## Seventh Framework Programme

SEVENTH FRAMEWORK PROGRAMME

**Call FP7-ICT-2009-6**

Project: 247708 - SUDPLAN

Project full title:

# Sustainable Urban Development Planner for Climate Change Adaptation

**Deliverable D 6.2.2**
**Wuppertal Pilot**

**Version 2 (V2)**

Due date of deliverable: 31/08/2011

Actual submission date: 21/12/2011

| | |
|---|---|
| | |
| **Title** | Pilot Description V2 |
| **Creator** | WUP/CISMET |
| **Editor** | Martin Scholl/Stefan Sander |
| **Description** | Brief description of the Wuppertal SUDPLAN pilot implementation in version 2. |
| **Publisher** | SUDPLAN Consortium |
| **Contributors** | WUP |
| **Type** | Text |
| **Format** | application/msword |
| **Language** | EN-GB |
| **Creation date** | 02-12-2011 |
| **Version number** | 1.0 |
| **Version date** | 21-12-2011 |
| **Last modified by** | CISMET (Sascha Schlobinski) |
| **Rights** | Copyright “SUDPLAN Consortium”.<br>During the drafting process, access is generally limited to the SUDPLAN Partners. |
| **Audience** | ☐ internal<br>☐ public<br>☒ restricted,<br>☒ access granted to: EU Commission |
| **Review status** | ☐ Draft<br>☒ WP Manager accepted<br>☐ PMC quality controlled<br>☒ Co-ordinator accepted<br><br>Where applicable:<br>☐ Accepted by the PMC as public document |
| **Action requested** | ☐ to be revised by Partners involved in the preparation of the deliverable<br>☐ to be revised by all SUDPLAN Partners<br>☐ for approval of the WP Manager<br>☐ for approval of the Quality Manager<br>☐ for approval of the Project Co-ordinator<br>☐ for approval of the PMC |
| **Requested deadline** | 31-08-2011 |

| Version | Date | Modified by | Comments |
|---|---|---|---|
| 0.1 | 02-12-2011 | MSc | Document created |
| 0.11 | 14-12-2011 | MSc | Draft |
| 0.2 | 15-12-2011 | SSc | Comments + changes |
| 0.3 | 15-12-2011 | MSc | Additional content + minor formatting |
| 0.4 | 19-12-2011 | SSa | Review of chapter 1-3.1 |
| 0.5 | 20-12-2011 | SSa | Chapter 5 completed |
| 0.6 | 20-12-2011 | MSc | Appendix A added, document formatting |
| 0.61 | 21-12-2011 | SSa | Revision of Chapter 3.3.1+ DYN description update |
| 1.0 | 21-12-2011 | SSc | Draft finalised |
| 1.0 | 21-12-2011 | LGi | Co-ordinator approval |

# Table of Contents

# Index of Figures

# Index of Tables

# 1. Management summary

This document is the second status report of the Wuppertal Pilot Application. The document summarizes the general objectives of the Wuppertal Pilot Application for the 2nd year of the project, reports on the implementation status including preparatory work performed regarding the local models used. The software that has been developed and deployed in the context of the Wuppertal Pilot Application is described to provide an overview on the currently available functionality. In case of GUIs (Graphical User Interfaces) software is documented in the form of augmented screenshots. In case of services and APIs a technical specification is given in Annex A. The implementation status is mapped to task and use cases to give an impression on the progress made. In addition, a separate paragraph reports on the End User involvement in WP6. So far considerable progress has been achieved and the software (with enhancements and additional features) will go live in the course of 2012.

## 1.1. Objectives of the Wuppertal Pilot

In the first year of the project the main focus of the Wuppertal pilot work was the selection of an appropriate local model configuration and the provision of an initial calculation model. The focus of second year's work was on integration. The overall aim was to integrate the whole chain of components involved in the Wuppertal pilot scenario.

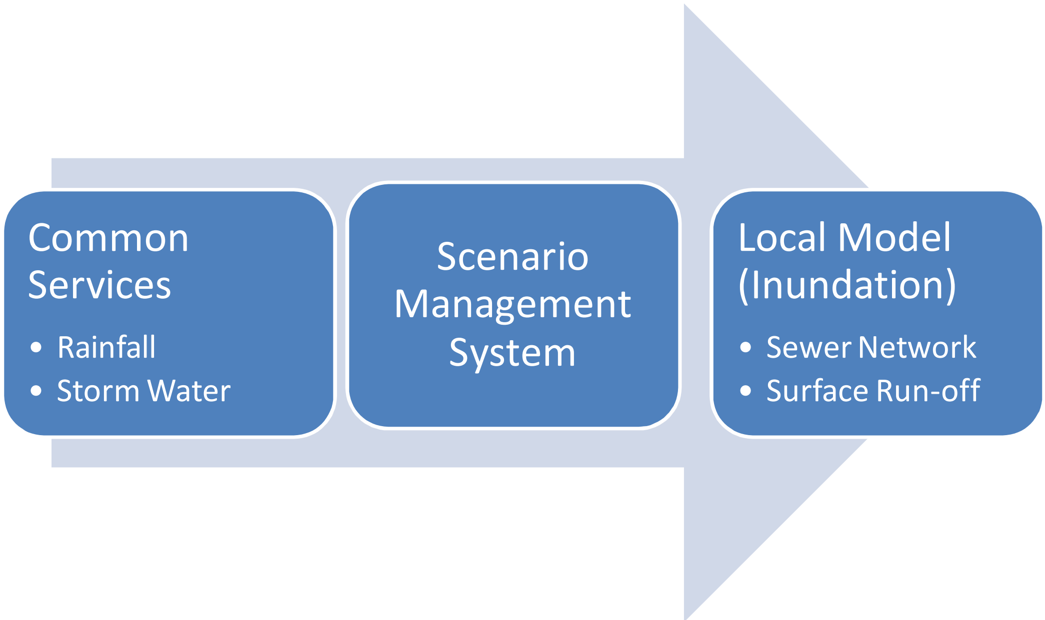


**Figure 1:** Wuppertal application component chain

In addition advanced parameterisation functionalities for the local model have to be designed and developed. The most complex parameter for the modelling of surface run-off is an optimised DEM in the form of a triangulated irregular network (TIN), the so-called 'calculation model'.

The work plan to meet the objectives of the 2nd year included the following activities:

- A concrete concept for the communication with Common Services and the local models needed to be established and implemented. This included

  - o the wrapping of the local model components (Service Wrapping)
  - o a customization of the local models with regard to the pilot requirements (e.g. modification of the topographic model)
- A specialized instance of the SMS had to be deployed that
  - o integrates the local models DYNA and GeoCPM
  - o enables users to change the model parameterization (e.g. raise road kerbs)
  - o provides a workflow support for the planning process
  - o enables users to include climate change information through the integration of Common Services
  - o supports appropriate visualization methods

After the 2nd year activities the resulting Wuppertal application is a working prototype and the basis for 2nd year validation. Advanced visualization (flooding 4D), enhanced workflow support, integration of design storm functionality will be part of the work to be performed in the 3rd year.

## 1.2. Concrete Implementation Status Summary

The current implementation status of the Wuppertal Pilot Application can be summarized as follows:

- Local Model component updated:

  The GeoCPM model component has been enhanced by a so called breakline support. This way the model configurations may contain further information about possible breaking edges so that planners can easily identify and manipulate them.

- Local Model configuration created:

  The existing model configurations have been updated according to the new breakline feature of the model component.

- Local Model configuration import and export facilities:

  The model configuration and the model results are is stored in files by default. To enable smooth visualisation and manipulation the input and output data is imported into a relational database. Exportability of this data enables further processing by means of external tools including the possibility to run simulations with the configurations that are manipulated through the SMS.

- Local Model services implemented and integrated:

  The SMS must be capable of running a local model. Hence a REST based web service has been developed that encapsulates the model code and provides an interface for model control such as model execution and status monitoring.

- Local Model input visualisation and manipulation:

  The decision to use a relational database as a backend for the model data is highly valuable for the input visualisation. The calculation model, that needs to be visualised, includes the TIN and the breaklines. The TIN data now is offered by a WMS (Web Map Service) which is a standard-based OGC Service and thus usable in a variety of use cases. The breaklines are offered by the cids application server thus supporting easy access and manipulation.

- Local Model result visualisation

  Model results are picked up and stored in the database. That way the cids application server can provide the raw results to the SMS user when needed. Moreover, a WMS can be used to easily process the results and to provide a meaningful visual representation of the simulation results. Additionally, the 3D component may use a fast and standards-based way to process the results for the 3D visualisation via WFS or native SQL.

 

# 2. Preparatory Work

In the second year some refinements regarding the local model software have been necessary to allow advanced model parameterization features in the context of the Wuppertal Pilot Application.

## 2.1. Dynamic Modifications in the Calculation model

In the SUDPLAN Wuppertal pilot GeoCPM is used to calculate the stormwater surface run-off in model runs created with the SMS. However, in its original version the GeoCPM model did not provide easy means to identify and manipulate so-called breaklines. Breaklines are man-made edges that result from real world construction. The most significant classes of such breaklines are the exterior walls of buildings ('building breaklines') and road kerbs or similar vertical structures ('road kerb breaklines'). As the main focus of the Wuppertal pilot is on running various surface run-off simulations not only by altering the stormwater event itself but also by simulating impacts of different urban development measures the possibility to easily manipulate breaklines is a main feature. As a result the GeoCPM model component has been enhanced.

Basically the configuration of breaklines within the triangulated irregular network (TIN) has already been possible. The creation or manipulation of a break line was done on a per-triangle basis. However, there was no possibility to aggregate these edges to a continuous break line, which represents e.g. the road kerb of a whole street. Consequently planners were not able to easily recognise and manipulate aggregated edges. Moreover, the possibility of free manipulation of triangles in terms of break line manipulation leads to two other problems:

1. Planners are easily overwhelmed by the sheer amount of manipulable objects within an investigation area.

2. There is no possibility to counter the creation of breaklines where they simply do not make any sense (e.g. in the middle of a road).

To overcome these issues the GeoCPM model component has been extended by the possibility to define compound breaklines that basically consist of all triangles somehow related to this very edge and a unique value for the height of the edge. That way modelling experts and planners can define meaningful breaklines during the course of the initial creation of the TIN. Planners can later on simulate various urban development measures by altering few breakline handles instead of multiple single triangles.

# 3. Inventory of data used

In the Wuppertal Pilot Application numerous data source have been integrated so far. This chapter gives an overview on the data relevant for the Wuppertal Pilot Application.

## 3.1. Common Services Data

The Wuppertal pilot will uses the two Common Services concerning precipitation that have to be established by SMHI to simulate intense short-term rainfall under the predicted future climatic conditions:

- Intense rainfall: Urban downscaling both with IDF-curves and time series as input
- Intense rainfall: Design storm generator

To use the urban downscaling service it is necessary to provide historical high resolution precipitation data for the area that your simulation covers. Alternatively the urban downscaling service can be used with IDF curve data. The downscaled data provided by the Common Services has the same properties (resolution, …) as the local data that is provided.

## 3.2. Local Times Series and IDF-Curves

The required rainfall data is available for two gauging stations in Wuppertal:

- Wastewater treatment plant ‘Buchenhofen’ in the western part of Wuppertal, keeping records since January 1$^{st}$ 1960, operated by Wupperverband
- Wastewater treatment plant ‘Schwelm’ close to the eastern boundary of Wuppertal, keeping records since November 2$^{nd}$ 1970, operated by Wupperverband

There are various other gauging stations in Wuppertal, but they were put into operation much later. Therefore the two stations with long-term records are considered to be the most valuable input for the urban downscaling service.

Though the gauges are operated by the Wupperverband it is possible to access them via an information system that belongs to the Wuppertaler Stadtwerke (WSW), the municipal utility that runs the sewage system on behalf of the City of Wuppertal. This information system is based upon the software AquaZIS by the company aqua_plan GmbH, a tool for the management and evaluation of time series data. Both Wupperverband and WSW have agreed to provide the required precipitation data for use in the SUDPLAN project.

IDF-curve data can be provided via two ways. The first is to use a published catalogue of IDF-curves, created by statistical analysis of historical precipitation data. For Germany the most prominent data source of this kind is the KOSTRA Atlas (KOSTRA-DWD-2000). The second way is to compute local IDF-curves from local long time rainfall series (extreme value statistics).

## 3.3. Other Relevant Data

In the SUDPLAN Wuppertal pilot GeoCPM will be used to calculate the stormwater surface run-off in model runs defined with the SMS. The most complex input parameter for such a model run - besides the simulated precipitation data - is an optimised high-resolution digital elevation model (DEM), the so-called ‘calculation model’. GeoCPM expects this in the form of a triangulated irregular network (TIN). The labour-intensive step in creating a calculation model is to define all the relevant man-made breaklines like the exterior walls of buildings (‘building breaklines’) and road kerbs or similar vertical structures (‘road kerb breaklines’).

To achieve further accuracy GeoCPM will mostly be used in conjunction with DYNA. DYNA can perform hydrodynamic calculations of sewer systems and thus requires a modelled sewer system as an input. The WSW and the City of Wuppertal are using DYNA for several years now, therefore a set of DYNA input parameter files is available that describes the current state of Wuppertal’s sewer network. It consists of the main parameter file DYNA.EIN and two more optional files. These files can be used as initial point for all combined model runs of GeoCPM and DYNA.

Meanwhile the modelling of the surface run-off during heavy stormwater events has been introduced into the continuous planning process of Wuppertal’s sewer system, called ‘Generalentwässerungsplanung‘ (General Drainage Strategy). This is an important achievement, because it implements the workflow the SUDPLAN SMS covers.

### 3.3.1 GeoCPM

So far four GeoCPM calculation models have been set up during the term of SUDPLAN. They refer to critical spots in the named catchment areas:

- Area around the ‘Lüntenbeck Manor House’ (catchment area of the stream ’Lüntenbeck’)
- Area around furniture industry in the street ‘Otto-Hausmann-Ring’ (catchment area of the stream ’Varresbeck’)
- Private site in the street ‘In der Beek’ (catchment area of the stream ’Varresbeck’)
- Private site in the street ‘Hintersudberger Straße’ (catchment area of the stream ‘Morsbach’)

The calculation model consists of different sections:

- Configuration: Section where the model calculation parameters can be set
- Points: This section contains all point (x,y,z) definitions of which the DEM consists
- Triangle: The previously defined points are used to define the triangles of the TIN
- Curves: GeoCPM allows the definition of hydraulic curves
- Source-Drain: Section where the inflow or drainage of a specific curve at a specific triangle of the TIN can be defined

- Manholes: GeoCPM takes manholes into account. This is especially important for combined Dyna/GeoCPM runs.
- Marked: Hydraulic results calculation can be constrained on a specific ‘marked’ subset of triangles.
- Raincurve: This section contains the rain event that is to be simulated. However, in case of a combined run this section will be empty because Dyna will provide rain data.
- BK-Connect: Compound breaklines can be defined here

For a more detailed and complete documentation see the GeoCPM Interface Specification [3].

### 3.3.2 DYNA

The Wuppertal pilot application will not be used to simulate different planning options for the sewer network, hence these parameter files can to a large extent be regarded as static for all surface run-off simulations in Wuppertal. The only variable parameter in the DYNA parameter files is the description of the rainfall event that is used for a combined model run of DYNA and GeoCPM. This rainfall event has to be described in DYNA’s main input parameter file ‘DYNA.EIN’, cf. [4] for detailed information on this topic. If a model run uses only GeoCPM (meaning no consideration of the sewer system in the simulation) the description of the rainfall event is expected in the GeoCPM input parameter file ‘GEOCPM.EIN’.

# 4. Pilot Application Design V2

In contrast to the Pilot V1 design, where the assessment of critical areas was focussed, the design of V2 mainly aims at the implementation of a well-defined and use case driven model execution workflow. This includes user friendly ways to manage and to find data as well as enriching visualisations of input and output data. Moreover the integration of Rainfall Common Services is a main topic.

## 4.1. Relevant Tasks (V2)

The focus of Year 2 work was on the following branch of the Tasks identified in the Pilot Definition Plan for Version 2 of the SUDPLAN Wuppertal pilot (For more details see D6.1.2 Wuppertal Pilot Definition Plan V2).

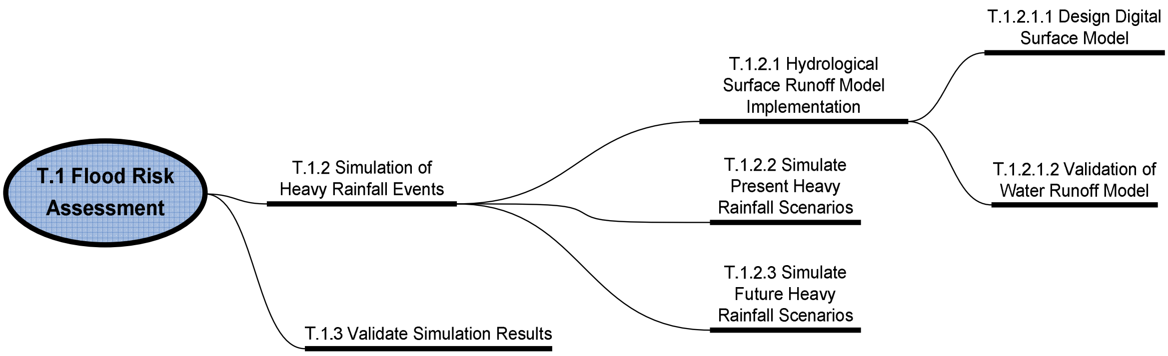


**Figure 2:** Relevant Tasks for Wuppertal Pilot V2

| **T.1.2 Simulation of Heavy Rainfall Events** |
|---|
| The surface water runoff will be simulated with the help of a comprehensive and state of the art hydrological surface run-off model. A multitude of different rain scenarios will be run through the model in order to identify endangered public and private property. The goal of the calculation is to simulate the flow of water and maximum water levels reached in case of heavy rainfall events and flooding. The simulation will not only examine past rain patterns but also analyse possible future rain scenarios based on established climate models. |
| **T.1.2.1 Hydrological Surface Runoff Model Implementation** |
| For the simulation of water run-off the city of Wuppertal must set up a hydrological surface run-off model ('calculation model') in the form of a TIN. Due to the fact that this is no trivial task the usual way to achieve an optimal result is to use an iterative approach of design (computation of the TIN from given 3D points, compare subtask 1.2.1.1) and validation (subtask 1.2.1.2). This means that at the end of each cycle the digital surface model has to be improved based on an evaluation report coming from subtask 1.2.1.2. The improvement may include the introduction of additional high accuracy data (e.g. from terrestrial surveys), the elimination of redundant 3D-points in the laser scan data and the addition of interpolated 3D-points in the laser |

scan data to get an evenly structured TIN. The iteration converges towards a model that is morphological correct and optimized with respect to the calculation time needed for a run of the hydrodynamical surface run-off model.

### T.1.2.1.1 Design Digital Surface Model

In order to use a hydrological surface runoff model a digital surface model of the region of Wuppertal has to be created. The municipality possesses a digital laser scan of the Wuppertal region in form of a laser point cloud where every point has an elevation and classification value. Based on this point cloud an elevation model will be derived. Therefore breaklines of objects such as street kerbs have to be identified with the aid of additional information e.g. streets, paved surfaces, impervious surface etc. The breaklines further refine the elevation model because the additional lines will make the elevation model more exact and necessary to calculate the water run-off. In a second step the not scannable areas like underground crossings, pedestrian underpass etc. need to be incorporated. The method of laser scanning itself has its limitations which manifest in measurement errors, for example the capturing of non surface objects like trees, lamp poles etc. These objects must be removed to avoid distortion of the digital surface model. The resulting digital surface model will be optimised regarding the removal of redundant point information with the goal to reduce the later calculation time of the surface runoff model. This task represents a joint venture between storm water managers, surveyors and model experts and has to be performed in close collaboration.

### T.1.2.1.2 Validation of Water Runoff Model

The accuracy of the surface runoff model results have to be compared to actual available damage records of areas where flooding already occurred. Because a 100 % validation will not be possible and additionally there are no historic data available which cover large areas, individual cases and experiences from the past will be taken into account to judge the quality of the model predictions. Furthermore the model results will be analysed with contextual information such as cadastral maps in order to discover errors in the simulation caused by wrong input parameters such as falsely detected breaklines, distortion in the surface model etc. which cannot be discovered automatically.

### T.1.2.2 Simulate Present Heavy Rainfall Scenarios

Simulations of heavy rainfall over the area of Wuppertal based on historical precipitation data. The surface runoff model is used to predict how the water will distribute over the surfaces. The Storm Water Manager wants the possibility to play through different kind of rain scenarios which imitate possible natural rainfalls or to use real data from past storm events to evaluate the water run-off. The simulation enables the Storm Water Manager to see different attributes, e.g. water run-off velocity, run-off direction, water levels etc., of the precipitation water for specified time steps. Not only can the single steps of the rainfall event be analysed but also an animation how the water will distribute throughout the city over time.

| T.1.2.3 Simulate Future Heavy Rainfall Scenarios |
|---|
| Similar to task 1.2.2 Simulate Present Heavy Rainfall Scenarios. Instead of historic precipitation data calculated precipitation data of future climate scenarios will be used. This approach should enable the storm water manager to simulate how the water will run off the surface in different climate scenarios with increased occurrence or intensity of storm water events. The so gained insights will hopefully help to find suitable solutions for the flooding problems in a long term view. |
| **T.1.3 Validate Simulation Results** |
| This task validates the results from Task 1.2.2 and Task 1.2.3 regarding the certainty of the model results. Due to the nature of models it is not possible to verify the simulation results completely or to assume a perfect representation of the real world. Instead the confidence of storm water managers in the model results shall be increased with single spot checks. Single flooding predictions of the model will be compared to present and past experiences. Examples are:<br>• Experience of storm water manager<br>• Input from public institutions (fire fighter, etc.)<br>• Enquiry by property owner<br>With the help of these facts the quality of the model predictions can be validated to a certain level. |

## 4.2. Relevant Use Cases (V2)

The tasks described above relate to a number of use cases that are the basis for the implementation effort in the second year. For more details see D6.1.2 Wuppertal Pilot Definition Plan V2).

- **UC-619** Browse 3D Map

  *A digital elevation model of the Wuppertal region should be rendered and presented to the user. The user should have the possibility to browse the landscape and to add additional information like overlay maps of cadastral context information and 3D objects like building geometries. This use case creates the base for the later surface run-off modelling. The modelling results (flow direction of water, maximum water heights) will be displayed in the 3D environment.*

- **UC-6110** Show Historic Precipitation

  *The user would like to display measured precipitation values for the region of Wuppertal in form of precipitation maps (static) and animations of the event over time (dynamic). Also the functionality to display the precipitation data in alternative representations for example as time trend diagrams should be provided. The shown visualisation should allow querying the values at each point in time. This functionality enables the user to click in the map and to receive the exact precipitation amount or temperature value.*

- **UC-6111** Show Simulated Precipitation

  *This use case provides similar functionality as the use case UC-6110 but instead of visualising historic precipitation data the user wants to generate and visualise future rain patterns. The simulated rain patterns will be generated based on an assumed climate scenario and historic precipitation data. The result is predicted precipitation in the future.*

- **UC-6112** Generate Rain Fall Pattern

  *The storm generator gives the user the possibility to generate a rain event based on measured maximum precipitation levels and the desired direction of the storm. This use case enables the user to create rain patterns to use with the surface run-off model to simulate different rain events and to study the water run-off. The generated storm patterns can be saved by the user for later usage. This functionality also enables the user to create a grid of "virtual" pluviographs because the storm generator can generate time series for specific geospatial points. The resulting virtual measurements could be used for further surface run-off simulation.*

- **UC-6113** Compare Precipitation data

  *This use case gives the user the possibility to compare different rain events (historic [UC-6110], predicted [UC-6111], and generated [UC-6112]). This feature will be used to increase the confidence in predicted and generated rain events by comparing them*

*with real world measurements. This use case also enables the user to study possibilities how the present/historic rain events will change in future climate scenarios.*

- **UC-6114** Model Surface Run-Off

  *The use case provides the possibility to configure and perform surface run-off modelling with precipitation data (historic, predicted, and generated) for a particular catchment area. The user has the possibility to choose the different input parameters for the model such as the digital surface model, precipitation data, etc. The execution of the model should be performed in the background so that the user still can work with the system. Upon completion of the surface run-off modelling the user will be notified and will be able to visualise the model results in the map.*

- **UC-621** Modify Digital Surface Model

  *This use case describes the creation of an alternative Digital Surface Model starting with an initial Digital Surface Model that was set up by means of the local model GeoCPM outside of the Wuppertal pilot application. The modification of the initial Digital Surface Model is restricted to the alteration of a set of existing breaklines that have to be defined in the initial Digital Surface Model (no recalculation of the TIN that defines the Digital Surface Model).The modified model represents certain structural measures for the physical protection of buildings or other facilities. It is possible input to UC-6114.*

- **UC-622** Annotate Local Model Results

  *This use case describes the annotation of Local Model results that are stored in the SMS repository. These results have been generated via UC-6114. Supported by a visual interpretation of the Local Model results (cf. UC-619) the user will assess the effectiveness and efficiency of the simulated structural measures for the physical protection of buildings or other facilities. (Note: this is done by means of expert knowledge outside the scope of the Wuppertal pilot application!) In the end of this use case the user will annotate the Local model results with his findings to keep hold of them.*

While in the first development iteration focussed on the identification of critical areas the Pilot V2 mainly focuses on the model execution. The design of the graphical user interface was mainly done in meetings, telephone conferences, and one-on-one interviews. In these sessions the design was discussed on a live and running development snapshot via screen sharing. However, mockups have been produced to further clarify the user interface design, especially for those who have not been able to join the live sessions. The current development status of the relevant uses cases is reflected in Table 1: Implementation status.

| Use Case/Status | Mockup | Working Prototype | Stable Solution |
|---|---|---|---|
| **UC-619**<br>**Browse 3D Map** | ✓ | ✓ | |

| | | | |
|---|---|---|---|
| **UC-6110**<br>**Show Historic Precipitation** | ✓ | ✓ | ✓ |
| **UC-6111**<br>**Show Simulated Precipitation** | ✓ | ✓ | |
| **UC-6112**<br>**Generate Rain Fall Pattern** | ✓ | | |
| **UC-6113**<br>**Compare Precipitation data** | ✓ | ✓ | |
| **UC-6114**<br>**Model Surface Run-Off** | ✓ | ✓ | |
| **UC-621**<br>**Modify Digital Surface Model** | ✓ | ✓ | |
| **UC-622**<br>**Annotate Local Model Results** | | | |

**Table 1:** Implementation status

## 4.3. Graphical User Interface

In the following we explain the status of the individual use case implementations.

- **MD-WP6-4** *Show Precipitation data*, referring to UC-6110 and UC6111: This mockup illustrates how precipitation data, that has been made available to the SMS, can be found and viewed within a geo-spatial context. Alternatively it can also be located via a predefined catalogue structure, well suited to the needs of the users.

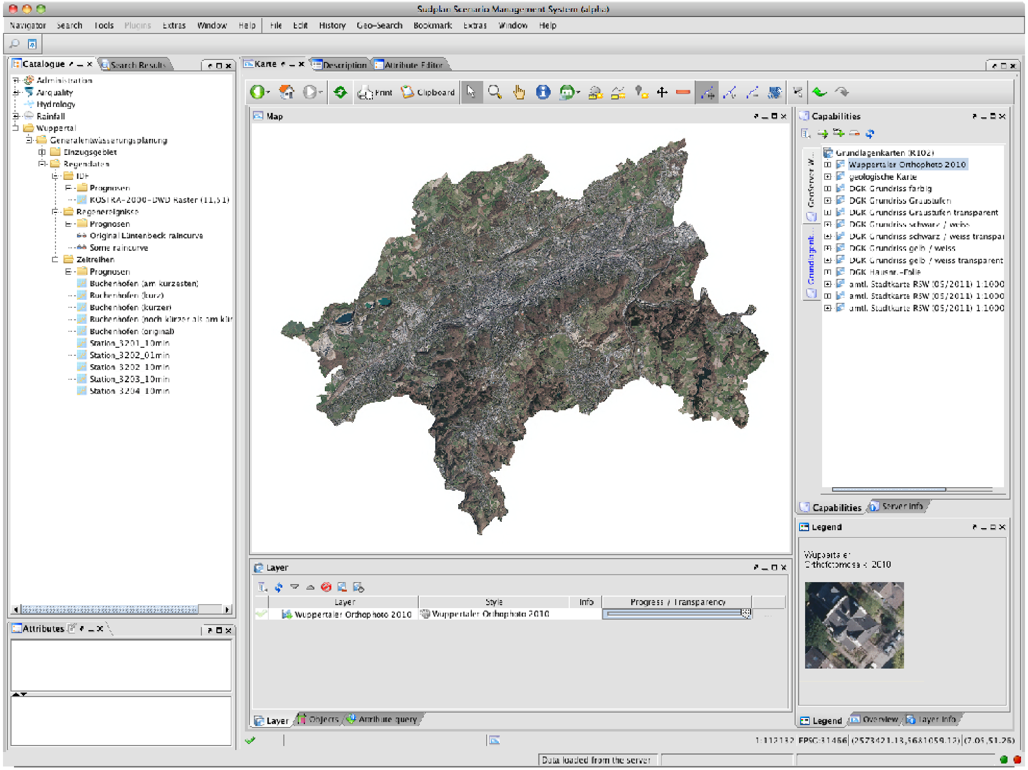


**Figure 3:** Load desired geo-spatial context

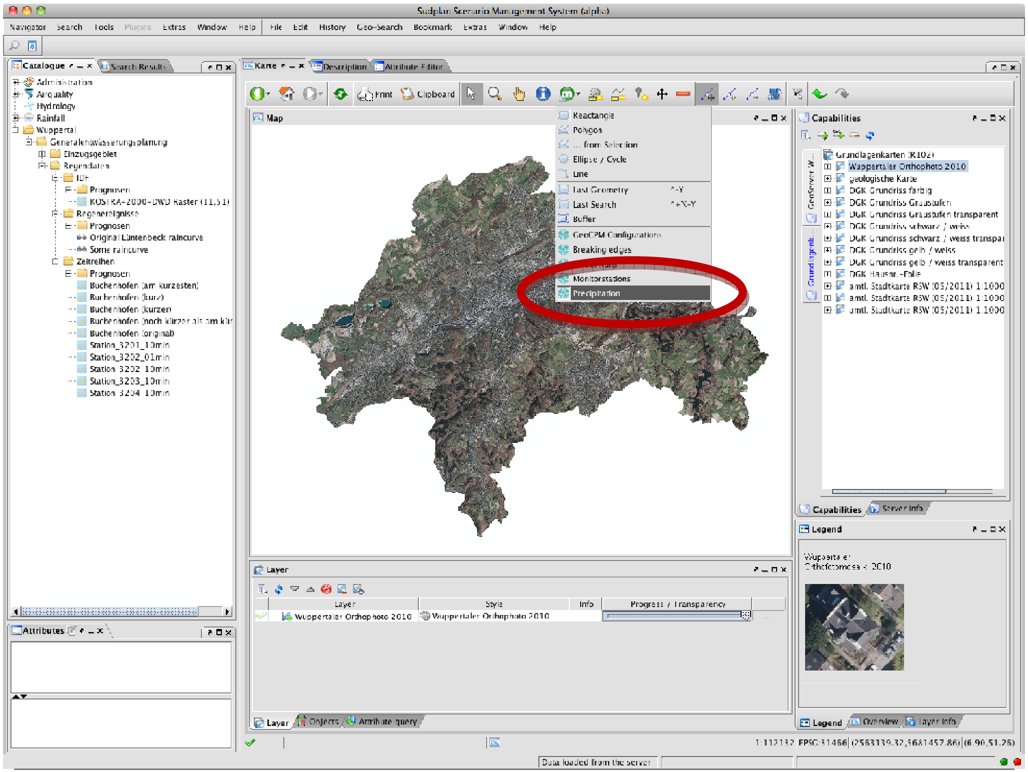


**Figure 4:** Select "Precipitation" as search context

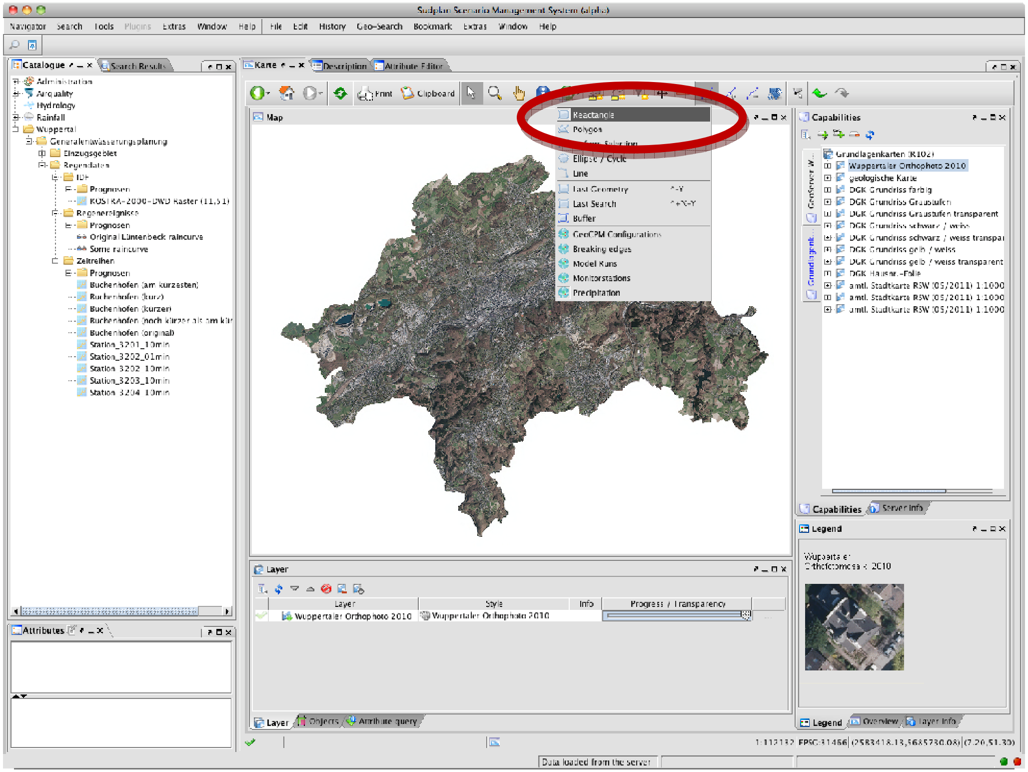

**Figure 5:** Select "Rectangle" as search geometry

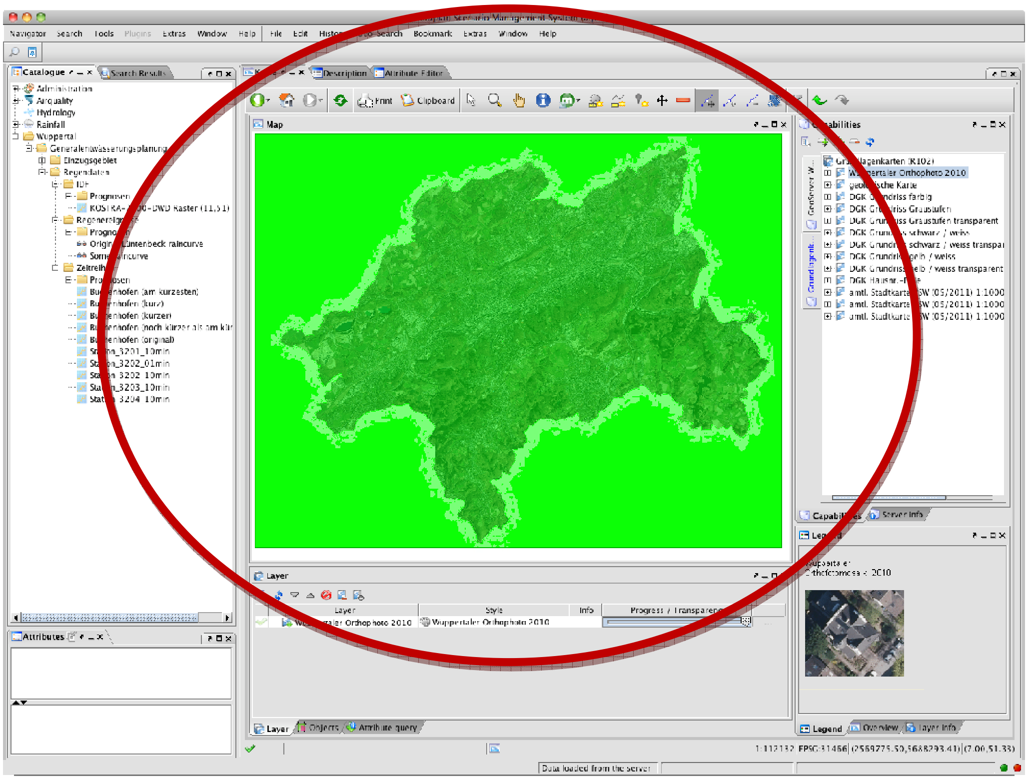

**Figure 6:** Select search area

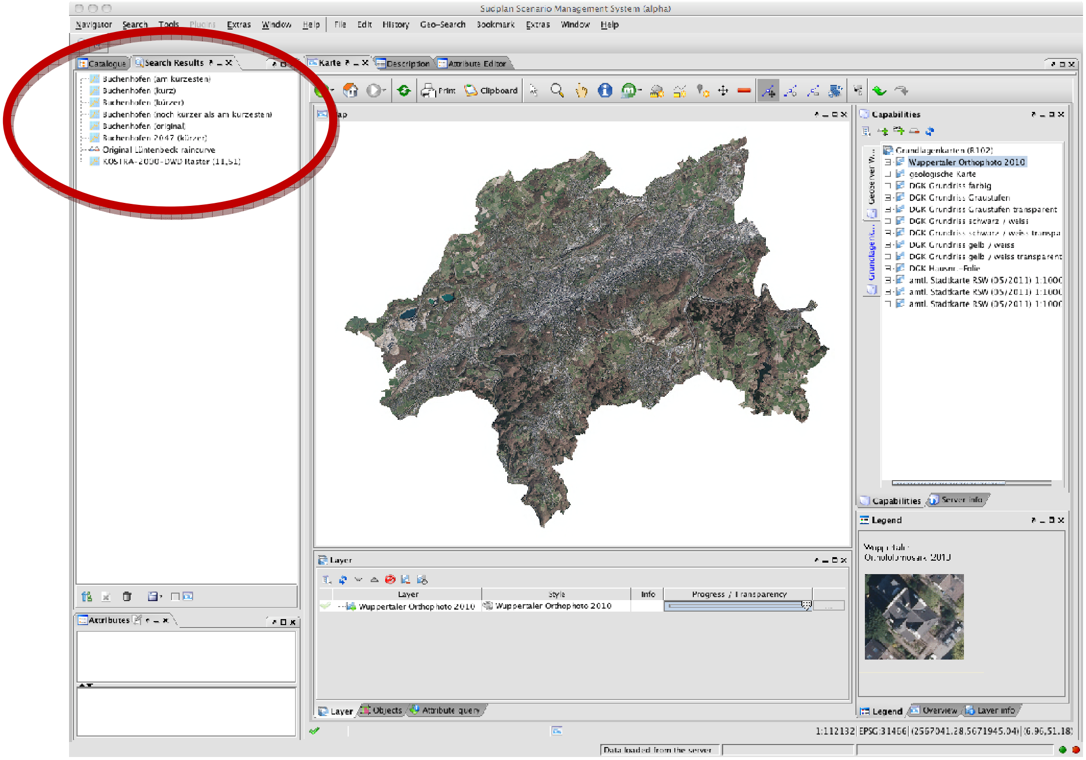


**Figure 7:** Search results are visualised in the "Search Results" tree

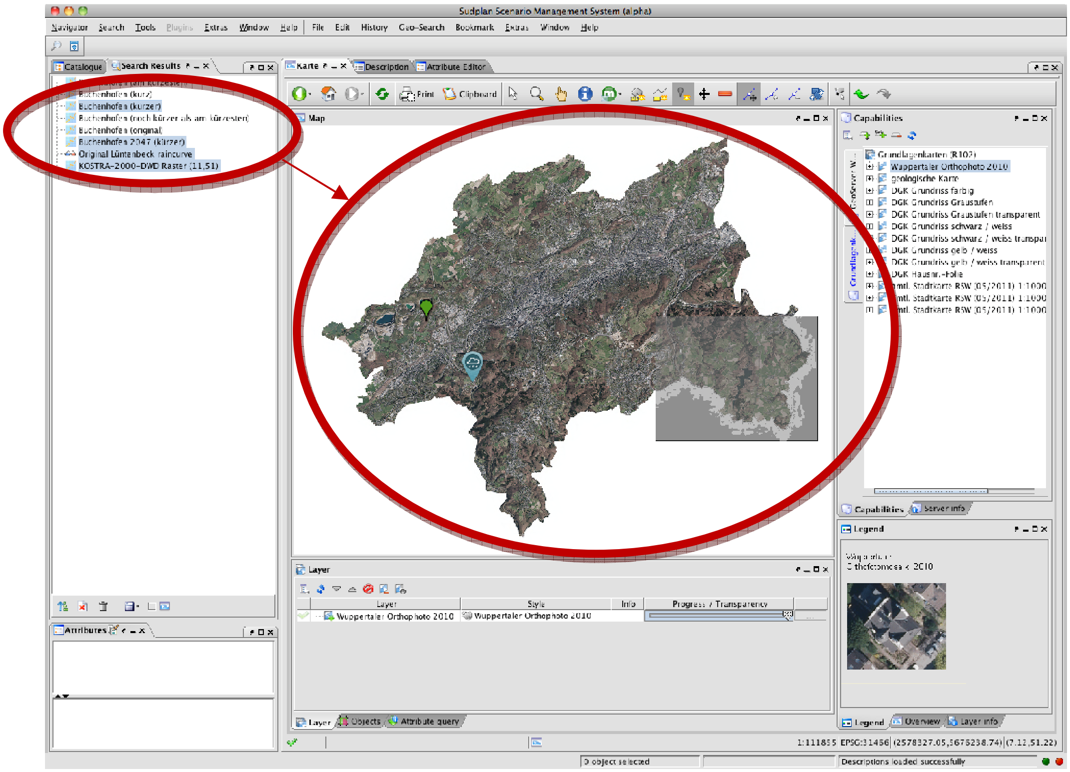


**Figure 8:** Drag & Drop desired results to the map

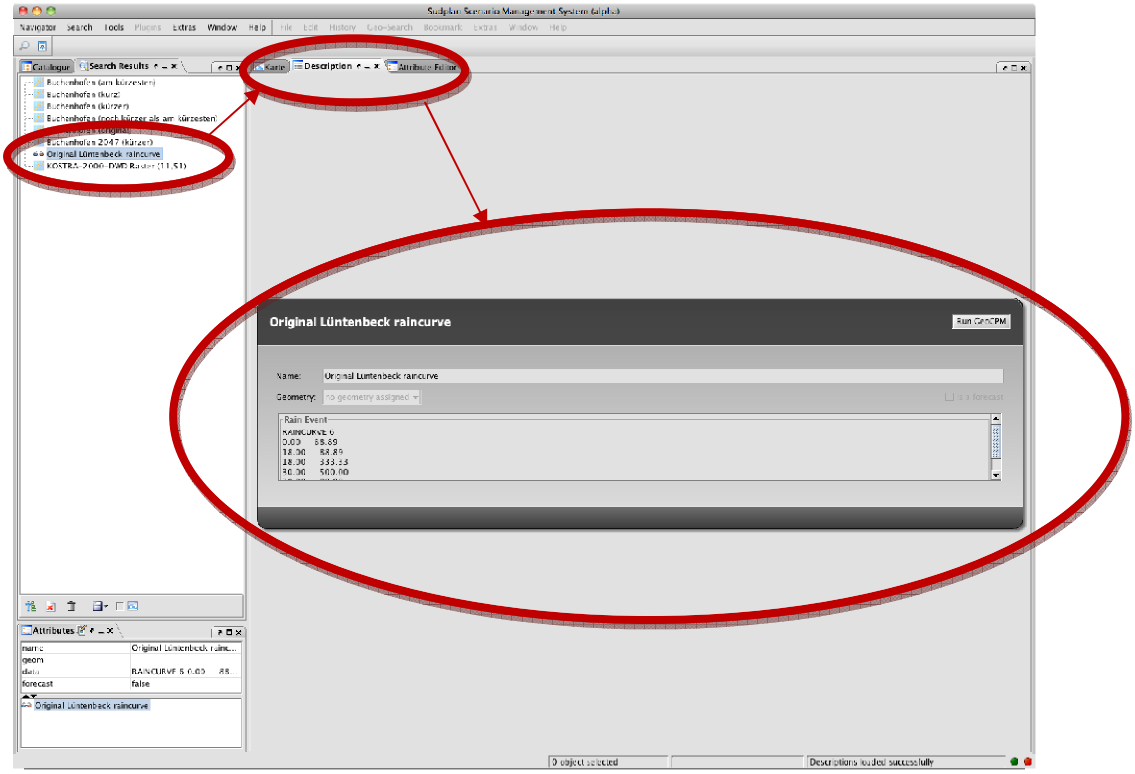

**Figure 9:** Select a rain event and switch to the "Description" pane to see its values

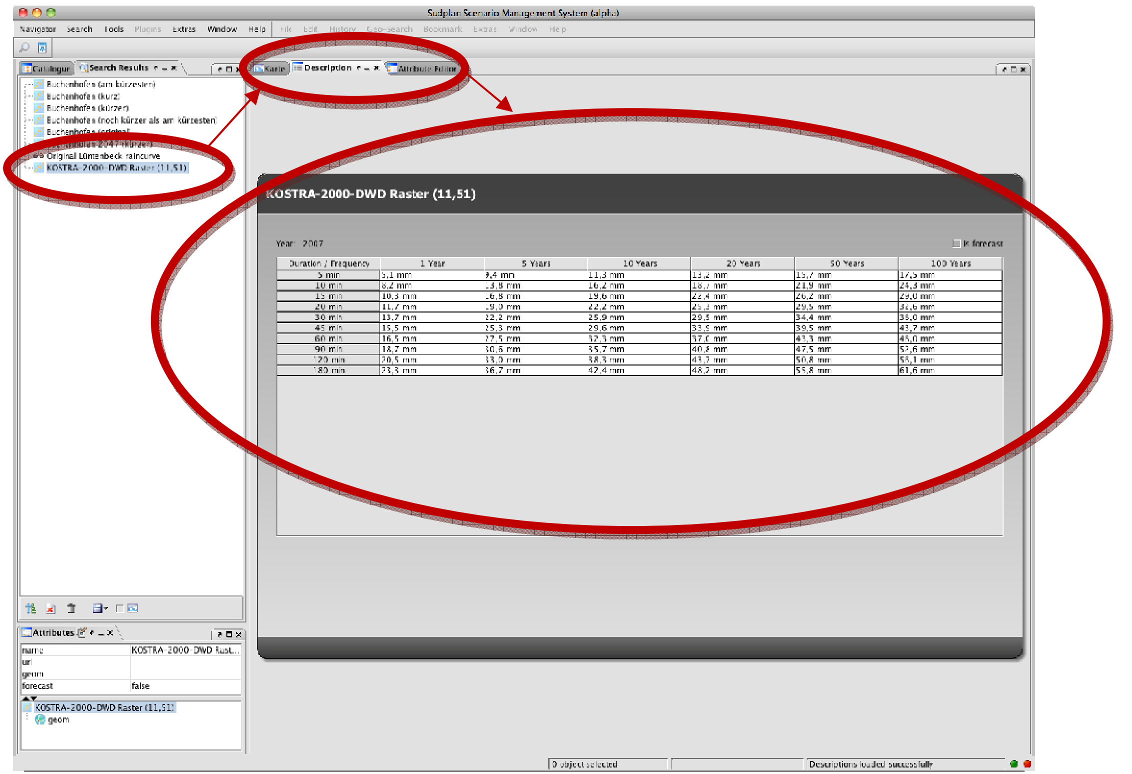

**Figure 10:** Select an IDF curve and switch to the "Description" pane to see its values

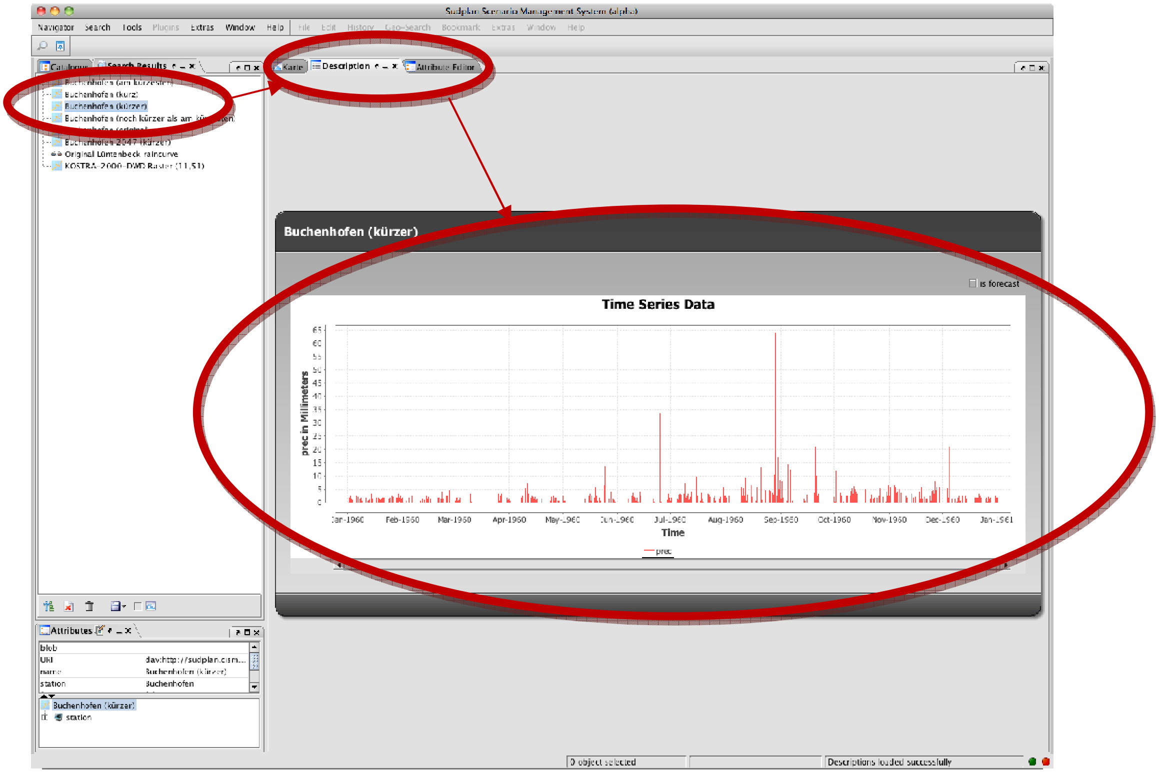


**Figure 11:** Select a time series and switch to the "Description" pane to see its values

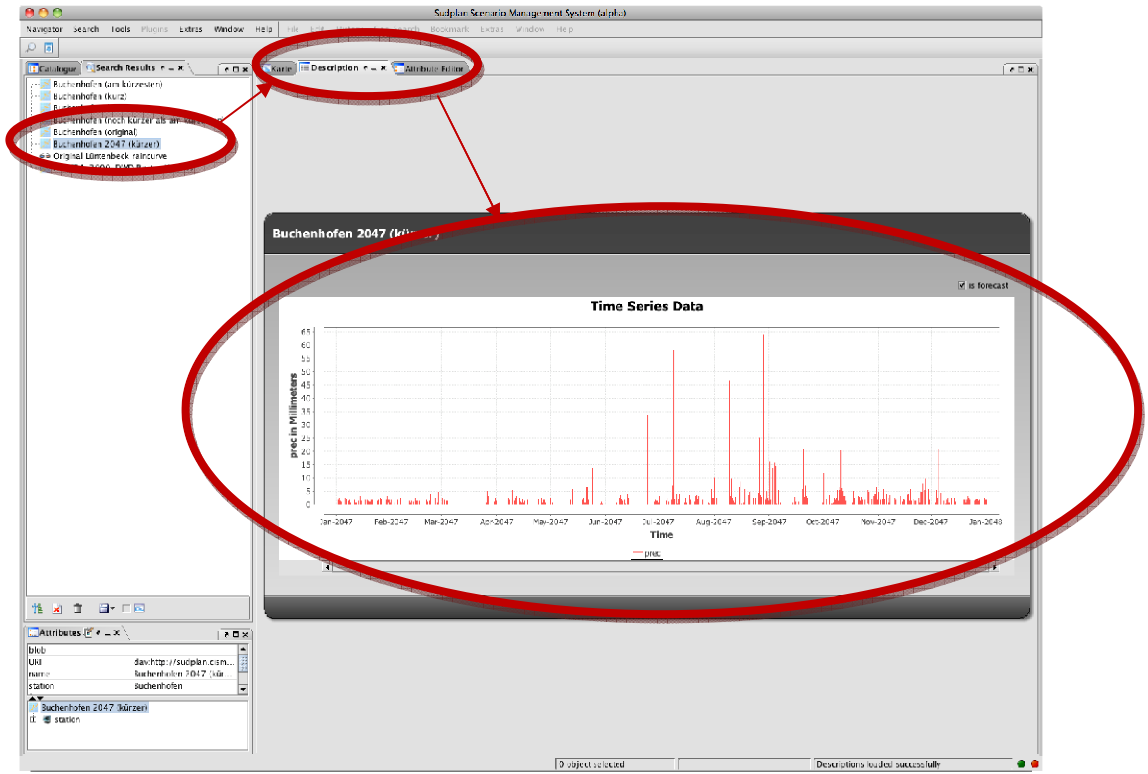


**Figure 12:** Select a forecast (downscaled) time series and switch to the "Description" pane to see its values

- **MD-WP6-5** *Compare time series of different time periods*, referring to UC-6113: This mockups shows basic comparison features for time series that do not lie within the same time period.

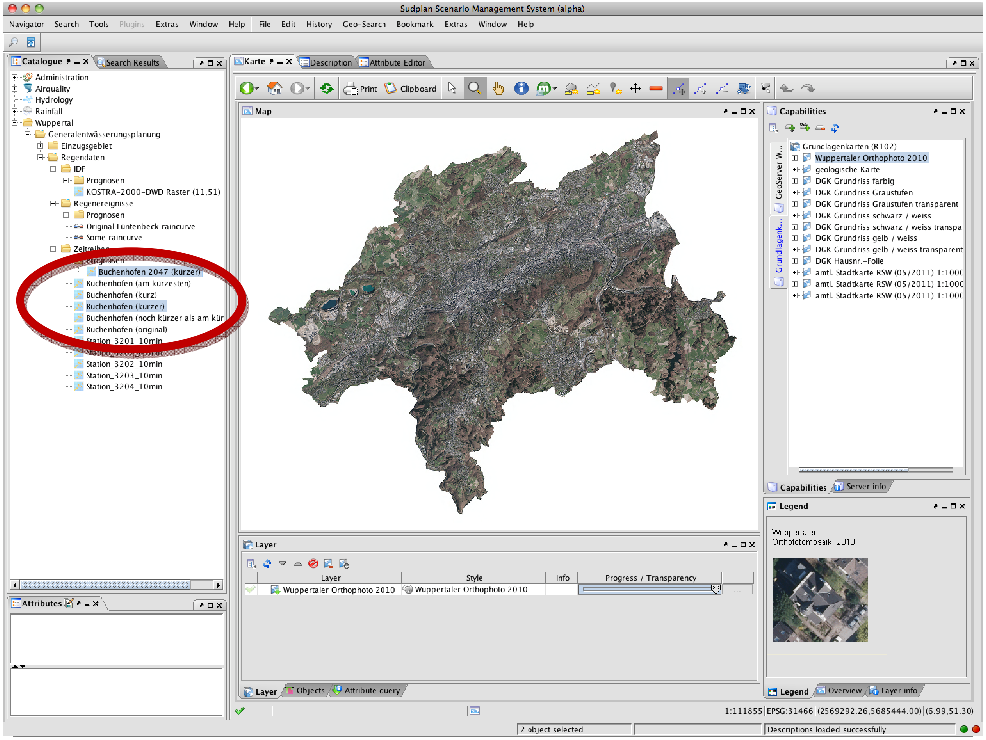


**Figure 13:** Select time series to compare

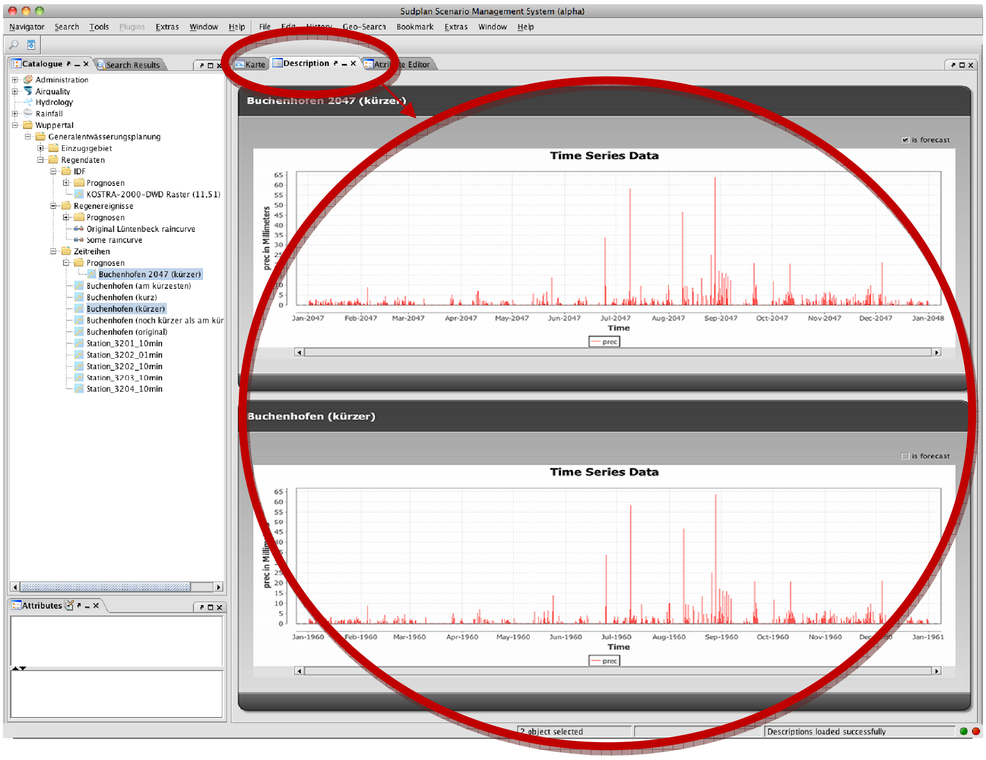


**Figure 14:** Select "Description" pane to view the selected objects

- **MD-WP6-6** *Generate rain event from IDF curve*, referring to UC-6112: This mockup shows the generation of new rain events from existing IDF curve data:

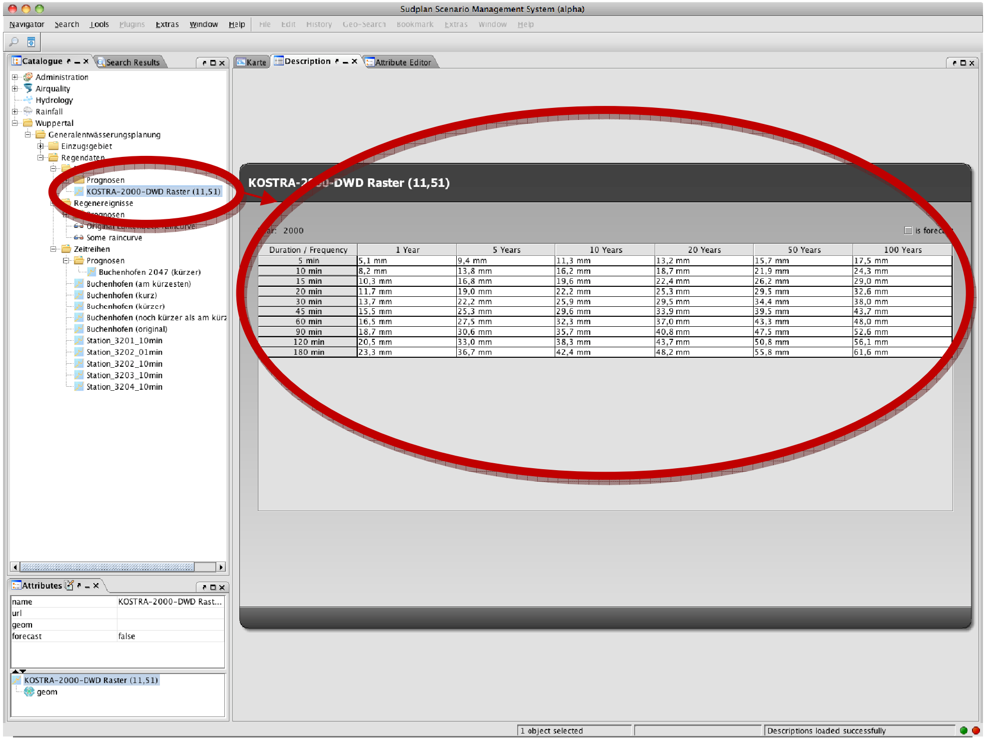

**Figure 15:** Select IDF curve and view it in the "Description" pane

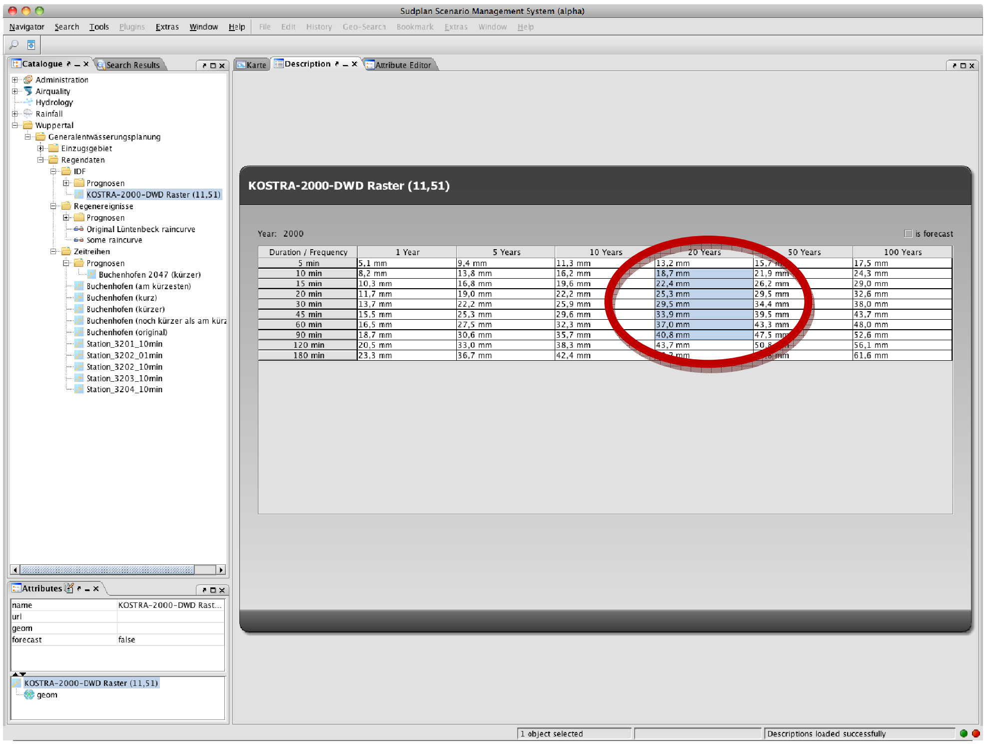

**Figure 16:** Select values from the IDF curve which shall be used for generation

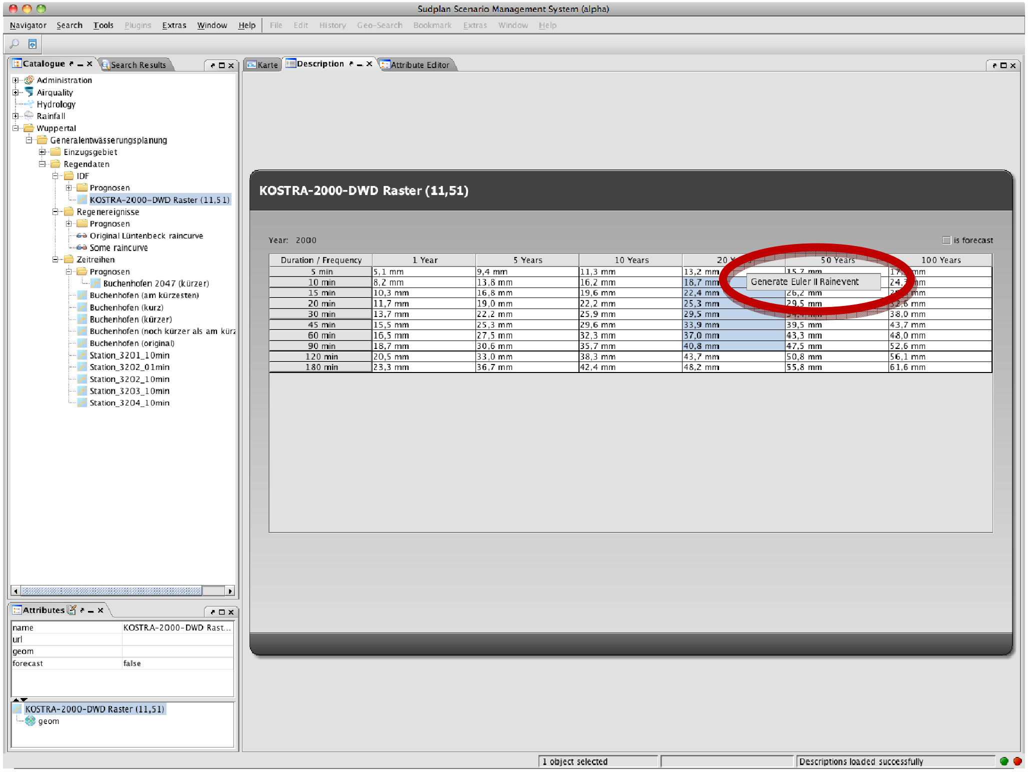


**Figure 17:** Select "Generate Euler II Rainevent" from the contextual menu

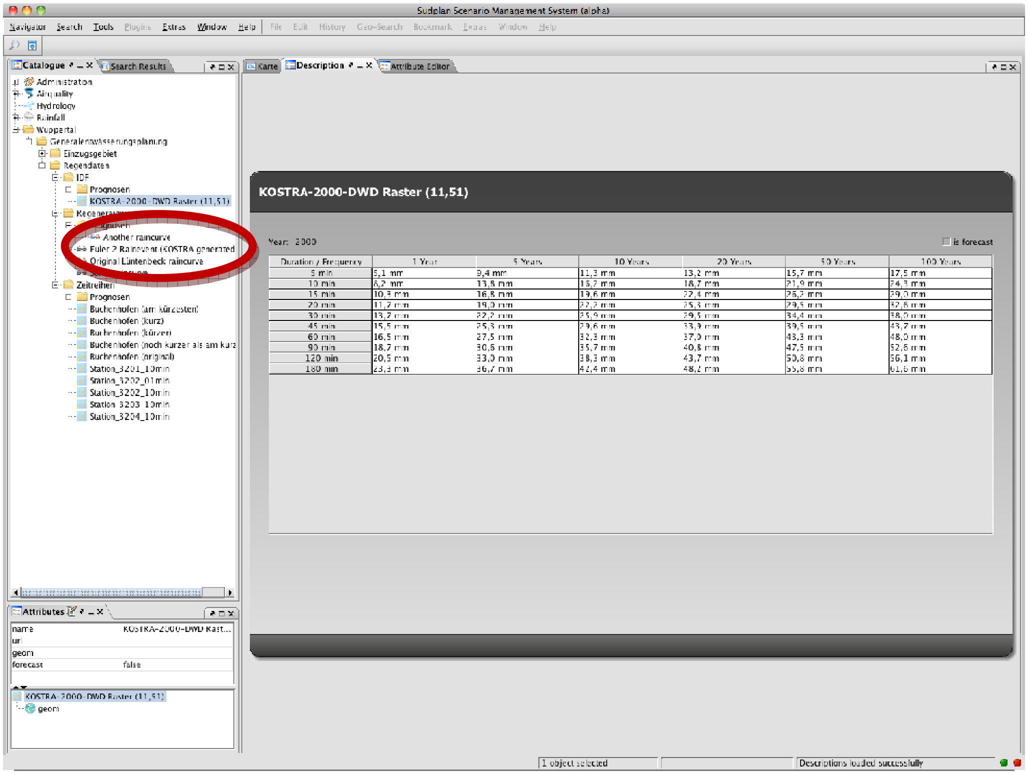


**Figure 18:** A new rain event appears in the catalogue

- **MD-WP6-7** *Modify and run Runoff simulation*, referring to UC-6114 and UC-621: This mockup illustrates how users can select GeoCPM configurations, view and manipulate their data. Moreover the users will then use this configuration to perform a new Runoff simulation. After the execution has finished the user is presented with the run results.

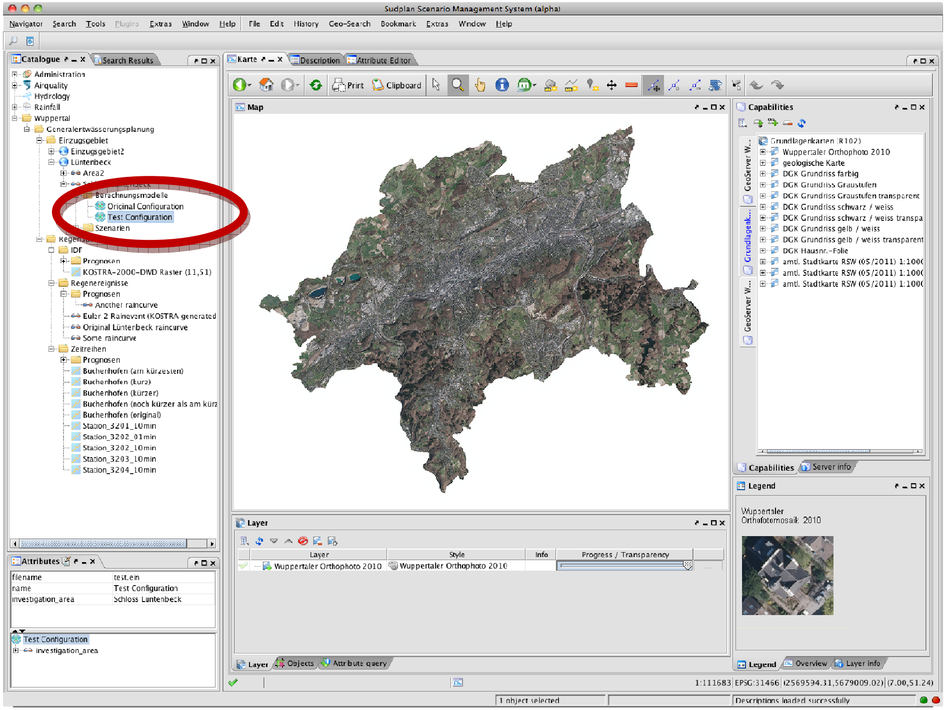

**Figure 19:** Select a configuration for a certain area of interest

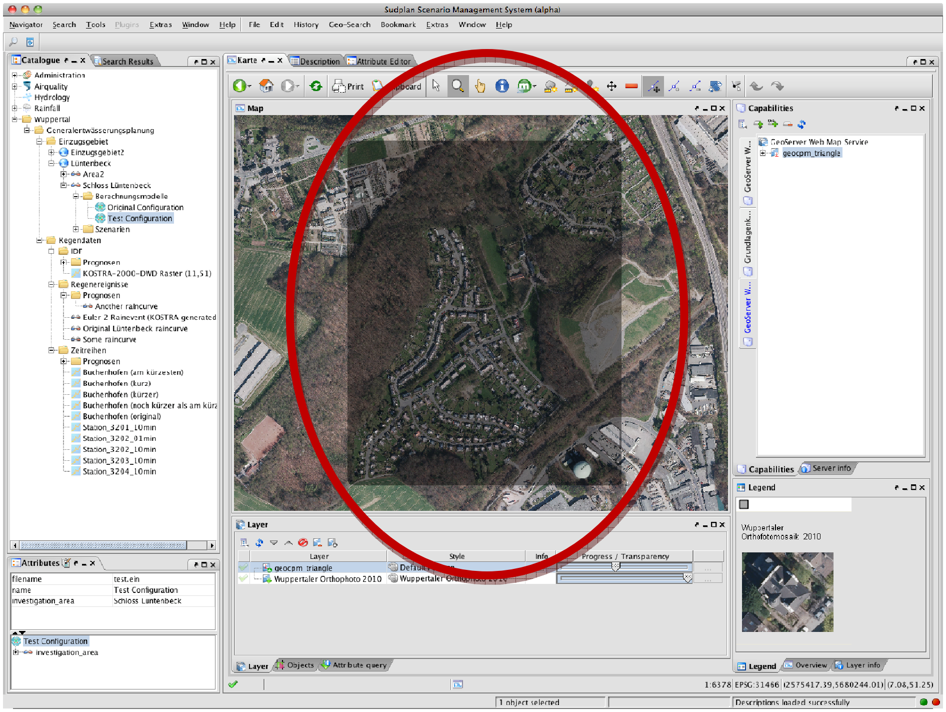

**Figure 20:** Get an overview of the area of interest with the configuration data

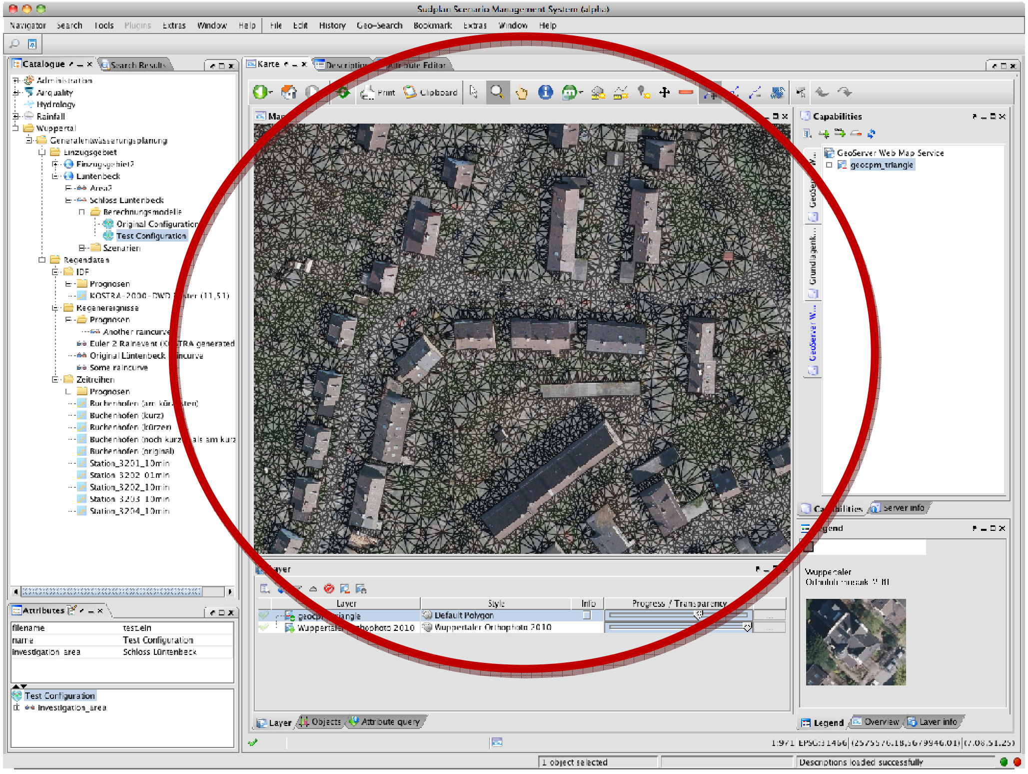

**Figure 21:** Zoom in to view the configuration in detail

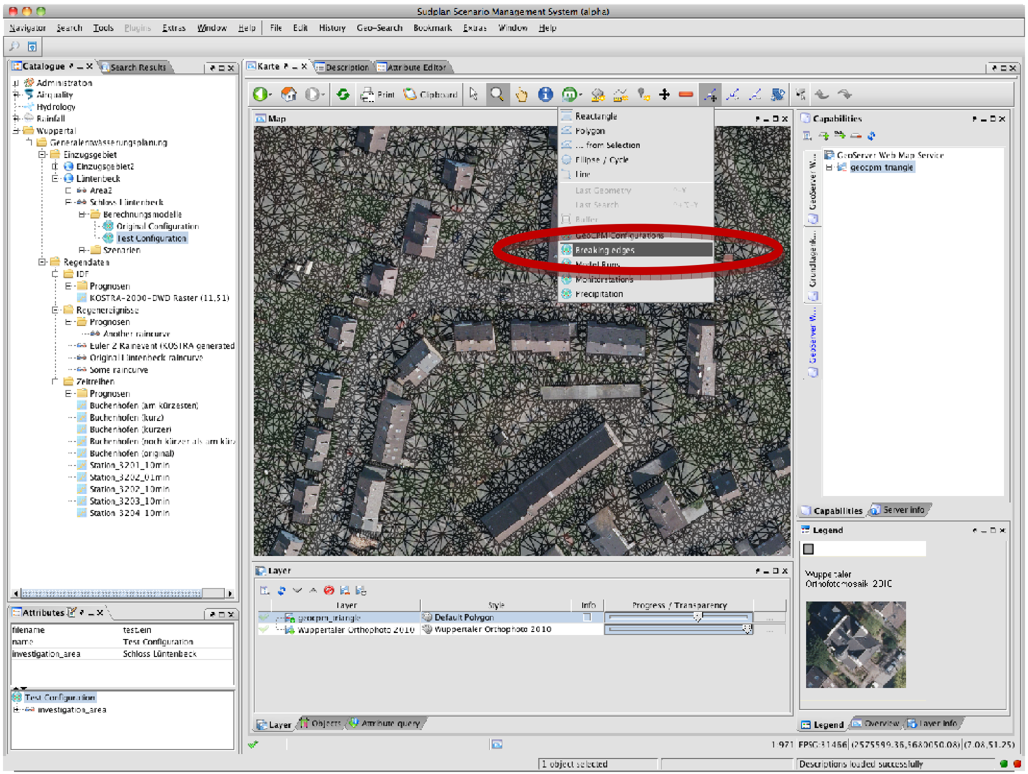

**Figure 22:** Select "Breaking edges" as a search context

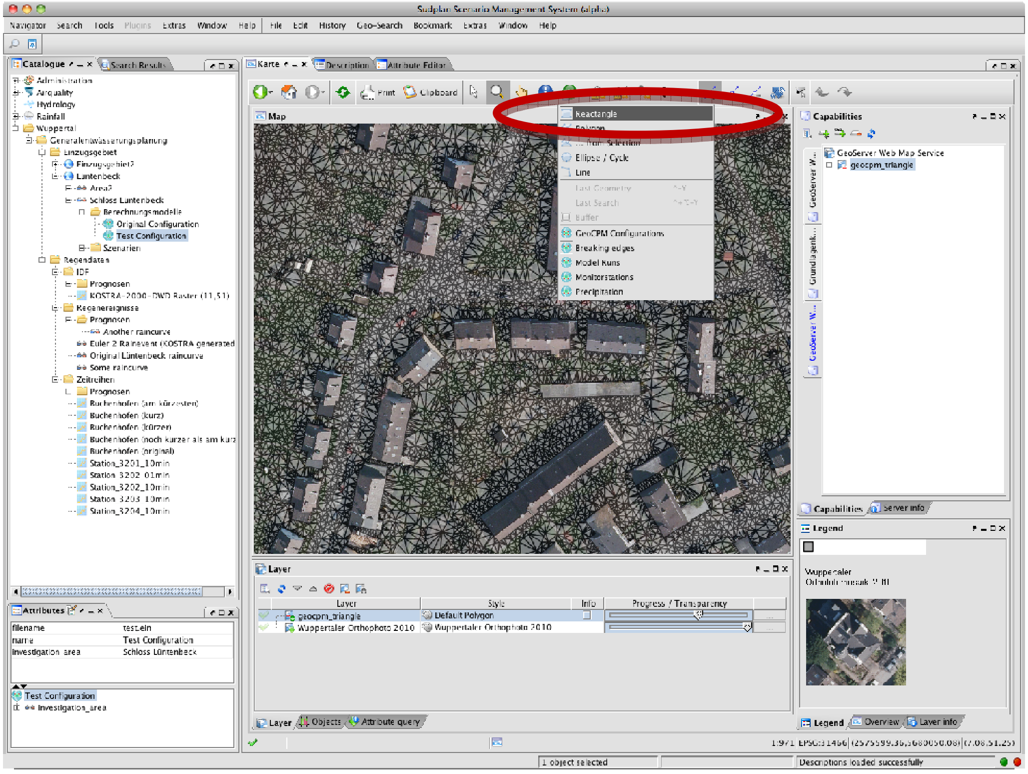

**Figure 23:** Select "Rectangle" as search geometry

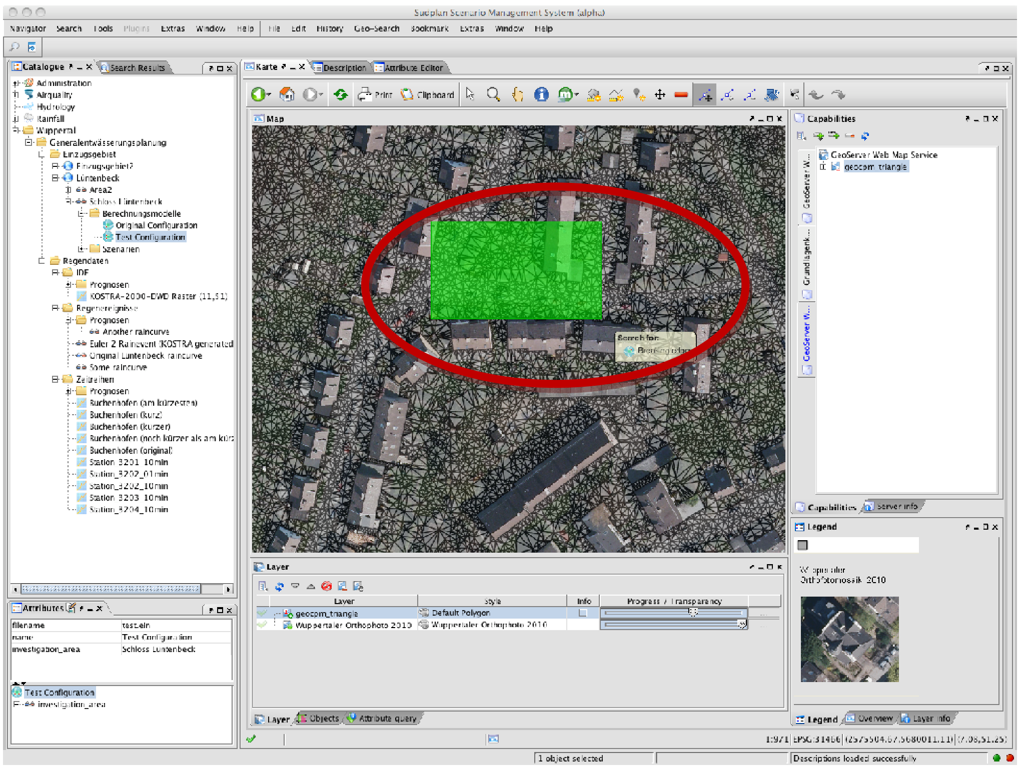

**Figure 24:** Select search area

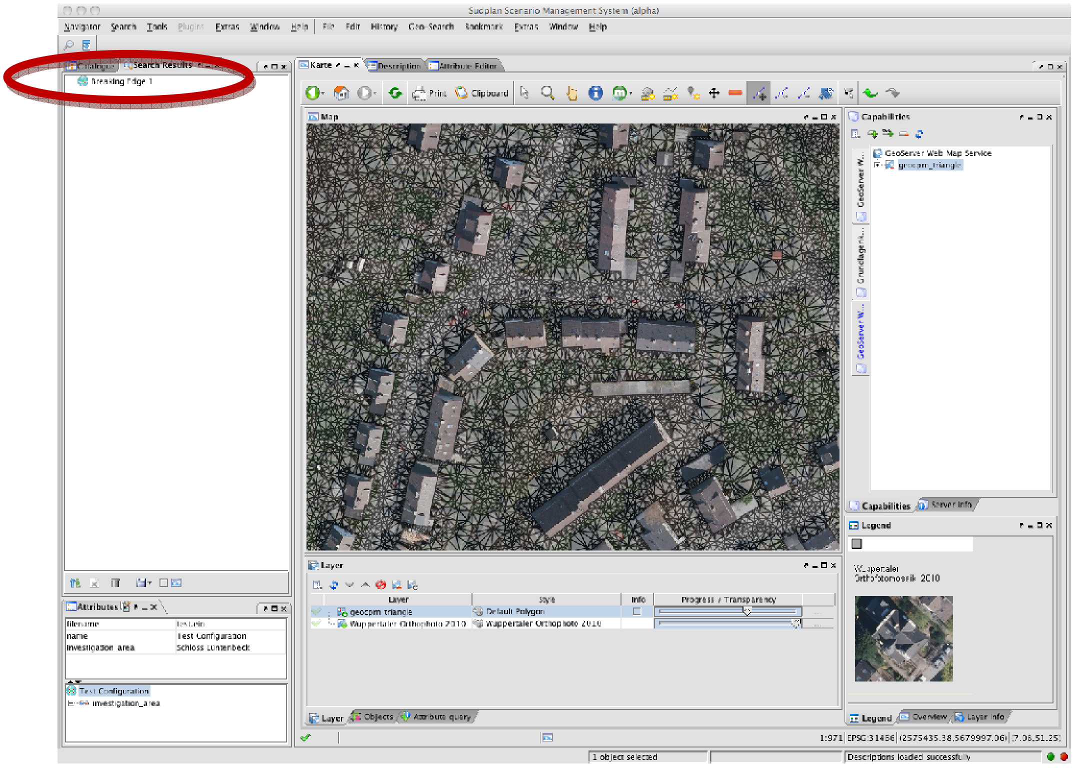


**Figure 25:** The breaking edges within the search area are shown in the "Search results" tree

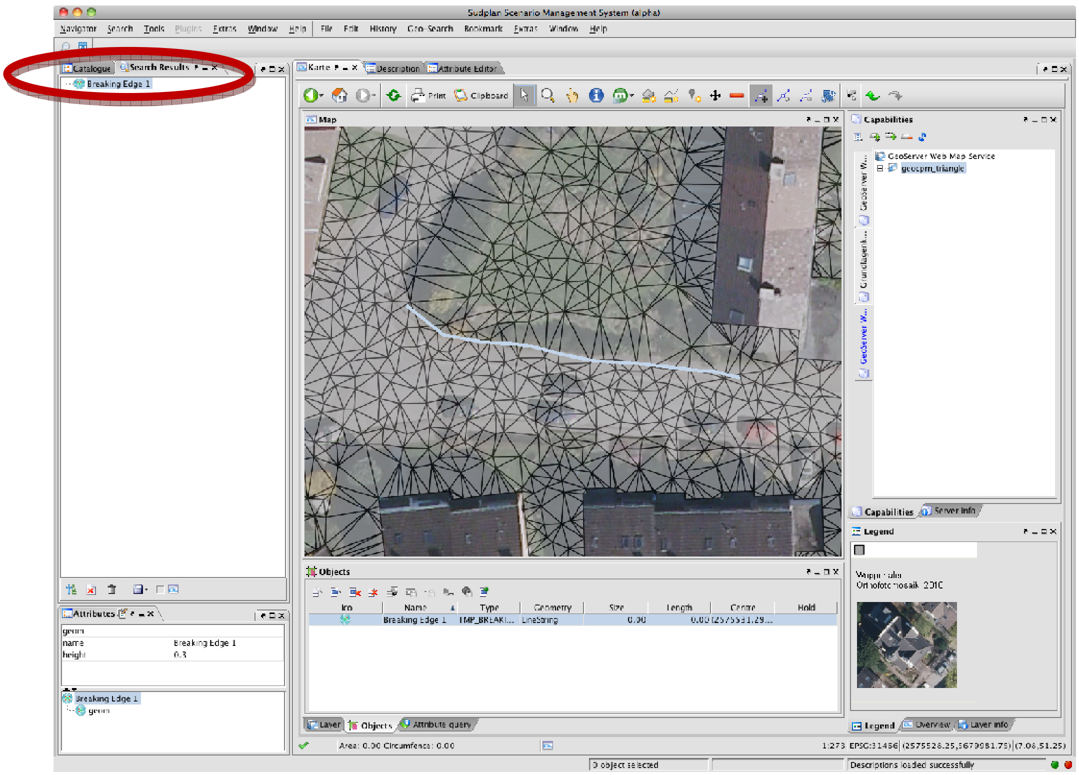


**Figure 26:** Drag & Drop the breaking edge to the map

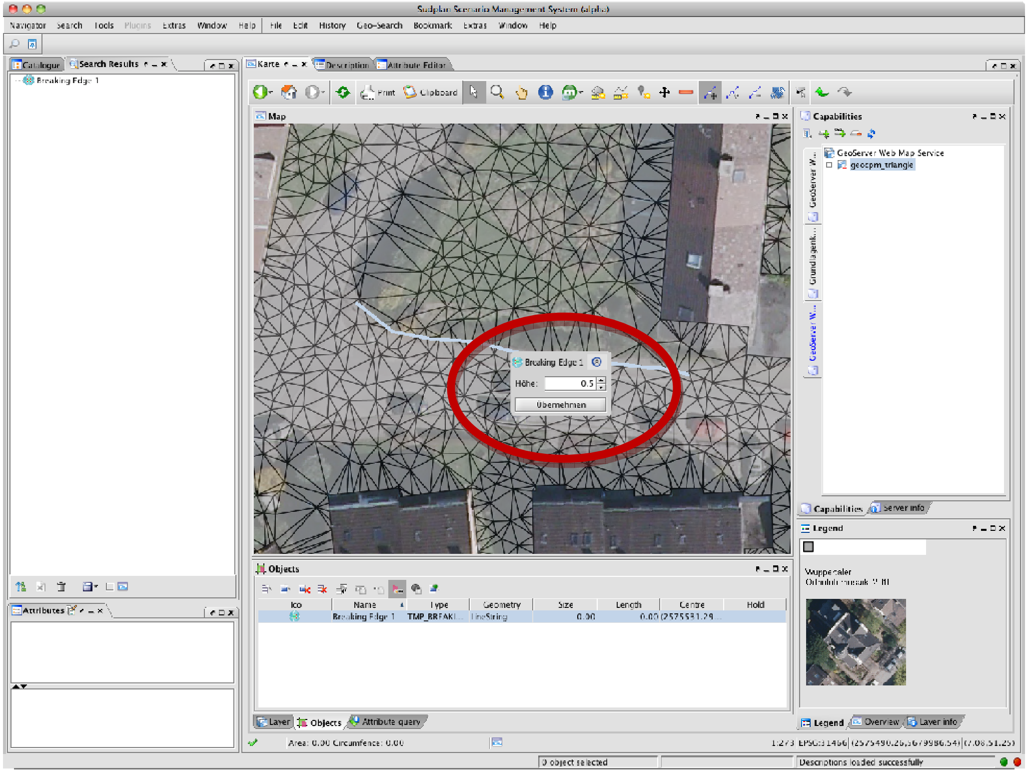

**Figure 27:** Manipulate the breaking edge height

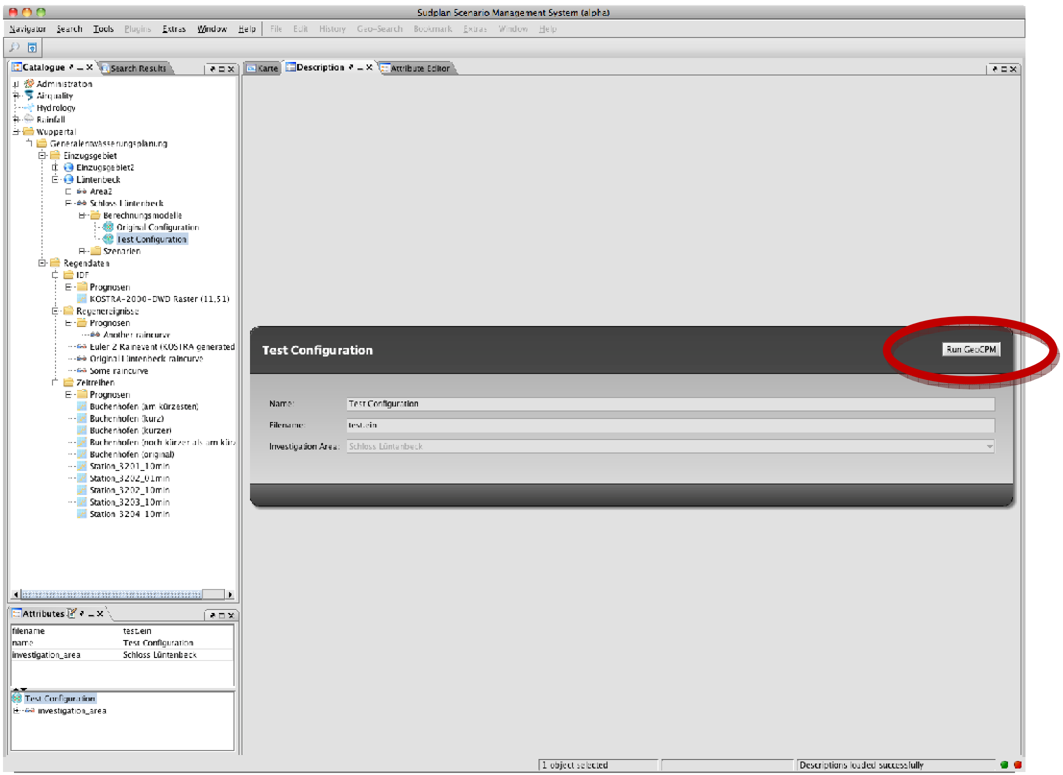

**Figure 28:** Choose "Run GeoCPM" to start a new simulation

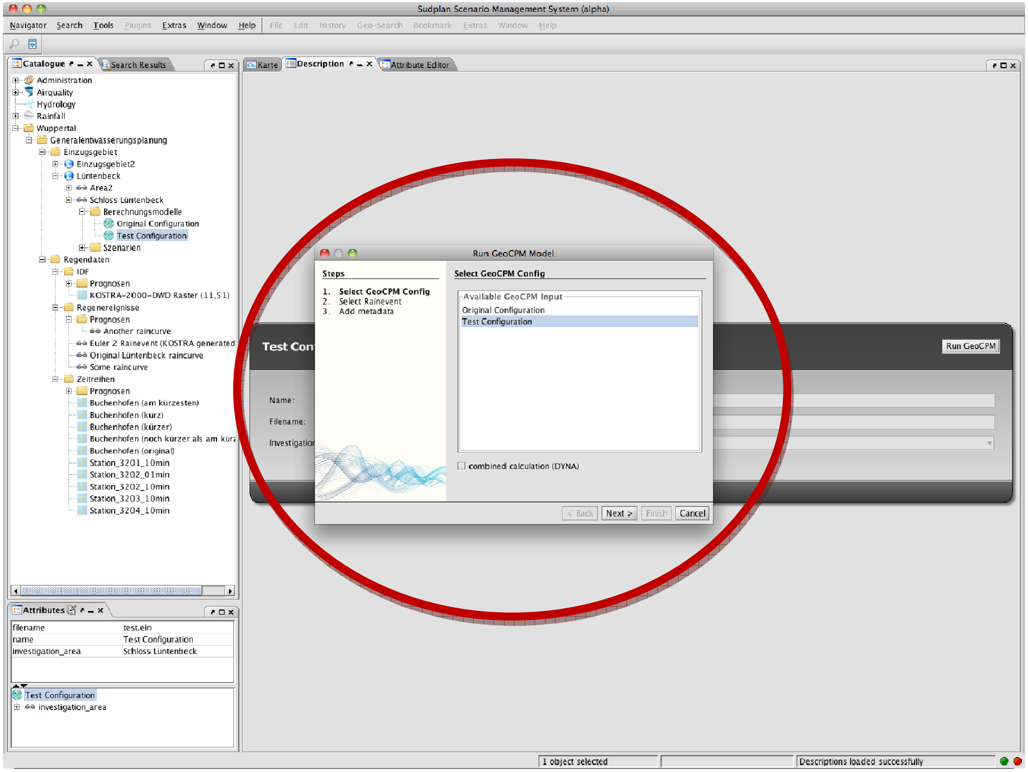


**Figure 29:** Choose the configuration to use

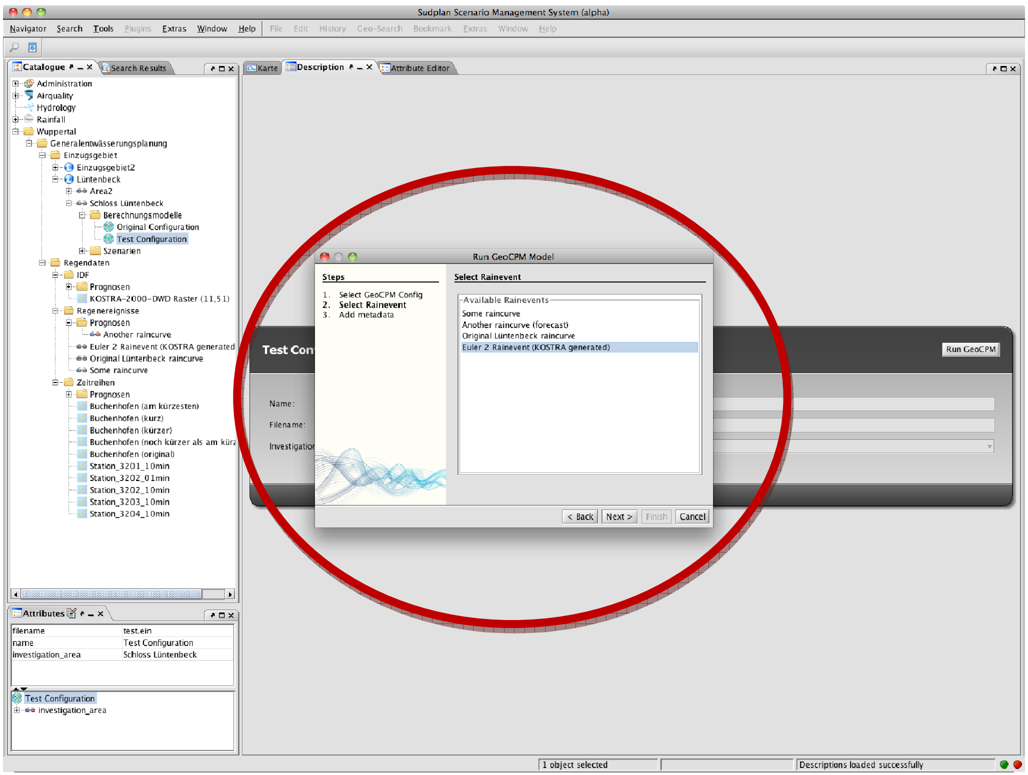


**Figure 30:** Choose the rain event to use

 

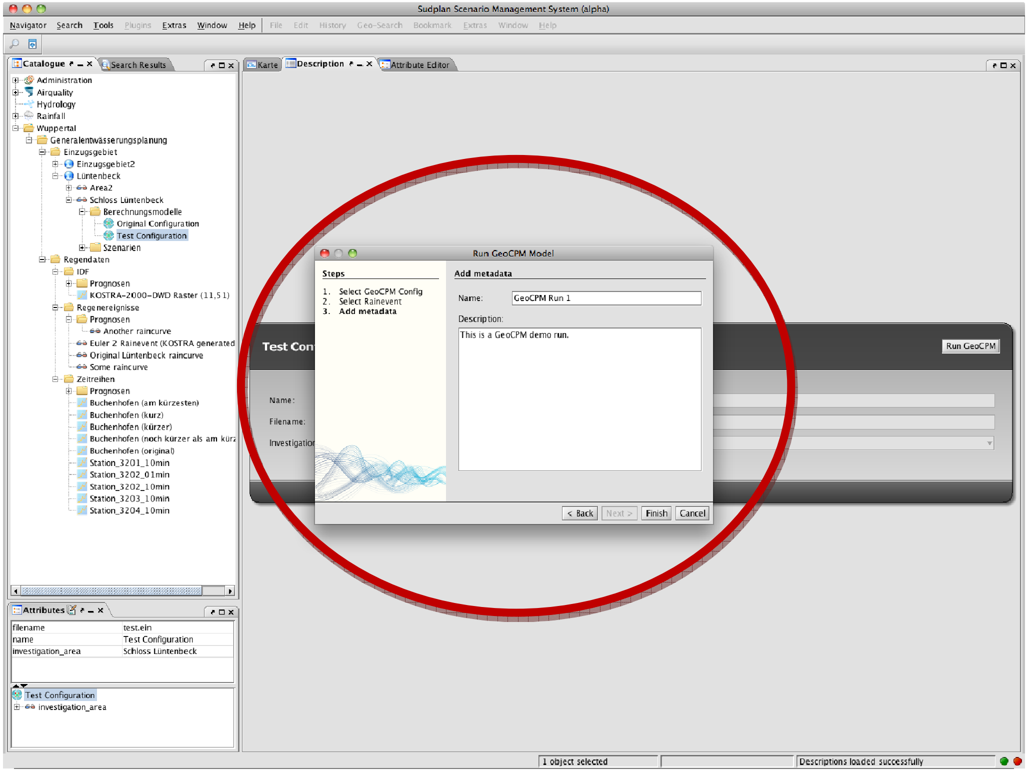


**Figure 31:** Add simulation metadata

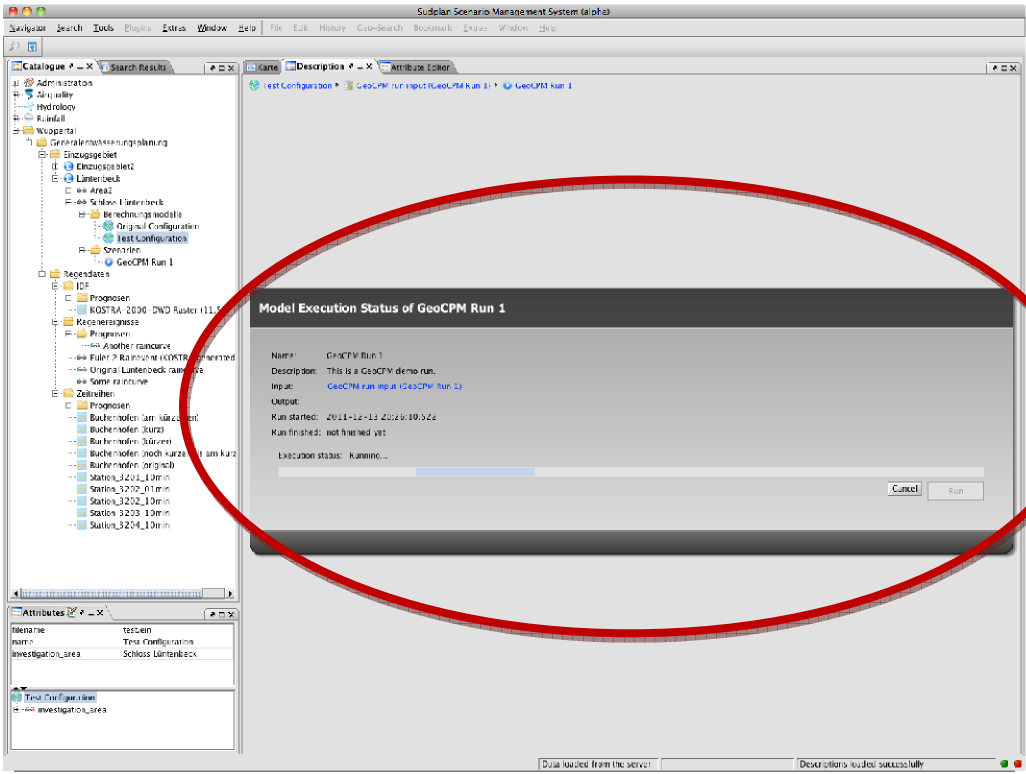


**Figure 32:** Wait for the simulation to finish

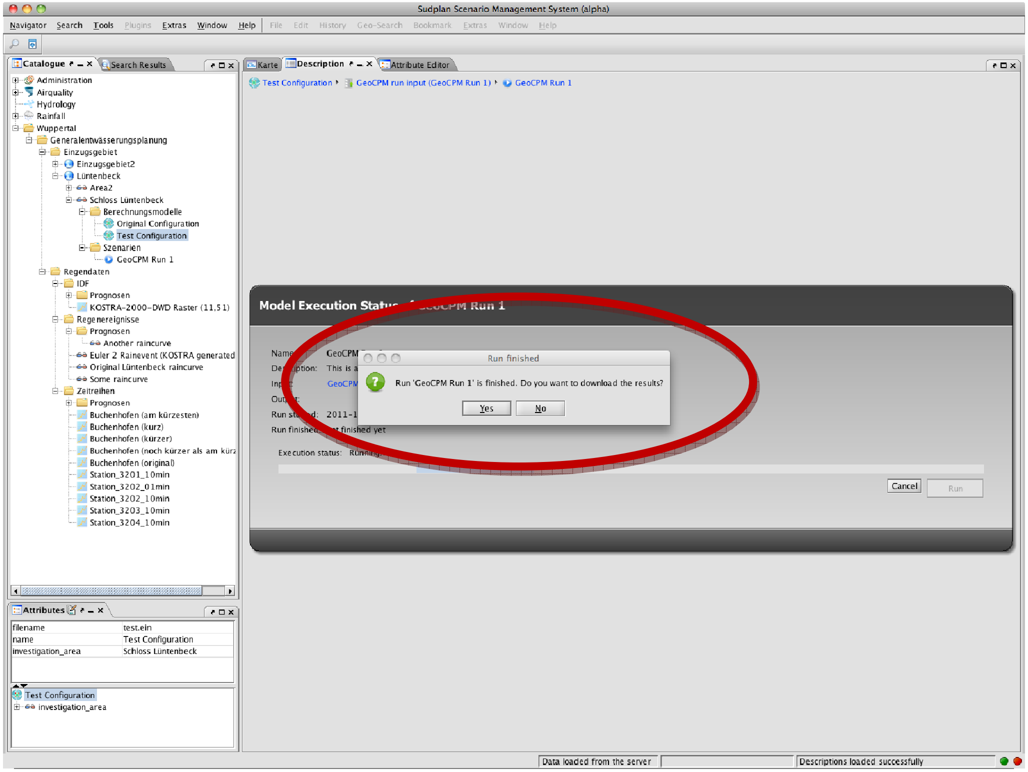


**Figure 33:** Choose whether to download the results of the simulation

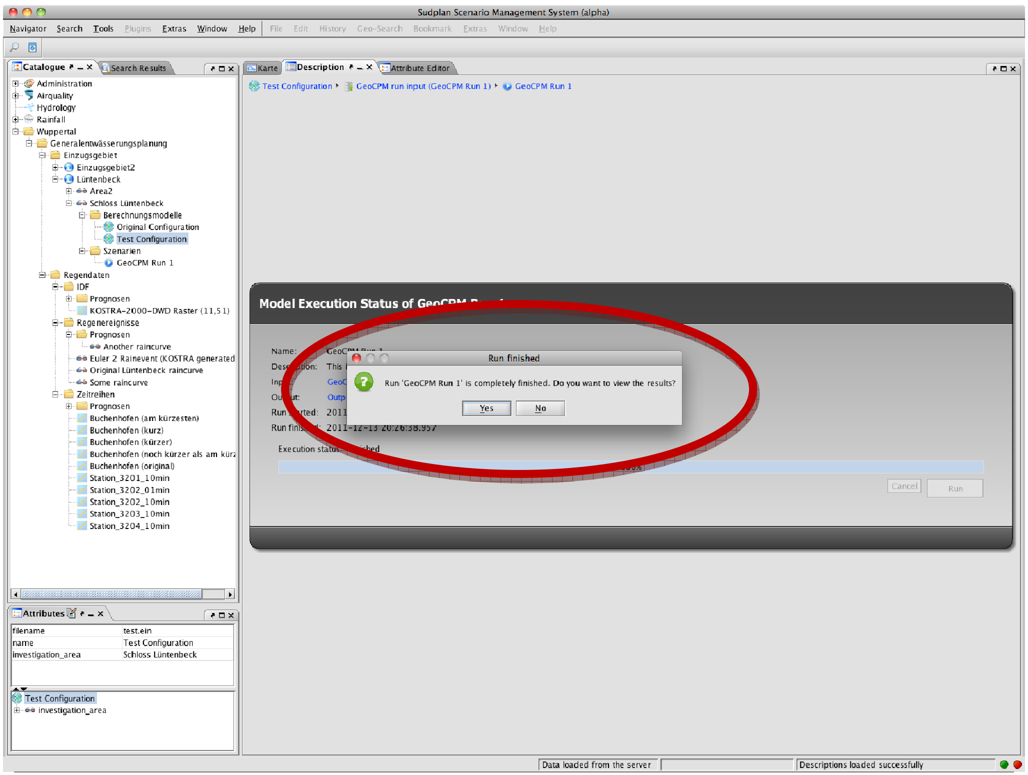


**Figure 34:** Choose whether to view the results of the simulation

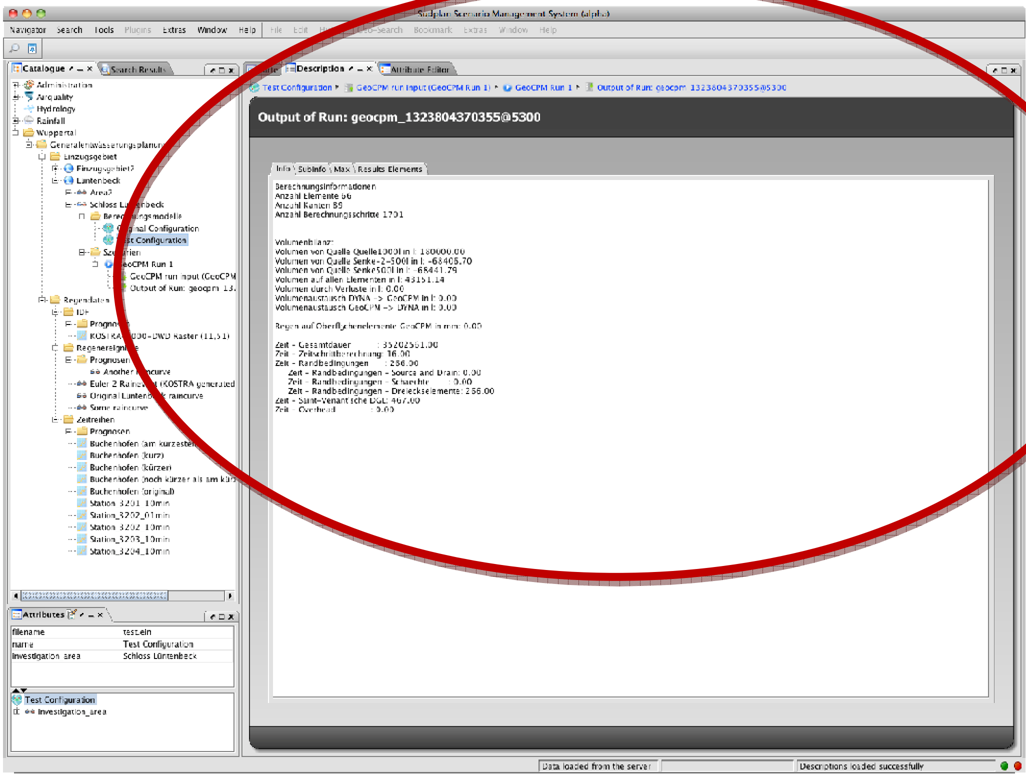


**Figure 35:** View the results in textual form

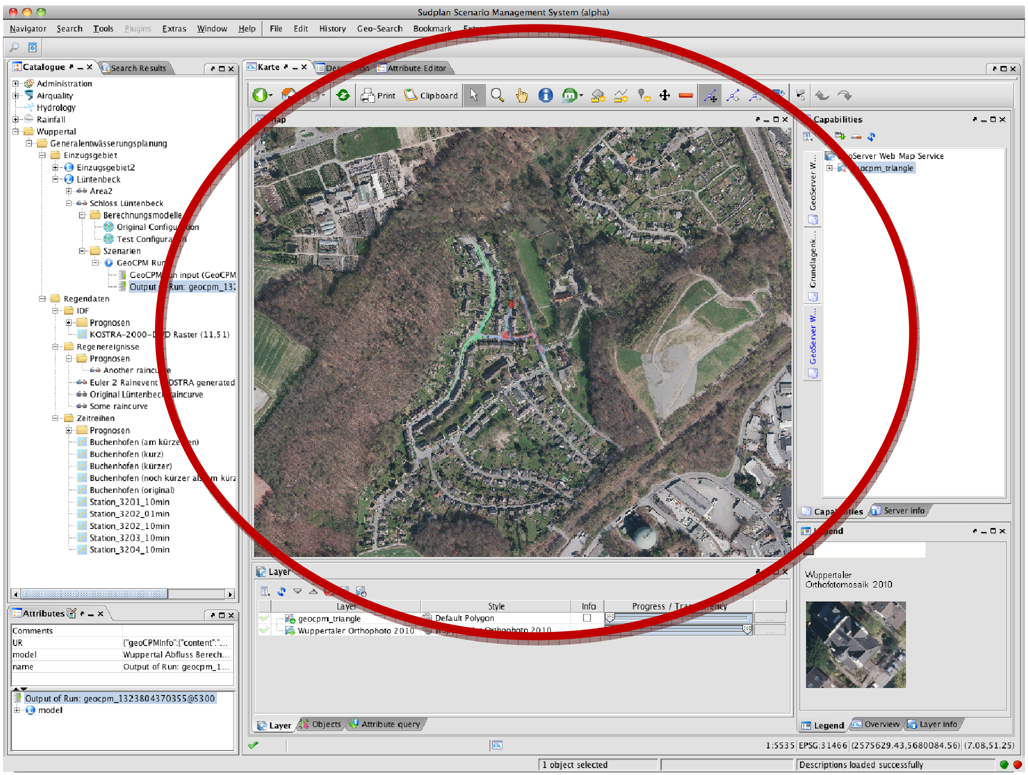


**Figure 36:** View the results on the map

## 4.4. Services

The WP6 pilot makes use of three services:

- **Web Map Service (WMS):** The Web Map Service is a wide spread and often used OGC service that is able to provide rendered images – so-called layers – for a geo-spatial query. As this service is well specified there exist a vast number of WMS clients, that are capable of visualising the service's offerings (e.g. what layers does it provide in what context and characteristics) as well as requesting those information for the desired use-case. On the client side the SMS uses the cismap plugin, which is a full-fledged and easy to use WMS/WFS client implementing various features. For more information on the WMS specification or the cismap WMS client see chapter '**Fel! Hittar inte referenskälla. Fel! Hittar inte referenskälla.**'.

  In the WP6 pilot application the WMS is mainly used to serve layers containing the TINs of the various GeoCPM configurations of the relevant areas of Wuppertal. Additionally it provides layers for the simulation results of the GeoCPM model executions.

- **cids application server:** The cids application server is the main component of the cids platform which is used to implement the SMS. It is used as the backend for cids clients such as the cids Navigator which in turn is the basic rich-client the SMS is built on. It manages SUDPLAN data, makes it accessible through virtual views, such as a tree-like catalogue structure, and provides enhanced search facilities in various contexts, e.g. the classic full-text search or advanced geo-spatial search options. Moreover, it provides a highly configurable, full-fledged role-based access control system and many other features essential for complex information systems. For more information on the cids application server or the whole cids integration platform see chapter '**Fel! Hittar inte referenskälla. Fel! Hittar inte referenskälla.**'.

  In the WP6 pilot application the cids application server is mainly used to provide various pilot data such as IDF curves, time series or rain events as well as GeoCPM configuration data that have to be editable within the SMS, such as the breaklines.

- **GeoCPM Service:** The GeoCPM Service is a proprietary REST (Representational State Transfer) based service wrapper for the GeoCPM and the DYNA model components of the WP6 pilot application. It has been developed specifically to serve as a model management and model control service that perfectly fits the requirements of the WP6 pilot. Thus it is capable of importing new model configurations, running simulations, providing simulation status information and processing of model results. For more information on the GeoCPM Service see chapter '**Fel! Hittar inte referenskälla. Fel! Hittar inte referenskälla.**'.

The design and relationship of the different components is illustrated in Figure 37: Component design. Additionally the major WP6 Pilot workflow design, referring to MD-WP6-7 is outlined in Figure 38: Basic model execution workflow. Please note that this is only a schematic diagram that provides an overview of the data flow in principle.

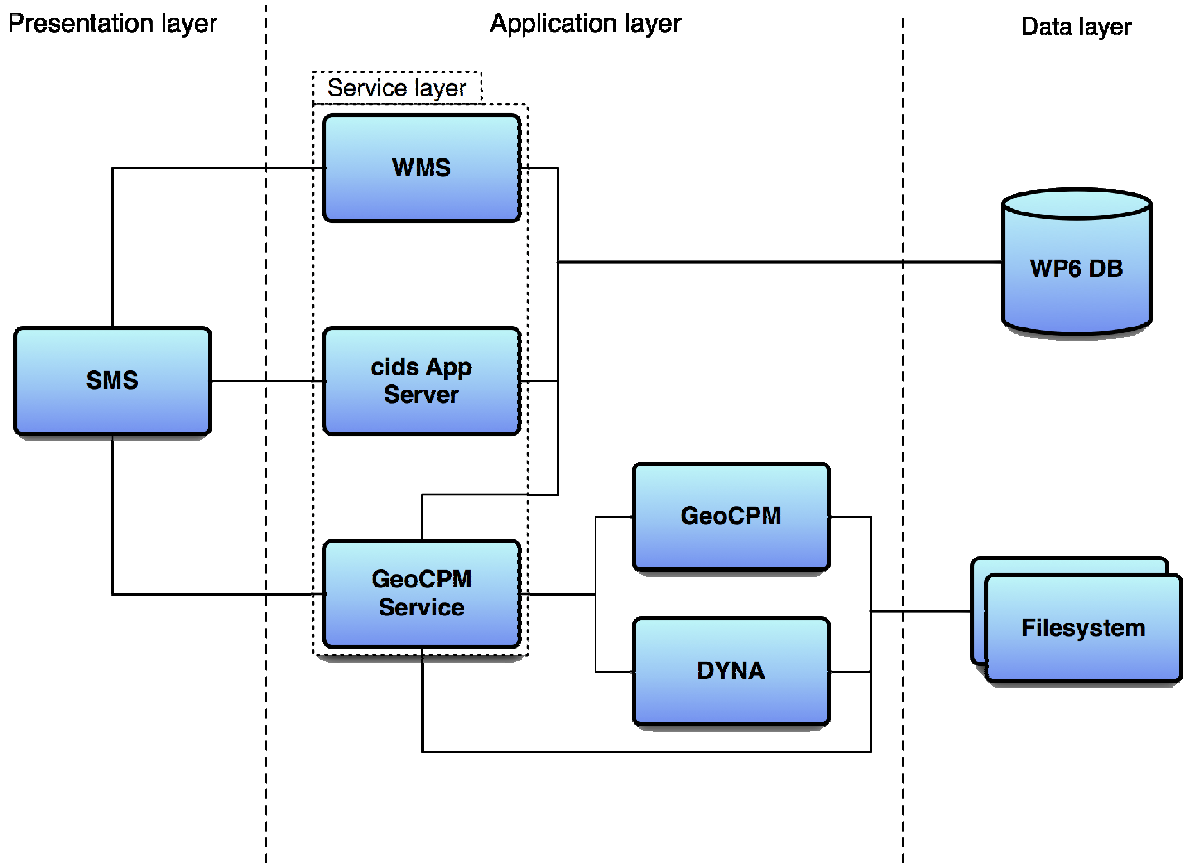


**Figure 37:** Component design

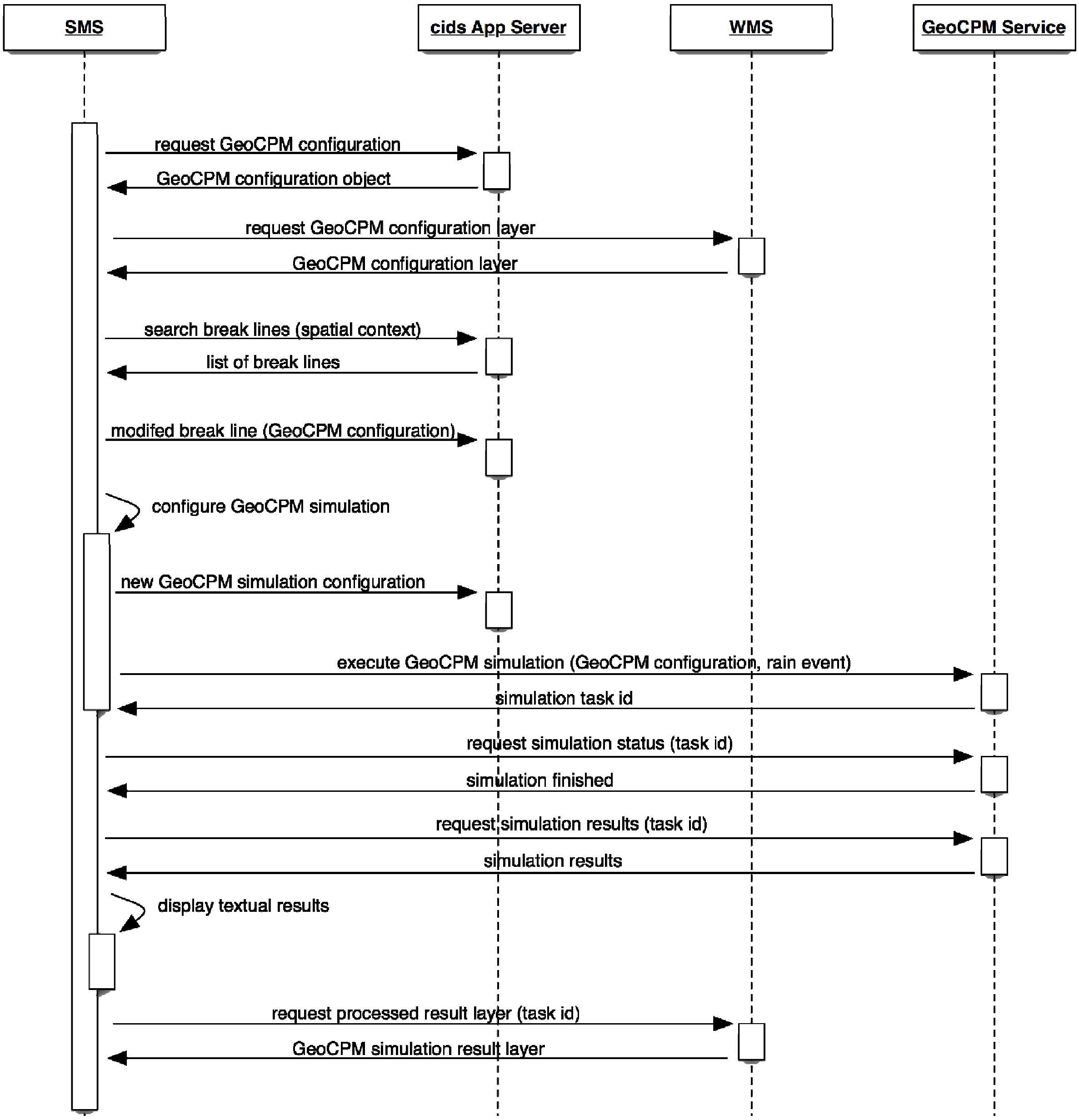


**Figure 38:** Basic model execution workflow

# 5. End User Involvement

End users have been involved on a continuous basis in dedicated workshops and meetings to discuss the required functionality of the Wuppertal pilot application. Since the focus in the second year was on the integration of the Common Services, the SMS and the local models, the most valuable input has come from the primary users who will intensely interact with these components in the future. The users concerning this matter are hydrological modellers employed by the City of Wuppertal, the Wuppertaler Stadtwerke (WSW) and the Wupperverband (WV). As a group they are designated as ‘Storm Water Managers’.

Secondary users, especially urban planners of other departments than Urban Drainage, are supposed to have their focus on the visualizations generated by the SMS, because they need a quick understanding of the outcome of a certain simulation. Hence they will be actively involved in the third year of the project, since the implementation of advanced visualizations is part of the work in that year.

## 5.1. End User Profiles

As stated above, Wuppertal’s **Storm Water Managers** constitute the first group of end users that has to be involved in this phase of the project. These individuals are very comfortable with computers. As modellers they will want to interact with the local models used in the pilot and they may have some familiarity with precipitation models. However, they will likely have little or no experience with climate models. They may or may not have sophisticated GIS experience. The Storm Water Managers form a small group of seven people (City of Wuppertal: 1 person, WSW: 3 persons, WV: 3 persons).

In the course of the general wastewater management process the WSW usually consults engineering companies with the run-off modelling for a certain critical spot. Consultants working for these companies may use the Wuppertal pilot application on behalf of the WSW or the City of Wuppertal in the future. Their professional background is similar to the Storm Water Managers’, but they have additional expertise from project implementations in different sites and under different surrounding conditions. Hence their input is beneficial to design an application that is not only suitable for Wuppertal, but also for other cities

## 5.2. Meeting Log

The following table gives an overview over the end user consultations for the Wuppertal pilot application in the first two years of the SUDPLAN project.

| NO. | Type of meeting | Date | Location | End Users Present | Main Result |
|---|---|---|---|---|---|
| 1 | Workshop on available local models | 10.05.2010 | Wuppertal, DE | *Storm Water Managers:*<br>Bernard Arnold (City of Wuppertal), Marc Scheibel (WV), Udo Laschet (WSW)<br>*Consultants:*<br>Holger Hoppe (DR. PECHER AG), Reinhard beck (IBBeck) | Survey on available local models for surface run-off modeling |
| 2 | Workshop on use cases and visualization (City of Wuppertal and cismet) | 11.05.2010 | Wuppertal, DE | *Storm Water Managers:*<br>Bernard Arnold (City of Wuppertal) | Sketches as background for the mock-up process |
| 3 | Meeting on local model selection (City of Wuppertal and WV) | 17.05.2010 | Wuppertal, DE | *Storm Water Managers:*<br>Bernard Arnold (City of Wuppertal), Marc Scheibel (WV) | Common understanding of use cases for the Wuppertal pilot application in both organizations |
| 4 | Meeting on evaluation of GeoCPM | 16.06.2010 | Erkrath, DE | *Storm Water Managers:*<br>Bernard Arnold (City of Wuppertal), Marc Scheibel (WV), Udo Laschet (WSW)<br>*Consultants:*<br>Holger Hoppe (DR. PECHER AG) | First concept of local model integration |
| 5 | Meeting on local model integration and manipulation of calculation models | 01.02.2011 | Erkrath, DE | *Storm Water Managers:*<br>Bernard Arnold (City of Wuppertal)<br>*Consultants:*<br>Holger Hoppe and Fabian Rost (DR. PECHER AG) | Rough concepts for communication with Common Services and for manipulation of breaklines via SMS |
| 6 | Series of meetings in the context of the general wastewater management process<br>(City of Wuppertal, WSW and WV bilateral) | 03/2011 – 11/2011 | Wuppertal, DE | *Storm Water Managers:*<br>Bernard Arnold (City of Wuppertal), Marc Scheibel (WV), Udo Laschet and Christian Massing (WSW) | Shared understanding of the general wastewater management process, now comprising optional assessment of precautionary measures against inundation by means of the Wuppertal pilot application |

| | | | | | |
|---|---|---|---|---|---|
| 7 | Meeting on integration of Common Services, SMS and local model (GeoCPM) | 24.11.2011 | Erkrath, DE | *Storm Water Managers:*<br>Bernard Arnold (City of Wuppertal)<br>*Consultants:*<br>Holger Hoppe, Fabian Rost and Nora Kirschner (DR. PECHER AG) | Refinement of use cases relating to communication with Common Services and manipulation of breaklines |

**Table 2:** List of end user consultations

# 6. Conclusions

In this document we have given an overview of the Wuppertal Pilot Application functionality available at the end of the year 2011. During the first year of the project the main focus of the Wuppertal pilot work was on the selection of an appropriate local model configuration and the provision of some initial calculation model. The focus of the second year was on integration of the whole chain of components involved in the Wuppertal pilot scenario. A concrete concept for the communication with Common Services and the local models has been established and implemented. This includes the wrapping of the local model components (Service Wrapping), a customisation of the local models with regard to the pilot use cases defined in the Wuppertal Pilot Definition PlanV2 (e.g. modification of the topographic model). Moreover, an instance of the SMS has been be deployed that integrates the local models DYNA and GeoCPM, which allows users to change the model parameterisation (e.g. raise road kerbs) and provides a workflow support for the planning process. It also enables users to access and include climate change information through the integration of Common Services. Moreover, the SMS supports appropriate visualisation methods for user input data. So far considerable progress has been made and the software (with enhancements and additional features) will go live in the course of 2012.

 

# 7. References


| | |
|---|---|
| [1] SUDPLAN D4.1.1 | D4.1.1 Common Services concerted approach V1.pdf, SUDPLAN Consortium, 09.08.2010 |
| [2] SUDPLAN D6.1.2 | D6.1.2 WP6 Pilot Definition Plan V2.pdf, SUDPLAN Consortium, 15.11.2011 |
| [3] GeoCPM Interface Spec | Schnittstellenbeschreibung GeoCPM – hydrodynamische Berechnung des Oberflächenabflusses Version 0.5, tandler.com GmbH, 05.10.2011 |
| [4] DYNA Interface Spec | DYNA Benutzerhandbuch Version 4, Pecher Software GmbH |
| [5] OGC WMS Spec | OpenGIS Web Map Service Implementation Specification V1.3.0, Open Geospatial Consortium, 15.03.2006, http://www.opengeospatial.org/standards/wms |
| [6] cismap WMS client | cismap Web Map Service client reference manual V2.0, cismet GmbH, 2009, http://cismet.de/en/products.html |
| [7] cids Integration Platform | cids Integration Platform factsheet, cismet GmbH, 2009, http://cismet.de/en/products.html |
| [8] Jersey JAX-RS | Jersey JAX-RS reference implementation for building RESTful Web services, Oracle, 2011, http://jersey.java.net/ |
| [9] JSON RFC | JSON format specification (RFC 4627), The Internet Society, 2006, http://www.ietf.org/rfc/rfc4627.txt?number=4627 |
| [10] Java Programming language | Java Programming language, Oracle, 2011, http://java.com/en/ |

# 8. Glossary

| *technical term* | *Explanation* |
|---|---|
| AquaZIS | software tool for the management and evaluation of time series data, produced by the German company aqua_plan GmbH |
| AQZ-ASCII | proprietary data format for precipitation data in the form of equidistant or variable time series, defined by aqua_plan GmbH |
| Calculation Model | a high-resolution digital elevation model in the form of a triangulated irregular network (TIN) that is optimised for the calculation of hydrodynamic surface run-off, therefore it comprises a detailed model of all relevant man-made breaklines like the exterior walls of buildings and road kerbs |
| DYNA | software component for ++SYSTEMS, used for the simulation of hydrologic surface run-off and hydrodynamic analysis of sewage systems (manufacturer: tandler.com GmbH / Pecher AG) |
| ED-format | proprietary data format for precipitation data in the form of time variation curves, used by software products of German company itwh GmbH |
| ES-format | standardised interchange data format for precipitation data (‘Einheitliche Schnittstelle für die Weitergabe von Regendaten’) defined by ‘Deutscher Verband für Wasserwirtschaft und Kulturbau e.V. (DVWK)’ |
| GeoCPM | software component for ++SYSTEMS, used for the simulation of hydrodynamic surface run-off<br>(manufacturer: tandler.com GmbH / Pecher AG) |
| IDF curve | Intensity-Duration-Frequency curve for rainfall data – an IDF curve indicates the rainfall intensity that will occur for a given duration and return period |
| LF-format | proprietary but widely-used data (interchange) format for precipitation data in the form of equidistant time series, used by the software ‘LWAFLUT’ (manufacturer: Hydrotec GmbH) |
| MD-format | proprietary data format for precipitation data in the form of equidistant time series, used by software products of German company itwh GmbH |
| NASIM-format | proprietary data format for precipitation data in the form of time series with 6 minutes equidistance, used by the software ‘NASIM’ (manufacturer: Hydrotec GmbH) |
| SM-format | proprietary but widely-used data (interchange) format for precipitation data in the form of equidistant time series, used by the software ‘Schmutzfracht-simulationsmodell SMUSI’ (manufacturer: Hessisches Landesamt für Umwelt und Geologie) |
| ++SYSTEMS | geographical information system with subject-specific modules for all aspects of the sewerage domain (manufacturer: tandler.com GmbH / Pecher AG) |
| UV-format | proprietary but widely-used data (interchange) format for precipitation data in the form of equidistant or variable time series, defined by German company Hydrotec GmbH |

# 9. Acronyms and Abbreviations

| *acronym / abbreviation* | *Definition* |
|---|---|
| DBF | dBASE Data Base File |
| DEM | Digital Elevation Model |
| GIS | Geographical Information System |
| GUI | Graphical User Interface |
| PDP | Pilot Definition Plan |
| SMS | Scenario Management System |
| TIN | Triangulated Irregular Network |
| WSW | Wuppertaler Stadtwerke (municipal utility) |

# 10. Appendix A: GeoCPM Service specification

## 10.1. Management Summary

This documentation serves as an implementation specification of the GeoCPM Service for the WP6 Pilot Application. The GeoCPM Service is a REST (Representational State Transfer) based web service component that wraps the GeoCPM and DYNA local model and enables client applications to easily add model input data as well as to manage model executions. Moreover, it processes results of model runs and can provide this data on request.

In contrast to other Pilots (e.g. the WP7 Pilot), which make use of the standardised Sensor Observation Service and Sensor Planning Service, this implementation is proprietary as e.g. this data is not intended to be published outside of the Wuppertal Pilot Application domain and thus it is specifically tailored to the Pilot needs. The reasons for this decision are:

- ***Demonstration of Flexibility:*** As this service is a tailored solution it demonstrates that it is very easy and cost-effective to integrate nearly any kind of model wrapping service in the well designed versatile SMS model integration framework

- ***Effectiveness:*** SOS is mainly suited to serve observed sensor data, mostly time series. Although it would be possible to use SOS for WP6 data setup and usage is rather much overhead, so using SOS here would like taking a sledgehammer to crack a nut. SPS is more likely fitting to control model runs than the SOS is to control model input and output data. However, as it is suited to the needs of sensor networks and their tasking and because of unnecessary schema overhead, using a tailored service is more effective.

- ***Ease of use:*** SOS and SPS are services suited to manage and access Sensor Networks and not necessarily to control arbitrary software components and their I/O data. However, their interface is general enough to provide such functionality. This comes with the price of complex, inflated and generic I/O messages which complicate the development of simple, easy to use client APIs.

## 10.2. Component usage

The GeoCPM Service makes use of several components to realise UC-1114:

- ***GeoCPM model:*** This is the component that is used for the simulation of the runoff. It is a Visual C++ based implementation and thus is limited to Microsoft Windows OS. The I/O of the GeoCPM model is file-based, thus it requires an input configuration to run in a specific format, defined in [3]. The model results are presented in various output files, defined in [3].

- ***DYNA model:*** This component supports the GeoCPM model component by enhancing the simulation with the comprehension of the effect of the sewer system efficiency during heavy rainfall. DYNA has been implemented in FORTRAN and is compiled for Microsoft Windows OS's. It uses specifically formatted input files, defined in [4]. As of the supportive nature of DYNA in the WP6 context the DYNA output is not of immediate interest.

- ***File System:*** The GeoCPM Service makes use of the local file system to provide the model components with input files and to access model results. Moreover the service has to be able to start the models in a local process. Please note that although these statements imply that the GeoCPM Service is restricted to run on the same machine as the model components this is not the case. There are various ways to access machines remotely.

- ***Postgresql DBMS:*** The Postgresql DMBS is the main data backend of the WP6 Pilot Application. It stores all kind of Pilot related data and thus model configurations, too. It is extended by PostGIS to add support for management and processing of geo-spatial objects.

- ***Generic Process Launcher:*** The Generic Process Laucher is a generic component that is able to launch arbitrary windows executables and can provide extended process information, regardless of the implementation specifics of the executable. This component is a C# implementation and is thus able to not only provide information such as process ids and process exit codes but also information exposed on the Windows Event log. This is extremely useful when launched processes crash due to any issues.

Figure 39: **GeoCPM Service and related components** shows the relation between the different components:

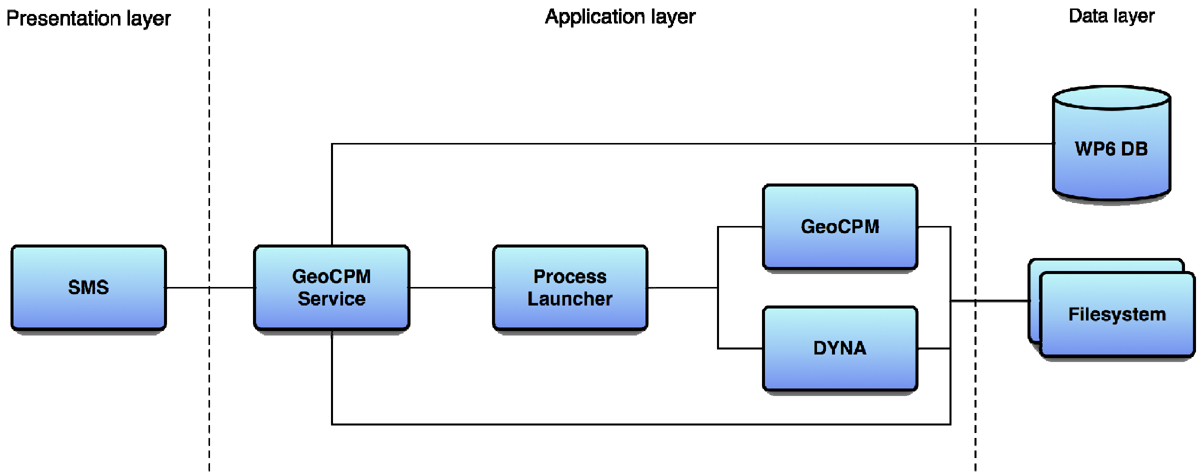


**Figure 39:** GeoCPM Service and related components

## 10.3. GeoCPM workflows

The GeoCPM Service covers various workflows related to the realisation of UC-1114:

- ***Import model configuration***
- ***Start simulation***
- ***Request simulation status***
- ***Request simulation results***
- ***Clean simulation workspace***

The following sections explain these workflows in greater detail.

### 10.3.1 Import model configuration

In order to be able to manipulate and run various simulations for arbitrary areas the GeoCPM Service provides means to import model configurations into the SMS, thus making them available to it. A model configuration consists of a valid GeoCPM configuration; optionally accompanied by a related DYNA configuration in case of the ability to run a combined simulation for the area shall be available, too. The steps that are involved in this workflow are the following:

- *Choose configuration files:* The SMS offers the user to choose configuration files that shall be made available for manipulation and execution.
- *Send configuration data:* The SMS sends the new GeoCPM configuration and optionally the related DYNA configuration.
- *Process configuration data:* The GeoCPM Service parses the data so that it can be stored in the relational backend.
- *Store configuration data:* If the configuration data was parsed successfully it will be store in the database and is then available to the SMS.
- *Return import status message:* The GeoCPM Service sends the status of the import the SMS: ‘success’ if everything went well and the data is available or ‘error’ in case of issues during import.

 

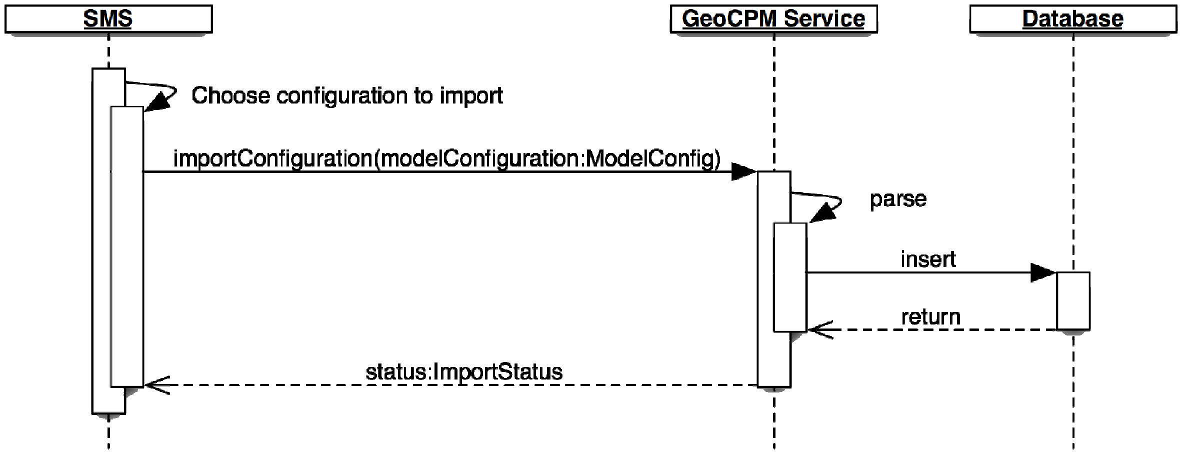


**Figure 40:** Model configuration import workflow

### 10.3.2 Start simulation

One of the most important tasks is to be able to actually start a GeoCPM simulation. For that the GeoCPM Service is instructed by the SMS to start a simulation with the provided run parameters. The run parameters contain information about which GeoCPM configuration to use, whether to do a combined execution (to use DYNA, too) and which rain event to use. Then the GeoCPM Service will do a number of steps:

- *Assemble simulation configuration:* The SMS provides the user the possibility to choose the parameters of the simulation. He may choose which configuration shall be used for the simulation, which rain event is investigated and whether sewer system data shall be used, too.

- *Start simulation:* The SMS sends the assembled simulation configuration to the GeoCPM Service. The Service checks the configuration for validity and complains if necessary.

- *Prepare model execution:* GeoCPM and DYNA work on basis of files. Hence the GeoCPM Service assembles a valid GeoCPM input file from the data in the database (and a valid DYNA input file in case of a combined run) and stores it in a temporary model execution folder on the local file system.

- *Execute model:* The GeoCPM Service uses the Process Launcher to execute the local model (either GeoCPM or DYNA). It receives a process id for the execution or is notified that the model could not be started for various reasons.

- *Return model execution status:* After the execution step the Service generates a task identifier and returns an execution status message to the SMS. The execution status contains this very task identifier and additional information on the model execution such as ‘started’ or ‘broken’.

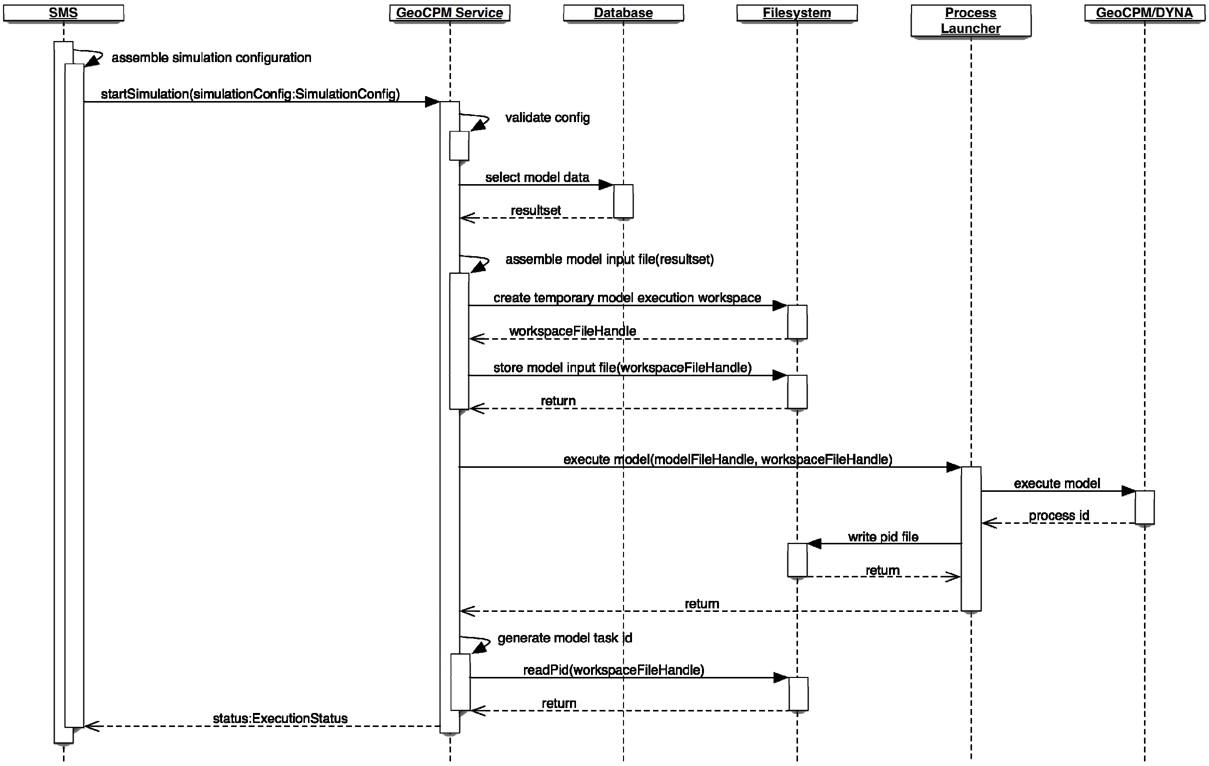


**Figure 41:** Start simulation workflow

### 10.3.3 Request simulation status

In order to provide the user with valuable information on the current status of his running simulations the GeoCPM service offers the possibility to request the simulation status for GeoCPM/DYNA executions. The GeoCPM service requires the task identifier obtained during the ***'Start simulation'*** workflow.

- *Show task execution status:* The user selects a simulation task and wants to be provided with information on its status.

- *Request execution status:* The SMS sends the task identifier of the selected simulation task to the GeoCPM Service. The task identifier was previously obtained as a result of the ***'Start simulation'*** workflow. The GeoCPM Service fetches the execution workspace handle from the task identifier information or complains if the task identifier is invalid.

- *Read execution status:* The GeoCPM service uses the workspace handle to read the execution status.

- *Return model execution status:* The execution status message is returned to the SMS. It contains the task identifier and information on the model execution such as 'running' or 'broken'.

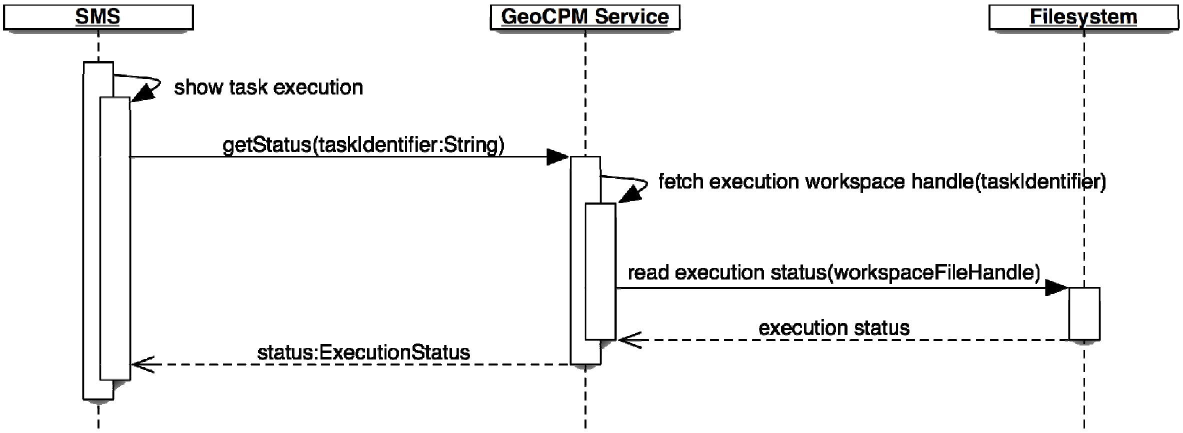


**Figure 42:** Request simulation status workflow

### 10.3.4 Request simulation results

Besides the '***Start simulation***' workflow this is another very important workflow as the SMS needs the results to be displayed. So the GeoCPM Service offers an operation that allows for obtaining results of a simulation, identified by the task identifier, that was created as a result of '***Start simulation***'. However, this operation not only provides an immediate result, such as the raw simulation results, it also processes the results so that they can be served by a WMS. In the following all the necessary steps are listed:

- *Download simulation results:* As soon as the SMS receives the 'finished' status the results are available. It can then offer the user to download the results.

- *Get results:* The SMS requests the results from the GeoCPM Service for a specific task identifier. The GeoCPM Service will complain if the task identifier is invalid or the task is not finished successfully (yet).

- *Process results:* The GeoCPM Service uses the task identifier to fetch the execution workspace and tries to read the results. The results are processed and inserted in the backing database so that a new WMS layer will be available.

- *Return simulation results:* After the WMS layer data has been created the GeoCPM Service assembles a message containing information about the WMS data, that is available and also includes some statistical information of this simulation and sends this to the SMS.

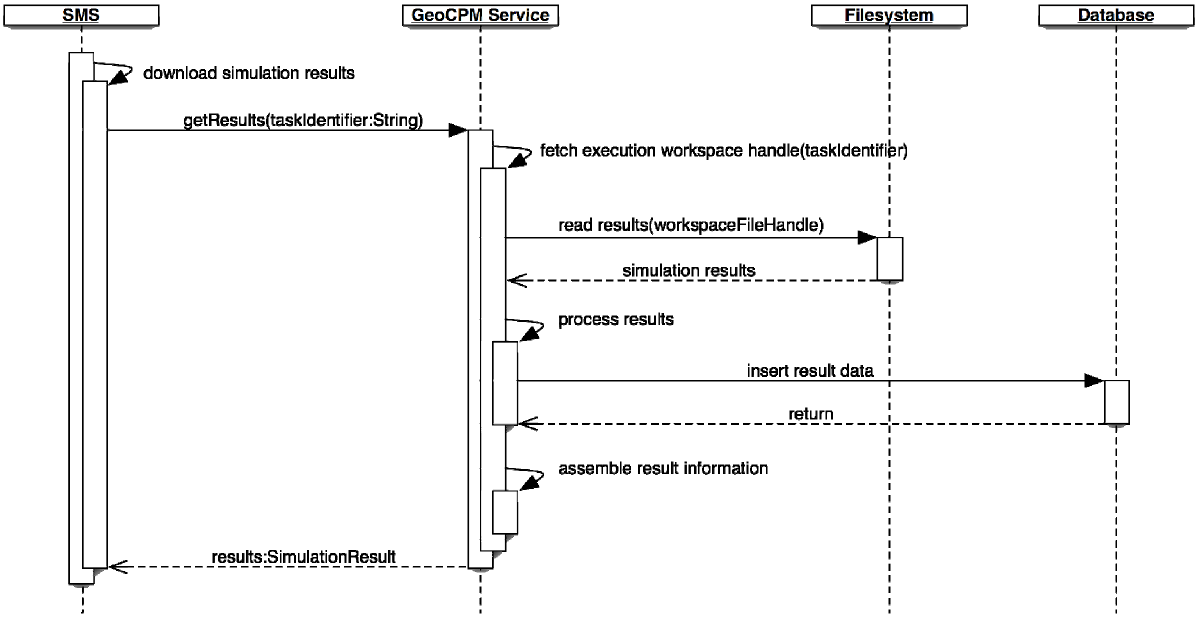


**Figure 43:** Get simulation results

### 10.3.5 Clean simulation workspace

Results of a simulation will not automatically be deleted after they have been requested by the SMS. This is because e.g. the SMS could have encountered issues while processing the result. In other words the data could be lost after a request for any reason. Thus the result data can be requested as long as no cleanup has been done. The cleanup causes all data related for a given task identifier to be removed. Consequently the task identifier will become invalid after a successful cleanup request.

- *Download simulation results:* The '***Request simulation results***' workflow is done successfully and thus the necessary data is available to the SMS.

- *Clean simulation workspace:* The SMS checked the integrity of the downloaded results and cleans the simulation workspace using the same task identifier that was used for the download. If the task identifier is invalid the GeoCPM Service will complain.

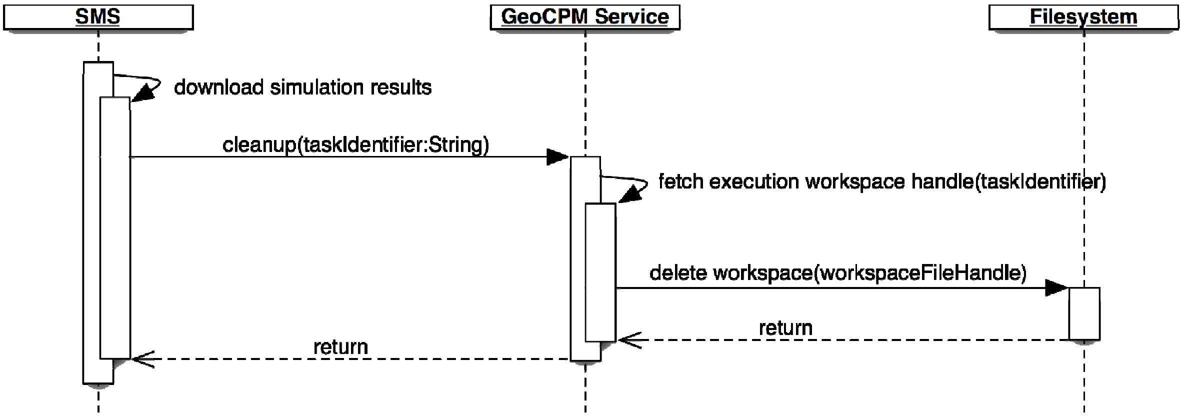


**Figure 44:** Cleanup simulation workspace

## 10.4. GeoCPM service interface

This section describes the interface of the GeoCPM Service. The GeoCPM Service is a RESTful web service implemented using Jersey API [8], the JAX-RS (JSR 311) Reference implementation. It uses POJOs with JAXB via JSON for I/O.

The server base resource path is http://sudplanwp6.cismet.de/GeoCPM

All the listed operations are synchronous by nature, so it is highly encouraged to use a separate thread for the requests. The implementation of the service is thread-safe.

Java default types will not be explained in greater detail. For more information on Java default types see JDK documentation [10].

### 10.4.1 Operations

<table>
<tr><td>Name</td><td colspan="3">importConfiguration</td></tr>
<tr><td>Description</td><td colspan="3">This operation imports GeoCPM and DYNA configuration data.</td></tr>
<tr><td>Operation signature</td><td colspan="3">ImportStatus importConfiguration(ImportConfig cfg) throws<br>    GeoCPMException,<br>    IllegalArgumentException</td></tr>
<tr><td>Resource path</td><td colspan="3">/importConfiguration</td></tr>
<tr><td>Supported methods</td><td colspan="3">HTTP POST</td></tr>
<tr><td>Accepts media</td><td colspan="3">application/json</td></tr>
<tr><td>Produces media</td><td colspan="3">application/json</td></tr>
<tr><td>Input parameters</td><td>cfg</td><td>ImportConfig</td><td>The GeoCPM and DYNA configuration data</td></tr>
<tr><td>Output parameter</td><td colspan="2">ImportStatus</td><td>In case of an successful import it contains the newly created GeoCPM configuration identifier</td></tr>
<tr><td rowspan="2">Exceptions</td><td colspan="2">GeoCPMException</td><td>If any error occurs during the configuration import, mapped to HTTP status code: 550</td></tr>
<tr><td colspan="2">IllegalArgumentException</td><td>If the given ImportConfig is null or invalid, mapped to HTTP status code: 450</td></tr>
<tr><td>Example usage</td><td colspan="3">final GeoCPMRestClient client = new<br>    GeoCPMRestClient(http://sudplanwp6.cismet.de/GeoCPM);<br>final File geocpmCfg = new File(“geocpmCfg”);<br>final File dynaCfg = new File(“dynaCfg”);<br>final String geocpmData = [… file io …];<br>final String dynaData = [… file io …];<br>final ImportConfig importCfg = new ImportConfig();<br>importCfg.setGeocpmData(geocpmData);<br>importCfg.setDynaData(dynaData);<br>final ImportStatus status =<br>    client.importConfiguration(importCfg);</td></tr>
</table>

| | |
|---|---|
| **Name** | ***startSimulation*** |
| **Description** | This operation starts a simulation with the given configuration |

| | | | |
|---|---|---|---|
| **Operation signature** | `ExecutionStatus startSimulation(SimulationConfig cfg) throws`<br>`GeoCPMException,`<br>`IllegalArgumentException` | | |
| **Resource path** | /startSimulation | | |
| **Supported methods** | HTTP POST | | |
| **Accepts media** | application/json | | |
| **Produces media** | application/json | | |
| **Input parameters** | *cfg* | SimulationConfig | The configuration for this simulation containing information about the GeoCPM configuration to use, which rain event etc. |
| **Output parameter** | *ExecutionStatus* | | Information about the execution, such as the task identifier and whether it is running or finished etc. |
| **Exceptions** | GeoCPMException | | If any error occurs during reading the simulation start setup, mapped to HTTP status code: 550 |
| | IllegalArgumentException | | If the given SimulationConfig is null or invalid, mapped to HTTP status code: 450 |
| **Example usage** | `final GeoCPMRestClient client = new`<br>`GeoCPMRestClient(http://sudplanwp6.cismet.de/GeoCPM);`<br>`final Rainevent rainevent = [… load event …]`<br>`final SimulationConfig cfg = new SimulationConfig();`<br>`cfg.setGeocpmCfg(387);`<br>`cfg.setRainevent(rainevent);`<br>`cfg.setCombinedRun(true);`<br>`final ExecutionStatus status =`<br>`client.startSimulation(cfg);` | | |

| | | | |
|---|---|---|---|
| **Name** | ***getStatus*** | | |
| **Description** | This operation requests the status for a previously started simulation. If the task identifier does not refer to a task an IllegalArgumentException will be thrown. | | |
| **Operation signature** | `ExecutionStatus getStatus(String taskId) throws`<br>`GeoCPMException,`<br>`IllegalArgumentException` | | |
| **Resource path** | /getStatus | | |
| **Supported methods** | HTTP POST | | |
| **Accepts media** | application/json | | |
| **Produces media** | application/json | | |
| **Input parameters** | *taskId* | String | The task identifier that was included in the 'startSimulation' response. |
| **Output parameter** | *ExecutionStatus* | | Information about the execution, such as the task identifier and whether it is running or finished etc. |
| **Exceptions** | GeoCPMException | | If any error occurs during reading the status from the workspace, mapped to HTTP status code: 550 |
| | IllegalArgumentException | | If the given task identifier is null or invalid, mapped to HTTP status code: 450 |

**Example usage**

```
final GeoCPMRestClient client = new
    GeoCPMRestClient(http://sudplanwp6.cismet.de/GeoCPM);
final Rainevent rainevent = [… load event …]
final SimulationConfig cfg = new SimulationConfig();
cfg.setGeocpmCfg(387);
cfg.setRainevent(rainevent);
cfg.setCombinedRun(true);
final ExecutionStatus status =
    client.startSimulation(cfg);

final ExecutionStatus currentStatus =
    client.getStatus(status.getTaskId());
```

| **Name** | ***getResults*** | | |
|---|---|---|---|
| **Description** | This operation requests the results of a previously started simulation. The results will contain some statistical information as well as information which WMS layer was created during the result processing. If the task identifier does not refer to a task an IllegalArgumentException will be thrown. If the task does not refer to a finished task an IllegalStateException will be thrown | | |
| **Operation signature** | `SimulationResult getResults(String taskId) throws GeoCPMException, IllegalArgumentException, IllegalStateException` | | |
| **Resource path** | /getResults | | |
| **Supported methods** | HTTP POST | | |
| **Accepts media** | application/json | | |
| **Produces media** | application/json | | |
| **Input parameters** | *taskId* | String | The task identifier that was included in the 'startSimulation' response. |
| **Output parameter** | *SimulationResult* | | Statistical results of the simulation or a |
| **Exceptions** | GeoCPMException | | If any error occurs during reading and processing the simulation from the workspace, mapped to HTTP status code: 550 |
| | IllegalArgumentException | | If the given task identifier is null or invalid, mapped to HTTP status code: 450 |
| | IllegalStateException | | If the given task identifier refers to a task that is still running, mapped to HTTP status code 451 |
| **Example usage** | (code below) | | |

```
final GeoCPMRestClient client = new
    GeoCPMRestClient(http://sudplanwp6.cismet.de/GeoCPM);
final Rainevent rainevent = [… load event …]
final SimulationConfig cfg = new SimulationConfig();
cfg.setGeocpmCfg(387);
cfg.setRainevent(rainevent);
cfg.setCombinedRun(true);
final ExecutionStatus status =
    client.startSimulation(cfg);

final ExecutionStatus currentStatus =
    client.getStatus(status.getTaskId());
```

| | |
|---|---|
| | `final SimulationResult results =`<br>`      client.getResults(status.getTaskId());` |

| Name | *cleanup* | | |
|---|---|---|---|
| **Description** | This operation cleans the workspace of a previously finised simulation. If the task identifier does not refer to a task an IllegalArgumentException will be thrown. If the task does not refer to a finished task an IllegalStateException will be thrown | | |
| **Operation signature** | `void cleanup(String taskId) throws`<br>`    GeoCPMException,`<br>`    IllegalArgumentException,`<br>`    IllegalStateException` | | |
| **Resource path** | /cleanup | | |
| **Supported methods** | HTTP POST | | |
| **Accepts media** | application/json | | |
| **Input parameters** | *taskId* | String | The task identifier that was included in the 'startSimulation' response. |
| **Exceptions** | GeoCPMException | | If any error occurs during workspace cleanup, mapped to HTTP status code: 550 |
| | IllegalArgumentException | | If the given task identifier is null or invalid, mapped to HTTP status code: 450 |
| | IllegalStateException | | If the given task identifier refers to a task that is still running, mapped to HTTP status code 451 |
| **Example usage** | `final GeoCPMRestClient client = new`<br>`    GeoCPMRestClient(http://sudplanwp6.cismet.de/GeoCPM);`<br>`final Rainevent rainevent = [… load event …]`<br>`final SimulationConfig cfg = new SimulationConfig();`<br>`cfg.setGeocpmCfg(387);`<br>`cfg.setRainevent(rainevent);`<br>`cfg.setCombinedRun(true);`<br>`final ExecutionStatus status =`<br>`    client.startSimulation(cfg);`<br><br>`final ExecutionStatus currentStatus =`<br>`    client.getStatus(status.getTaskId());`<br><br>`final SimulationResult results =`<br>`      client.getResults(status.getTaskId());`<br>`client.cleanup(status.getTaskId());` | | |

### 10.4.2 Complex types

All the listed types are 'Bean' types, thus containing getters and setters for all the members.

| Name | *ImportConfig* | | |
|---|---|---|---|
| **Description** | This type contains GeoCPM and DYNA configuration data. | | |
| **Member** | *geocpmData* | String | The content of a GeoCPM configuration file according to GeoCPM interface specification [3], must never be null |
| **Member** | *dynaData* | String | The content of a DYNA configuration file, |

| | | | |
|---|---|---|---|
| | | | according to DYNA interface specification [4], or null if there is no associated DYNA configuration |

| **Name** | ***ImportStatus*** | | |
|---|---|---|---|
| **Description** | This type contains information about the import status. | | |
| **Member** | *geocpmId* | int | The id of the imported GeoCPM configuration. It is e.g. used to start a new simulation. |

| **Name** | ***ExecutionStatus*** | | |
|---|---|---|---|
| **Description** | This type contains information about the current execution status. | | |
| **Member** | *status* | String | The status of the simulation execution, one of:<br>"**STARTED**" if simulation has just started<br>"**FAILED**" if simulation could not be started<br>"**RUNNING**" if simulation is running<br>"**FINISHED**" if simulation is finished<br>"**BROKEN**" if simulation execution stopped due to errors.<br>never null |
| **Member** | *taskId* | String | The task identifier for the simulation, never null |

| **Name** | ***SimulationConfig*** | | |
|---|---|---|---|
| **Description** | This type contains information necessary to start a new GeoCPM simulation | | |
| **Member** | *geocpmCfg* | int | The id of the imported GeoCPM configuration, that is included in the ImportStatus response |
| **Member** | *rainevent* | Rainevent | The rainevent that shall be used for this simulation, never null |
| **Member** | *combinedRun* | boolean | Flag whether to do a combined run (GeoCPM and DYNA) or a single run (GeoCPM only) |

| **Name** | ***Rainevent*** | | |
|---|---|---|---|
| **Description** | This type contains the data of the rain event that is used for a simulation | | |
| **Member** | *secondsToMm* | Map<Integer, Double> | The rain event data map, whereas seconds are mapped to rain amount in millimeters, such as 300s -> 3.4mm, never null |

| **Name** | ***SimulationResult*** | | |
|---|---|---|---|
| **Description** | This type contains statistical results of a GeoCPM simulation as well as a reference to the newly created WMS result layer | | |
| **Member** | *geocpmInfo* | String | Statistical results of the GeoCPM simulation, never null |
| **Member** | *wmsResults* | URL | The generated WMS result layer URL, never null |
| **Member** | *taskId* | String | The task identifier for the simulation, never null |

 

## 10.5. GeoCPM client API

The GeoCPM Service can be accessed through an easy-to-use Client API. The API implements all the operations and data types described in section 10.4.1 Operations and 10.4.2 Complex types. It uses Jersey Client API to access the GeoCPM Service.

The GeoCPM Client API is synchronous and thread-safe.

### 10.5.1 Example Usage

```
public final class Sample {

    //~ Methods ----------------------------------------------------------------

    /**
     * DOCUMENT ME!
     *
     * @throws  RuntimeException  DOCUMENT ME!
     */
    void example() {
        // read configuration(s) to import
        BufferedReader geocpmR = null;
        BufferedReader dynaR = null;

        final StringBuilder geocpmData = new StringBuilder();
        final StringBuilder dynaData = new StringBuilder();
        try {
            geocpmR = new BufferedReader(new FileReader(new File("geocpm.in")));
            dynaR = new BufferedReader(new FileReader(new File("dyna.in")));

            String line;
            while ((line = geocpmR.readLine()) != null) {
                geocpmData.append(line);
            }

            while ((line = dynaR.readLine()) != null) {
                dynaData.append(line);
            }
        } catch (final IOException e) {
            throw new RuntimeException("cannot read config data", e);
        } finally {
            if (geocpmR != null) {
                try {
                    geocpmR.close();
                } catch (final IOException e) {
                    // ignore
                }
            }

            if (dynaR != null) {
                try {
                    dynaR.close();
                } catch (final IOException e) {
                    // ignore
                }
            }
        }

        // create a new client for the GeoCPM Service
        final GeoCPMRestClient client =
            new GeoCPMRestClient("http://sudplanwp6.cismet.de/GeoCPM");

        // assemble import configuration
        final ImportConfig importConfig =
            new ImportConfig(geocpmData.toString(), dynaData.toString());

        // send import data, may take long
```

```
        final int geocpmId;
        try {
            final ImportStatus status = client.importConfiguration(importConfig);
            geocpmId = status.getGeocpmId();
        } catch (final Exception e) {
            throw new RuntimeException("cannot import new configuration", e);
        }

        // start a new simulation
        final String taskId;
        try {
            final Rainevent rainevent = loadRainEvent();
            final SimulationConfig simulationConfig =
                new SimulationConfig(geocpmId, rainevent, true);
            final ExecutionStatus initStatus = client.startSimulation(simulationConfig);
            if ("FAILED".equals(initStatus.getStatus())) {
                throw new IllegalStateException("the simulation start failed");
            } else {
                taskId = initStatus.getTaskId();
            }
        } catch (final Exception e) {
            throw new RuntimeException("cannot start simulation", e);
        }

        // poll the status, a simulation may take several hours
        ExecutionStatus currentStatus = null;
        try {
            while ("RUNNING".equals(currentStatus.getStatus())) {
                Thread.sleep(1000);
                currentStatus = client.getStatus(taskId);
            }
        } catch (final Exception e) {
            throw new RuntimeException("error while waiting for simulation to finish", e);
        }

        // if the status is not finised here, there was an error
        if (!"FINISHED".equals(currentStatus.getStatus())) {
            throw new RuntimeException("current status not finished, run must be erroneous");
        }

        // fetch the run results
        final SimulationResult result;
        try {
            result = client.getResults(taskId);
        } catch (final Exception e) {
            throw new RuntimeException("error while waiting for simulation to finish", e);
        }

        // cleanup, because the result fetching was ok
        try {
            client.cleanup(taskId);
        } catch (final Exception e) {
            throw new RuntimeException("could not clean up finised simulation", e);
        }

        visualiseResultLayerInWMS(result.getWmsResults());
        visualiseStatisticalResultsInGui(result.getGeocpmInfo());
    }
}
```