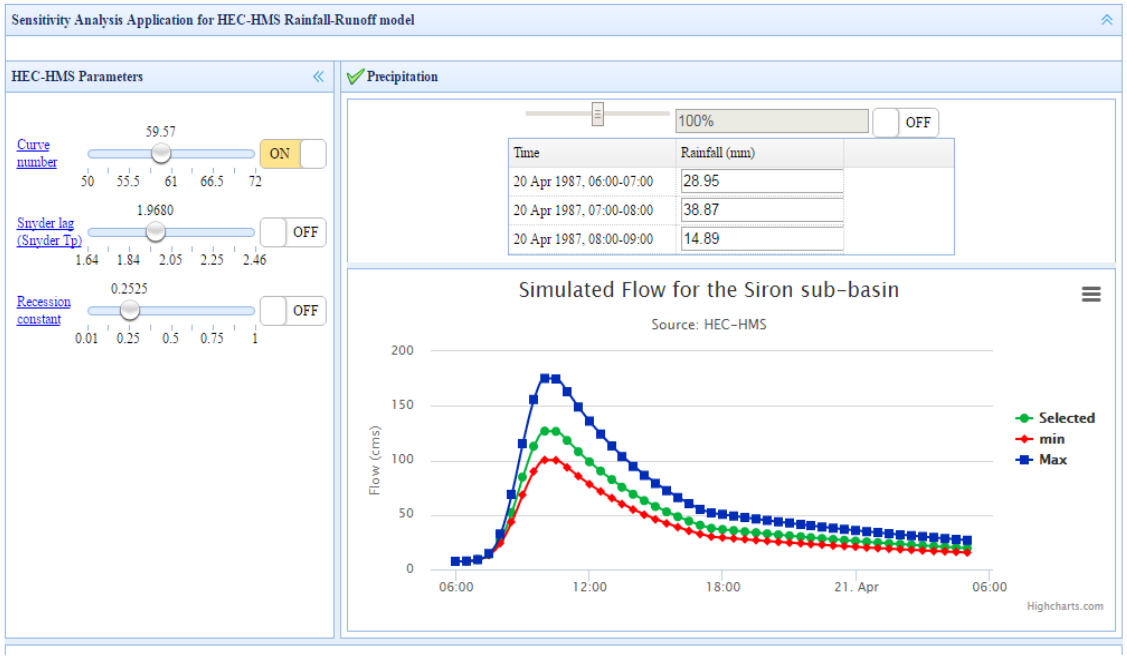
Summer School - **„Hydroinformatics approaches for river basin related problems" 21st June - 8th July, 2016 organised by UNESCO-IHE Institute for Water Education and University Politehnica of Bucharest**

Assignment: **Development of a Sensitivity Analysis Application for an event-based HEC-HMS Rainfall-Runoff model**

Team: Sorin Ciolofan, Adrian Visanu

(Politehnica University of Bucharest, Automatic Control and Computer Science Faculty)



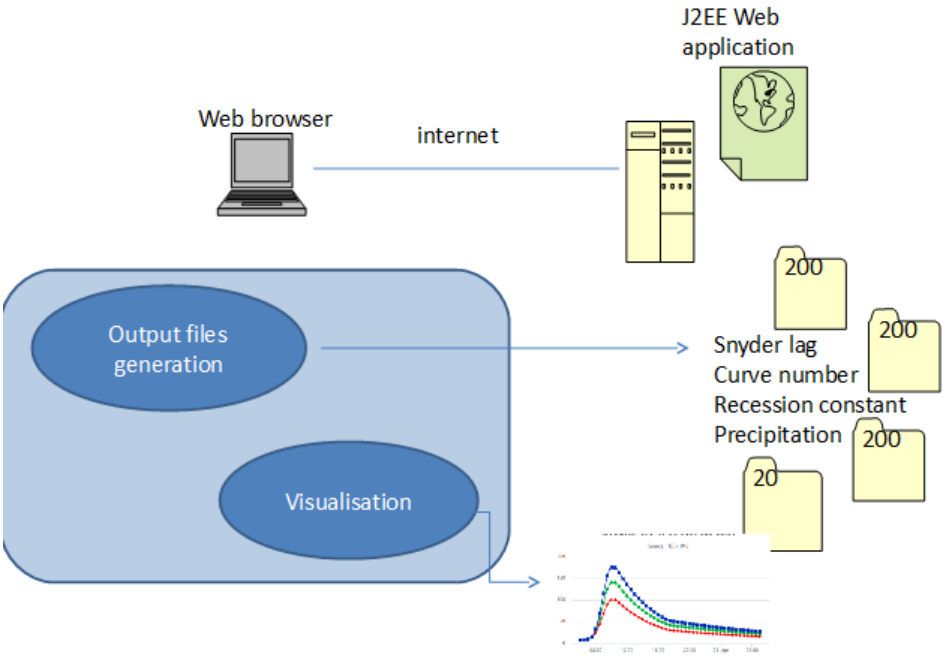
TECHNICAL REPORT

We implemented a J2EE web application considering that this approach has a few major advantages over a Desktop application:

- A web application can be accessed by anyone from a web browser assuming an Internet connection is working (A desktop application is usually used only by the users who have access to the computer where the application is installed locally)
- For a web application the end-user does not have to install on his machine HEC-HMS or HEC-Vue utilities. These are installed on the server where the web application is hosted.

We have a typical client-server application where the application is hosted on a web server (Apache Tomcat) and client has a browser and a working Internet Connection. The application consists of 2 main modules:

- 1) Generation of HEC-HMS output files
- 2) Visualization of generated output files



1) Generation of HEC-HMS output files

We considered 3 parameters: Snyder lag, Curve number, Recession constant. For each parameter we considered a fixed range [min,Max] and took N=200 realization by using a step s=(Max-min)/200

a) parameters

STEPS	Modify input	execute simulation	parse output
REQUIRE	modify *.basin file	run HEC-HMS	Read Run.dss file and write data to JSON files
IMPLEMENTED BY	Java code to modify txt file	Java process to run compute.script which executes HEC-HMS	Java process to run script ReadSironFlow.py

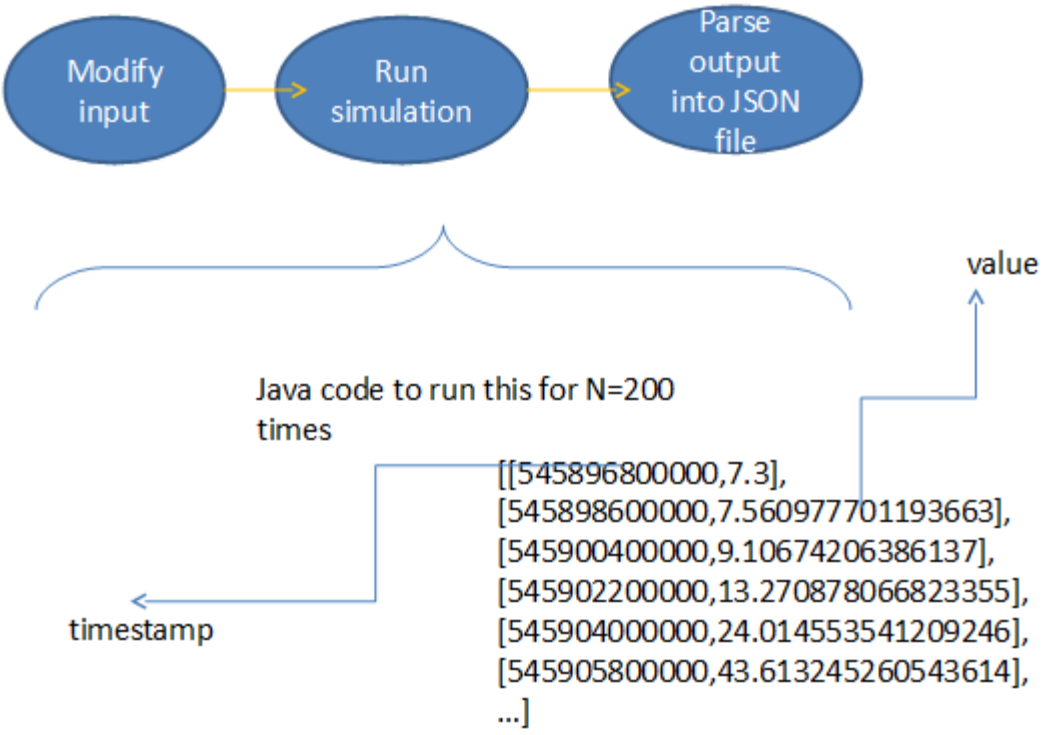
Basically we had to modify an input file (.basin file which is a txt file), then run HEC-HMS and finally parse the output Run.dss file using a Jython script (ReadSironFlow.py)

For precipitation we used 20 realizations, starting with a default value and allowing end-user to modify this value in range 50%-150% from the default value. The approach is similar to the above approach for parameters, only thing that changes is how we updated the input file (in this case is Run.dss) with the new values for precipitation. We have used a Jython script called ModifySiron.py

b) precipitation

STEPS	Modify input	execute simulation	parse output
REQUIRE	Update Run.dss with the new values for precipitation	run HEC-HMS	Read result.dss file and write data to JSON files
IMPLEMENTED BY	Java code to run script ModifySiron.py	Java process to run compute.script	Java process to run script ReadSironFlow.py

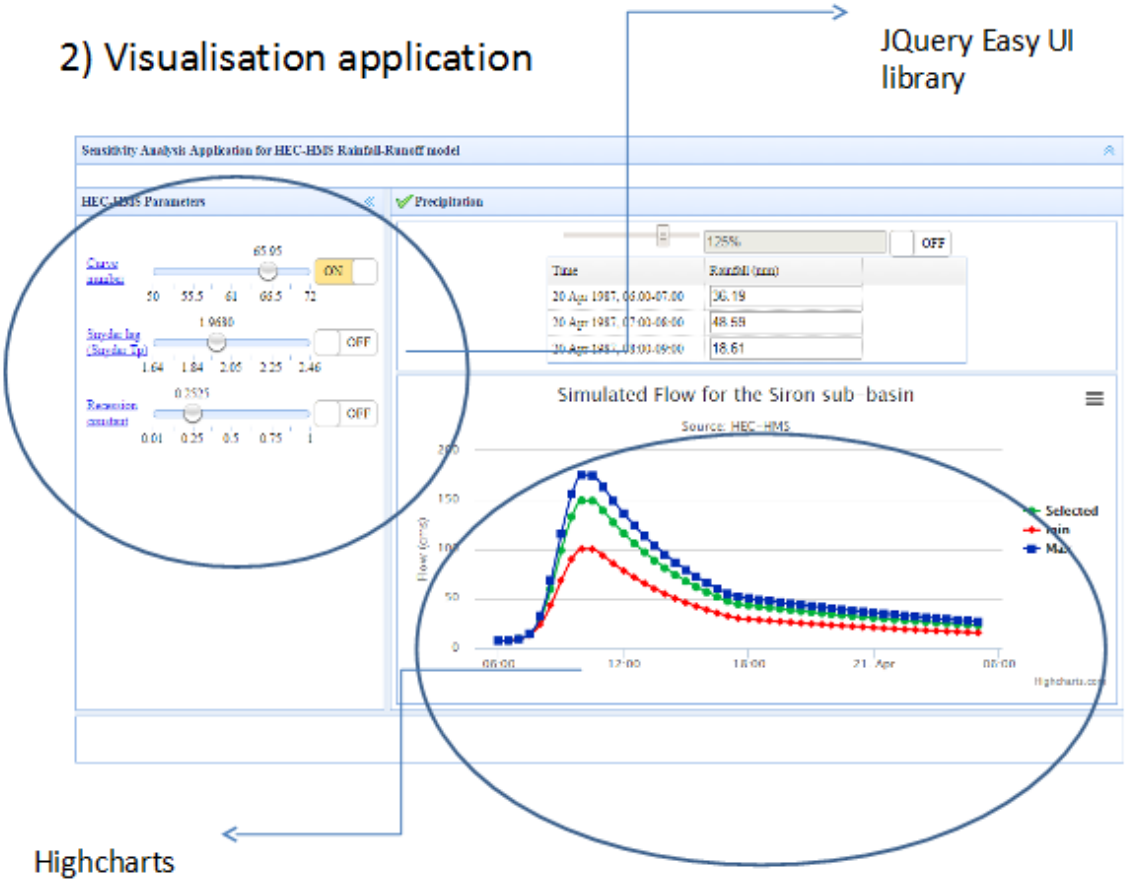
The output (which is basically a time series for the Flow) is parsed into a JSON file as described in the Figure bellow, obtaining an array where each element is a pair [timestamp, value]



Timestamp is a value expressed in milliseconds from epoch. The second value is a float value with high precision expressing Flow [cms].

2) Visualization application

For the panel representing the charts we used Highcharts library [1] which is based on Javascript and JQuery. It offers features such as zoom in / zoom out, export to pdf, png, printing, etc.



For the UI elements such as sliders, buttons, edit boxes, etc we used JQuery Easy UI library [2] which offer AJAX functionality for nice user experience (the page does not have to be reloaded in browser in order to see the changes, these changes are shown immediately).

3) Implementation details

The project was developed in Eclipse [3] development studio. We have used Apache Tomcat [4] to host the resulting .war file.

Servlets	Controller
JSP	View
Data in JSON files	Model

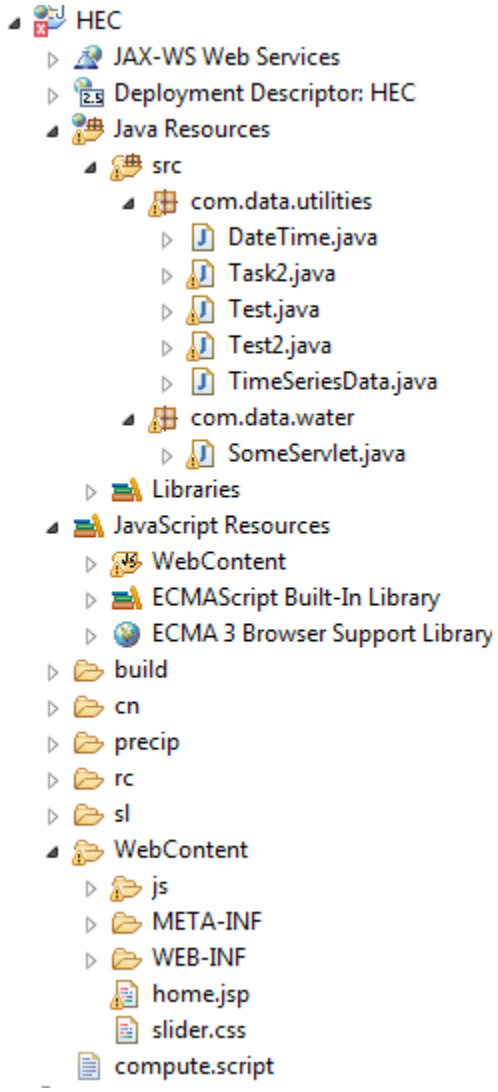
Server to host the web application

Apache Tomcat

Development Environment (IDE)

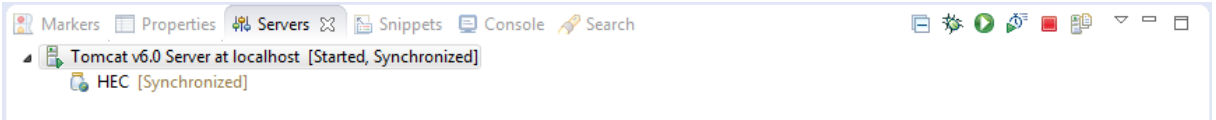
eclipse

The project is built using the MVC (Model-View-Controller) architecture.



The project is named HEC and has the structure presented in the left image. The servlet is placed in package com.data.water (that is the controller of the application). In package com.data.utilities there is a class named Task2.java that has a few functions used for updating parameters, running simulations, etc. The Test.java class is responsible for testing updating parameters and running simulations, and most important for generating the output files (txt files) for Snyder lag, Curve number and Recession constant realizations (genereazaFisiereXX() where XX is the acronym of the parameter). The output files are placed in 4 folders in the project (cn,rc,sl,precip) and are used further by the servlet and JSP for visualization. So the model is represented by all these output files (620 files totally) stored locally in folders cn,rc,sl,precip. The view is represented by home.jsp

The project can be run in the development mode, from Eclipse, form Servers perspective, Run.



REFERENCES

[1] Highcharts library, http://www.highcharts.com/
[2] Easy UI library, http://www.jeasyui.com/
[3] Eclipse, https://eclipse.org/home/index.php
[4] Apache Tomcat, http://tomcat.apache.org/