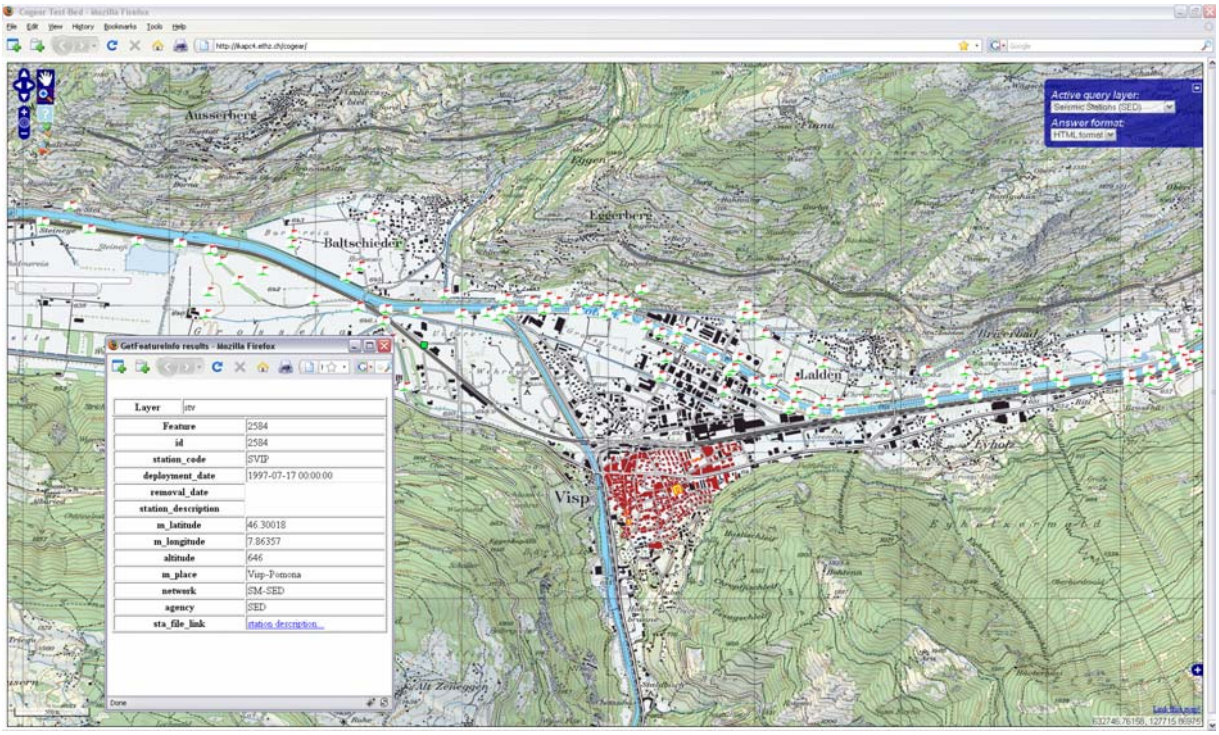

Figure 10: Combined testbed with IGT, IMAC and IKA data

Current Tasks

Since some project partners plan to use GSN as sensor and instrumental platform, IKA currently works on the integration of GSN sensors in the workflow (which is also a SwissExperiment task). Furthermore, the GeoVITe GUI is currently adapted to the COGEAR needs. The proposed GUI for COGEAR can be seen in Figure 11.

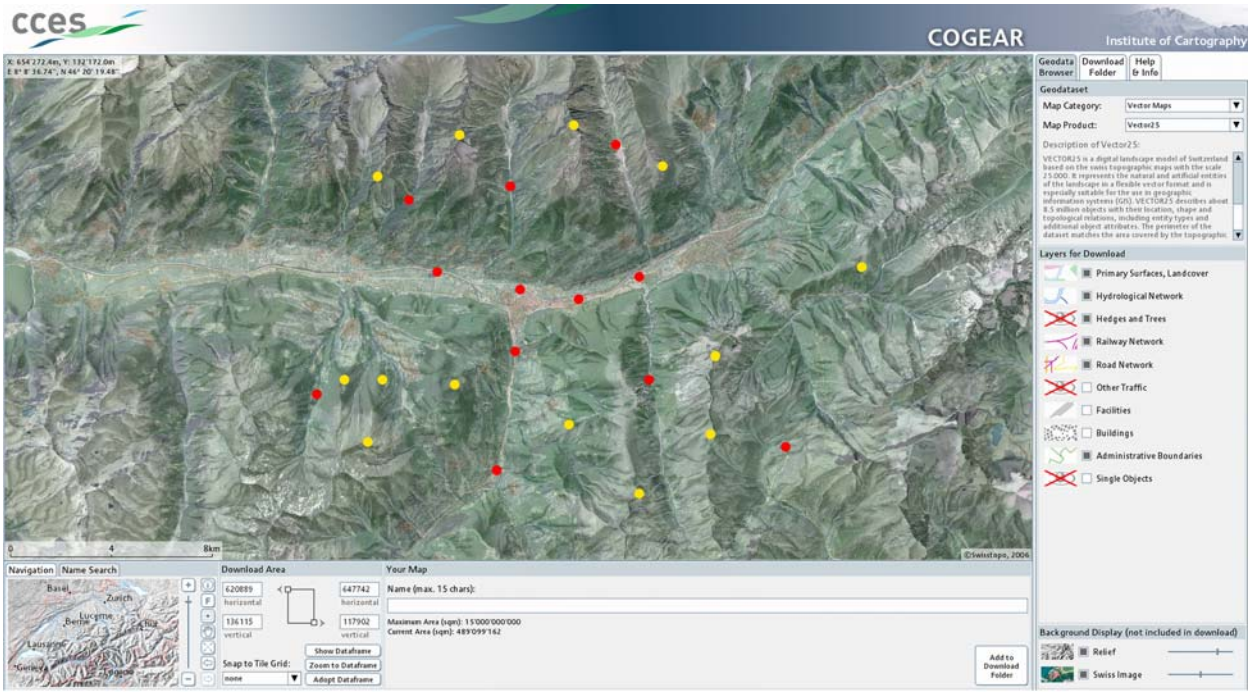

Figure 11: Proposed GUI for COGEAR

Annex 1: Extract of the SED database schema

```
--
-- TOC entry 1973 (class 1259 OID 26219)
-- Dependencies: 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 5
-- Name: amplitude; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE amplitude (
    _oid bigint NOT NULL,
    _parent_oid bigint NOT NULL,
    _last_modified timestamp without time zone DEFAULT now(),
    m_amp_value double precision NOT NULL,
    m_amp_uncertainty double precision,
    m_amp_loweruncertainty double precision,
    m_amp_upperuncertainty double precision,
    m_type character varying(10) DEFAULT ''::character varying NOT NULL,
    m_period_value double precision NOT NULL,
    m_period_uncertainty double precision,
    m_period_loweruncertainty double precision,
    m_period_upperuncertainty double precision,
    m_pickid character varying(80) DEFAULT ''::character varying NOT NULL,
    m_waveformid_networkcode character varying(2) DEFAULT ''::character varying NOT NULL,
    m_waveformid_stationcode character varying(5) DEFAULT ''::character varying NOT NULL,
    m_waveformid_channelcode character varying(3) DEFAULT ''::character varying NOT NULL,
    m_waveformid_locationcode character varying(2) DEFAULT ''::character varying NOT NULL,
    m_waveformid_resourceuri character varying(80) DEFAULT ''::character varying NOT NULL,
    m_waveformid_used numeric(1,0) DEFAULT 0 NOT NULL,
    m_filterid character varying(80) DEFAULT ''::character varying NOT NULL,
    m_methodid character varying(80) DEFAULT ''::character varying NOT NULL,
    m_timewindow_begin double precision DEFAULT (0)::double precision NOT NULL,
    m_timewindow_end double precision DEFAULT (0)::double precision NOT NULL,
    m_timewindow_reference timestamp without time zone NOT NULL,
    m_timewindow_reference_ms integer NOT NULL,
    m_timewindow_used numeric(1,0) DEFAULT 0 NOT NULL,
    m_scalingtime_value timestamp without time zone NOT NULL,
    m_scalingtime_value_ms integer NOT NULL,
    m_scalingtime_uncertainty double precision,
    m_scalingtime_loweruncertainty double precision,
    m_scalingtime_upperuncertainty double precision,
    m_scalingtime_used numeric(1,0) DEFAULT 0 NOT NULL,
    m_magnitudehint character varying(10) DEFAULT ''::character varying NOT NULL,
    m_evaluationmode character varying(64),
    m_creationinfo_agencyid character varying(32) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_agencyuri character varying(80) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_author character varying(32) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_authoruri character varying(80) DEFAULT ''::character varying NOT NULL,
```

```
    m_creationinfo_creationtime timestamp without time zone,
    m_creationinfo_creationtime_ms integer,
    m_creationinfo_version character varying(48) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_used numeric(1,0) DEFAULT 0 NOT NULL
);


--
-- TOC entry 1974 (class 1259 OID 26246)
-- Dependencies: 2394 2395 5
-- Name: amplitudereference; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE amplitudereference (
    _oid bigint NOT NULL,
    _parent_oid bigint NOT NULL,
    _last_modified timestamp without time zone DEFAULT now(),
    m_amplitudeid character varying(80) DEFAULT ''::character varying NOT NULL
);


--
-- TOC entry 1975 (class 1259 OID 26250)
-- Dependencies: 2396 2397 2398 2399 2400 2401 2402 2403 2404 5
-- Name: arrival; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE arrival (
    _oid bigint NOT NULL,
    _parent_oid bigint NOT NULL,
    _last_modified timestamp without time zone DEFAULT now(),
    m_pickid character varying(80) DEFAULT ''::character varying NOT NULL,
    m_phase_code character varying(20) NOT NULL,
    m_timecorrection double precision,
    m_azimuth double precision,
    m_distance double precision,
    m_residual double precision,
    m_weight double precision,
    m_earthmodelid character varying(80) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_agencyid character varying(32) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_agencyuri character varying(80) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_author character varying(32) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_authoruri character varying(80) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_creationtime timestamp without time zone,
    m_creationinfo_creationtime_ms integer,
    m_creationinfo_version character varying(48) DEFAULT ''::character varying NOT NULL,
    m_creationinfo_used numeric(1,0) DEFAULT 0 NOT NULL
);
```

Annex 2: Extract of the LMS database schema

```
--
-- Name: lms_boreholes; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE lms_boreholes (
    gid integer NOT NULL,
    "CODE_VS" double precision,
    "OTHER_NAME" character varying(10),
    "X" double precision,
    "Y" double precision,
    "Z" double precision,
    "NR_COMMUN" double precision,
    "NAME_COMM" character varying(8),
    "DATE_BORE" character varying(10),
    "TYPE_PER_F" character varying(17),
    "TYPE_PER_D" character varying(20),
    "T_DIAM_FIN" double precision,
    "BORELENGTH" double precision,
    "BORE_INCLI" double precision,
    "SITE" character varying(25),
    "ALTI_WATER" character varying(17),
    "DATE_MEASR" character varying(10),
    "BORETYPE_F" character varying(23),
    "BORETYPE_D" character varying(15),
    "GOAL" character varying(14),
    "ZQUALITY" character varying(13),
    "COMMENT" character varying(192),
    the_geom geometry,
    CONSTRAINT enforce_dims_the_geom CHECK ((ndims(the_geom) = 2)),
    CONSTRAINT enforce_geotype_the_geom CHECK (((geometrytype(the_geom) = 'POINT'::text) OR (the_geom IS NULL))),
    CONSTRAINT enforce_srid_the_geom CHECK ((srid(the_geom) = 21781))
);


--
-- Name: lms_boreholes_attrdata; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE lms_boreholes_attrdata (
    gid integer NOT NULL,
    "CODE_VS" double precision,
    "DEPTH_MIN" double precision,
    "DEPTH_MAX" double precision,
    "CODE_LITH1" double precision,
    "DESCRIP1_F" character varying(7),
    "DESCRIP1_D" character varying(4),
    "CODE_LITH2" double precision,
```

```
    "DESCRIP2_F" character varying(5),
    "DESCRIP2_D" character varying(4),
    "CODE_LITH3" double precision,
    "DESCRIP3_F" character varying(7),
    "DESCRIP3_D" character varying(4),
    "CODE_LITH4" double precision,
    "DESCRIP4_F" character varying(10),
    "DESCRIP4_D" character varying(11),
    "PERMEABIL" character varying(11),
    "ATTR_F" character varying(8),
    "ATTR_D" character varying(8),
    "GEOLOGY_F" character varying(36),
    "GEOLOGY_D" character varying(39)
);


--
-- Name: lms_displacements; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE lms_displacements (
    gid integer NOT NULL,
    "POINT_ID" double precision,
    "X" double precision,
    "Y" double precision,
    "DISPLAC_MM" double precision,
    "ANGLE_GON" double precision,
    "ANGLE_DEG" double precision,
    "PREC_PLANI" character varying(3),
    "PREC_ALTI" character varying(3),
    "OWNER" character varying(37),
    "CONTACT" character varying(15),
    "MAIL" character varying(23),
    "COMMENTS" character varying(44),
    the_geom geometry,
    CONSTRAINT enforce_dims_the_geom CHECK ((ndims(the_geom) = 2)),
    CONSTRAINT enforce_geotype_the_geom CHECK (((geometrytype(the_geom) = 'POINT'::text) OR (the_geom IS NULL))),
    CONSTRAINT enforce_srid_the_geom CHECK ((srid(the_geom) = 21781))
);
```

Annex 3: Extract of the GEOLENG database schema

```
--
-- TOC entry 1585 (class 1259 OID 394330)
-- Dependencies: 1926 1927 1928 1 597
-- Name: enggeol_grounds; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE enggeol_grounds (
    gid integer NOT NULL,
    "GROUND_ID" character varying(7),
    "X_CH" double precision,
    "Y_CH" double precision,
    "X_WORLD" double precision,
    "Y_WORLD" double precision,
    "OWNER" character varying(42),
    "CONTACT" character varying(10),
    "MAIL" character varying(26),
    "MAINTN_BY" character varying(30),
    the_geom geometry,
    CONSTRAINT enforce_dims_the_geom CHECK ((ndims(the_geom) = 2)),
    CONSTRAINT enforce_geotype_the_geom CHECK (((geometrytype(the_geom) = 'POINT'::text) OR (the_geom IS NULL))),
    CONSTRAINT enforce_srid_the_geom CHECK ((srid(the_geom) = 21781))
);


SET default_with_oids = true;
--
-- TOC entry 1587 (class 1259 OID 394355)
-- Dependencies: 1654 597 1
-- Name: grounds; Type: VIEW; Schema: public; Owner: -
--
CREATE VIEW grounds AS
    SELECT enggeol_grounds.gid, enggeol_grounds.the_geom, enggeol_grounds."GROUND_ID", enggeol_grounds."X_CH",
enggeol_grounds."Y_CH", enggeol_grounds."X_WORLD", enggeol_grounds."Y_WORLD", enggeol_grounds."OWNER",
enggeol_grounds."CONTACT", enggeol_grounds."MAIL", enggeol_grounds."MAINTN_BY" FROM enggeol_grounds;


--
-- TOC entry 1588 (class 1259 OID 394358)
-- Dependencies: 1655 597 1
-- Name: sensor_grounds; Type: VIEW; Schema: public; Owner: -
--
CREATE VIEW sensor_grounds AS
    SELECT enggeol_grounds.gid, enggeol_grounds.the_geom, enggeol_grounds."GROUND_ID", enggeol_grounds."X_CH",
enggeol_grounds."Y_CH", enggeol_grounds."X_WORLD", enggeol_grounds."Y_WORLD", enggeol_grounds."OWNER",
enggeol_grounds."CONTACT", enggeol_grounds."MAIL", enggeol_grounds."MAINTN_BY", ((((('<a
href="http://karlinapp.ethz.ch/enggeol/sensorsinfo.php?i='::character varying)::character varying(60))::text ||
(enggeol_grounds."GROUND_ID")::text) || (('">sensors</a>'::character varying)::character varying(20))::text) AS sensorsinfo FROM
enggeol_grounds;


SET default_with_oids = false;
```

```
--
-- TOC entry 1586 (class 1259 OID 394341)
-- Dependencies: 1
-- Name: sensorsinfo; Type: TABLE; Schema: public; Owner: -; Tablespace:
--

CREATE TABLE sensorsinfo (
    gid integer NOT NULL,
    "GROUND_ID" character varying(7),
    "SENSOR_ID" character varying(17),
    "S_TYPE" character varying(20),
    "S_MODEL" character varying(26),
    "S_INFO" character varying(39),
    "INFO_LINK" character varying(51)
);
```

Annex 4: Extract of the IGT database schema

```
--
-- Name: igt_boreholes; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE igt_boreholes (
    gid integer NOT NULL,
    "CODE_VS" double precision,
    "CODE_VS_TX" character varying(12),
    "OTHER_NAME" character varying(25),
    "X" double precision,
    "Y" double precision,
    "Z" double precision,
    "NR_COMMUN" double precision,
    "NAME_COMM" character varying(12),
    "DATE_BORE" character varying(10),
    "ID_PERFOR" double precision,
    "TYPE_PER_F" character varying(20),
    "TYPE_PER_D" character varying(21),
    "TOOLDIAMTR" double precision,
    "T_DIAM_FIN" double precision,
    "BORELENGTH" double precision,
    "BORE_INCLI" double precision,
    "DESCRIP" character varying(51),
    "ALTI_WATER" double precision,
    "DATE_MEASR" character varying(10),
    "ARCHIVING" character varying(53),
    "ID_BORETYP" double precision,
    "BORETYPE_F" character varying(33),
    "BORETYPE_D" character varying(32),
    "ID_QUALITY" double precision,
    "QUALITY_F" character varying(12),
    "QUALITY_D" character varying(11),
    "ID_GOAL" double precision,
    "GOAL_F" character varying(31),
    "GOAL_D" character varying(32),
    "SITE" character varying(43),
    "ID_ZQUALIT" double precision,
    "ZQUALITY_F" character varying(12),
    "ZQUALITY_D" character varying(11),
    "SMP_INCOMP" integer,
    "PRB_INCOMP" integer,
    "DATA" integer,
    "COMMENT" character varying(234),
    the_geom geometry,
    CONSTRAINT enforce_dims_the_geom CHECK ((ndims(the_geom) = 2)),
```

```
    CONSTRAINT enforce_geotype_the_geom CHECK (((geometrytype(the_geom) = 'POINT'::text) OR (the_geom IS NULL))),
    CONSTRAINT enforce_srid_the_geom CHECK ((srid(the_geom) = 21781))
);


--
-- Name: igt_boreholes_attr; Type: TABLE; Schema: public; Owner: -; Tablespace:
--
CREATE TABLE igt_boreholes_attr (
    gid integer NOT NULL,
    "CODE_VS" double precision,
    "DEPTH_MIN" double precision,
    "DEPTH_MAX" double precision,
    "CODE_LITH1" double precision,
    "DESCRIP1_F" character varying(7),
    "DESCRIP1_D" character varying(12),
    "CODE_LITH2" double precision,
    "DESCRIP2_F" character varying(7),
    "DESCRIP2_D" character varying(4),
    "CODE_LITH3" double precision,
    "DESCRIP3_F" character varying(12),
    "DESCRIP3_D" character varying(17),
    "CODE_USCS" character varying(17),
    "CLASSIFIC" character varying(8),
    "SAMPLES" character varying(3),
    "PROBE_SPT" character varying(3),
    "OTH_PROBES" character varying(1),
    "PERMEABIL" character varying(1),
    "TYPE_ATTR" character varying(8),
    "ATTR_F" character varying(10),
    "ATTR_D" character varying(9),
    "ID_GEOL" character varying(3),
    "GEOLOGY_F" character varying(59),
    "GEOLOGY_D" character varying(64)
);
```