# GIS-Publisher: From a Geographic Data Set to a Deployed Product with One Command

David de Castro*
Universidade da Coruña
A Coruña, Spain
david.decastro@udc.es

Alejandro Cortiñas
Universidade da Coruña
A Coruña, Spain
alejandro.cortinas@udc.es

Victor Lamas
Universidade da Coruña
A Coruña, Spain
victor.lamas@udc.es

Miguel R. Luaces
Universidade da Coruña
A Coruña, Spain
miguel.luaces@udc.es

## ABSTRACT

In our research laboratory, we have been working on developing a software product line (SPL) specifically tailored for generating web-based geographic information systems (GIS). In addition, we have also designed a domain specific language (DSL) to make configuring our products as easy and flexible as possible. Over time, we have utilized this product line to create small GIS products, aiming to simplify the process of publishing and sharing geographic data. The steps involved in generating and deploying this kind of products are consistently repeated, so they can be easily automated. Doing so, we further reduce the time to market for this set of simple products, and minimize the complexity associated with the entire process.

This article introduces GIS-Publisher, a tool that allows users to easily generate web applications from a directory containing a collection of shapefiles (a popular format for storing geographic data). These web applications can be also automatically deployed on their preferred machine, whether it is locally, remotely (via SSH), or even on an AWS instance. Moreover, the tool also supports the definition of custom styles for each shapefile, granting users full control over the visual representation of their geographic data.

## CCS CONCEPTS

• **Software and its engineering** → **Software product lines**; • **Information systems** → *Geographic information systems.*

## KEYWORDS

software product lines, tool, automatic deployment, geograhpic data

**ACM Reference Format:**
David de Castro, Alejandro Cortiñas, Victor Lamas, and Miguel R. Luaces. 2023. GIS-Publisher: From a Geographic Data Set to a Deployed Product with One Command. In *27th ACM International Systems and Software Product Line Conference - Volume B (SPLC '23), August 28-September 1, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3579028.3609009

*All authors have contributed equally and the names are listed in alphabetical order.

## 1 INTRODUCTION

In recent years, the development of Geographic Information Systems (GIS) has been booming. Nowadays, everyone uses GIS in their day-to-day with applications such as Google Maps to calculate routes between two points, or transport company webpages where we can check the location of the delivery driver who carries our last purchase. GIS are also very valuable to data scientists or, in general, to any scientist who handles data with a geographical dimension.

There are many examples of this type of application, such as the visualization of the evolution of the COVID-19 pandemic in the world[1] or the visualization of the evolution of the climate in the world[2]. These applications offer visualization, analysis, and interpretation of geographic data, allowing users to interact with this spatial data.

In previous publications, steps were already taken to facilitate the generation of this type of web applications by applying software product line engineering [3, 4], configuring the products by selecting variability through a feature model, and specifying the data model for each product. Later, we saw that the flexibility offered by our SPL was not enough, and we define a domain-specific language (DSL) to configure the products [1], also achieving a remarkable simplification of the generation of this type of systems. Since then, we have identified an unexpected use case for our product line: the rapid creation of simple web applications to publish and share geographic data.

Accessing geographic data is now easier than ever, thanks to Open Data initiatives of many governments and public institutions such as INSPIRE[3]. However, most alternatives to share this information within a website with a map viewer are subscription services, such as ArcGIS[4] or Carto[5], with limited features and in which the users cannot easily add new functionality in case they need it. When we are working on the requirements of a new project with

[1]https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6
[2]https://www.climate.gov/maps-data
[3]https://inspire.ec.europa.eu/
[4]https://www.esri.com/en-us/arcgis/products/arcgis-online/overview
[5]https://carto.com/

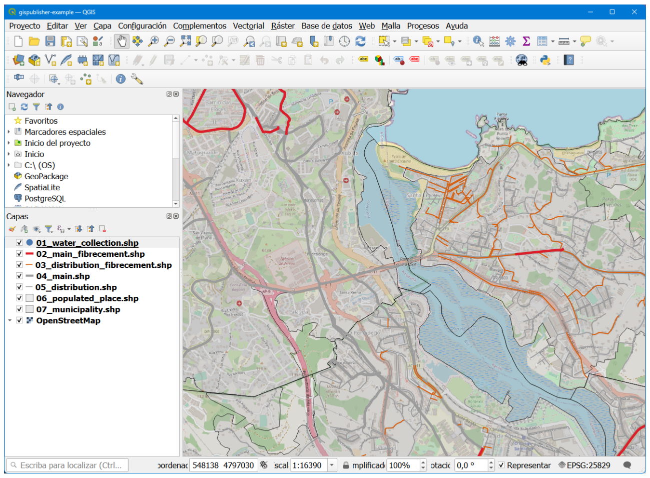

**Figure 1: QGIS with sample shapefiles**

a potential client, or working in collaboration with other research groups, we usually need to show them some example data. We have been using our product line to generate small "use and trash" GIS applications to do this, and we realize that we were just using the DSL to define the model of the geographic data we need to use. Geographic data is usually handled in shapefiles, which is a file format that has become a *de facto* standard. Shapefiles are a set of files that store the geometry of the geographic data and the attributes associated with it. So we load the shapefile in a GIS tool, inspect the data model, and define the model in our DSL. Then, we generate the product and deploy it on a server. These steps are purely mechanical, so we implemented a tool that allows us to do this automatically, just by selecting the folder with the shapefiles.

In this work, we present a tool that automatically generates a web-based small geographic information system from a set of shapefiles. This application has a map viewer, where all the geographic data of the imported files are available, as well as a series of generic lists to see the details of each geographic element, etc. In addition, in the case of having an available server (for example, a dedicated Linux server accessed through SSH, or an AWS instance), it is possible to deploy this product automatically after its generation, which facilitates publishing and sharing the data in a very simple way.

## 2 GIS-PUBLISHER

This section presents GIS-Publisher, a tool built on top of an existing software product line for geographic information systems [3, 4] to facilitate the generation and deployment of products straight from a data set, taken from a local file-system folder. The tool generates a web-based GIS and publishes the generated product on the internet. This section reviews the tool's requirements and specifications.

In addition, the tool's features will be discussed, and its implementation's underlying technology will be covered in detail.

### 2.1 Requirements

The tool must be able to retrieve the data model from a set of shapefiles, which is the most common geospatial vector data format used in GIS software. It is a file format that contains both geometry (such as points, lines, and polygons) and attribute data (such as associated information about the features). Shapefiles have been the most widely used for storing, analyzing, and displaying spatial data for long, even when they have serious drawbacks[6]. Shapefiles can be opened with several tools, being QGIS[7] the most common one (among the Open Source alternatives). QGIS is a desktop application that can load geographic data from many different sources. In Figure 1, we show a set of shapefiles loaded into QGIS, with a base layer from OSM[8]. Each shapefile is represented as a layer with a specific style.

Since the visualization order of the layers within a map viewer is highly relevant, the name of the shapefiles used in GIS-Publisher can be prefixed with a numeric string that determines this order.

An important aspect of the geographic data is the visualization style, usually specified with the styled layer descriptor (SLD) language, an OGC standard[9]. An SLD XML defines the visual appearance of geospatial data layers, such as points, lines, and polygons, including their colors, symbols, transparency, and labeling. Aiming to keep the tool simple, if we want to apply some specific SLD to a shapefile, we can directly put the SLD file in the same folder with the same file name, and with the extension *sld*.

The tool must be able to generate a web-based GIS product using the definition of the data model retrieved from the shapefiles. A default selection of features is automatically selected for the product. For example, the product must contain a map viewer with standard tools, such as panning, zoom, layer manager, measures over the map, etc. This default configuration will be enough for most cases, but the tool shall also export the final configuration for the product, compliant with the SPL used behind the scenes. In this way, a more advanced user could adapt the product if required.

The tool must be able to automatically deploy the generated product, using containers (Docker), in different environments, depending on the configuration provided. In particular, in this first version, we consider deployment to a local machine, deployment via SSH to a remote machine, and deployment to Amazon Web Services (AWS):

(1) **Local deployment** lets the user deploy the code in the machine locally, so it can access via localhost.
(2) **SSH** deploys the final application in a remote machine that has enabled the SSH protocol. The tool copies the generated code into the machine, configures Docker and Docker Compose automatically, and executes the Docker containers with the final product code.
(3) **AWS** deployment creates an AWS instance with a public IP so the user can share the deployed product, the tool takes care of both creating the AWS instance and configuring and installing the necessary programs to finally deploy the product using docker.

Finally, since the main goal is to publish the geographic data, GIS-Publisher is also responsible for importing the shapefiles and their SLD into the application database after completing the deployment operations. As a result, users have a finished application that shows

[6]http://switchfromshapefile.org/
[7]https://www.qgis.org/
[8]https://www.openstreetmap.org/
[9]https://www.ogc.org/standard/sld/

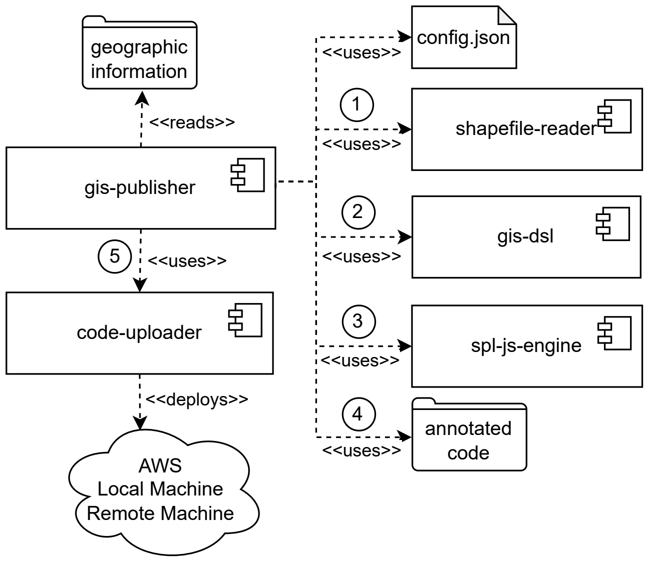

**Figure 2: Architecture component diagram overview**

all the geographic information files in a map with layers and styles that have previously been defined.

Using the tool should be as simple as possible. The simplest case should be just running the tool indicating the path to the folder where a set of shapefiles, containing the geographic information to share, are located. One of the problems of the shapefile format is that each shapefile is composed of several files itself: a main file (.shp) that contains the geometry of the elements, an index file (.shx) that speeds up spatial searches, and an attribute file (.dbf) that stores the information associated with the elements. It is common for shapefiles to be compressed in a zip file. The tool must support both scenarios: compressed or uncompressed shapefiles.

## 2.2 Workflow

This tool is designed to be easily used from the terminal command line. With the command `gispublisher route/to/the/files`, indicating as the first argument the directory where the shapefiles are located, the tool automatically generates and deploys the generated product. The tool has been designed to be as flexible as possible, encapsulating in single-responsible components every small feature required. These components can be seen in Figure 2, and are described next:

(1) *shapefile-reader*. This component processes the shapefiles inside a folder, extracting the data schema and geometry types of each of the files. As mentioned above, the current configuration for the products of the web-based GIS SPL is done using a DSL [1]. Therefore, with the information from the shapefiles data model, *gis-publisher* builds the DSL definition of the required product.
(2) *gis-dsl*. This library parses the DSL definition of a product and converts it into a JSON file, which is later used as input for the derivation engine of the product line. This JSON file includes the data model definition extracted from the shapefiles, the definition of a map viewer which includes the layers with the data from the shapefiles, and, as mentioned above, a set of default features for the generated product.
(3) *spl-js-engine*. This is a JavaScript derivation engine [2], used in the web-based GIS SPL. The JSON produced by the *gis-dsl* is used as input to this engine, which generates the final product.
(4) *annotated-code*. The derivation engine needs to access the annotated code of the components of the product line in order to generate the final product.
(5) *code-uploader*. This last component deploys the product resulting from the *spl-js-engine* derivation on a local or remote machine, or on an AWS instance, performing pre-machine setup tasks if necessary, and running the product via Docker. After the deployment, *gis-publisher* is also in charge of submitting the shapefile data to the final generated product. This can be easily done since the generated products include a feature for the importation of shapefiles.

GIS-Publisher uses a configuration file that defines the deployment parameters, including the type of deployment and the specific parameters required for each of these types. The tool supports automatic deployment for three different alternatives: local deployment, that is, the generated product is deployed in the same machine the tool runs; Linux-based ssh deployment, that is, the generated product is deployed in a Linux server, using an ssh connection to interact with it, and the web server is exposed through the specific port defined in the configuration; and AWS deployment, that is, the specific steps to configure an AWS instance to run the product and expose the web through an AWS URL. An example of a configuration file for an SSH instance can be seen in Figure 3. The user can override the configuration file when running the tool, but setting a default configuration for the tool should be enough on most occasions.

```
{
  "deploy": {
    "type": "ssh",
    "host": "ip or domain",
    "port": 22,
    "username": "ci",
    "certRoute": "/Users/acortinas/.ssh/id_ed25519",
    "remoteRepoPath": "/home/ci/code"
  },
  "host": "https://public-domain.lbd.org.es"
}
```

**Figure 3: config.json**

Running the tool is as simple as running the script in Figure 4, supposing we have a set of shapefiles in the folder `examples/WaterSuply`. After the process, which can take a long time if it is the first run in a new environment, the product should be automatically deployed and accessible. In Figure 5, we show the product built and deployed from the shapefiles in Figure 1, using also the same styles extracted as SLD.

The execution of the tool and the demonstration of its operation can be seen in the following video: https://zenodo.org/record/8116611.

```
gispublisher examples/WaterSupply
```

**Figure 4: Running the cli tool**

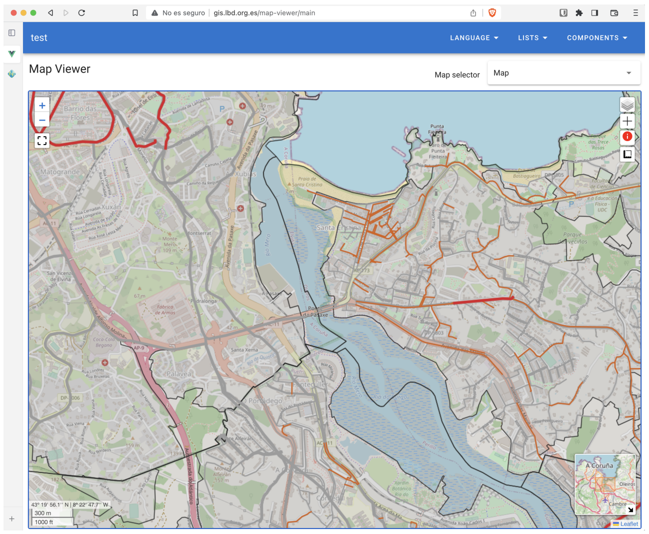

**Figure 5: Deployed product**

## 2.3 Technology

As we have seen throughout this paper, the tool is modularized and each of the modules is responsible for a specific function. The tool itself and all the components are implemented in JavaScript, using NodeJS[10] and dependencies from npm[11].

In order to generate the products, the tool makes use of a derivation engine called *spl-js-engine* [2]. This is a JavaScript library that, following the annotative approach, can generate the source code of the final product from the annotated code, the feature model of the product line, and a product specification.

The grammar for the DSL and the parser that reads it (the component named *gis-dsl*) are implemented with ANTLR[12] [1].

Finally, the **code-uploader** is a Javascript library that enables users to publish their generated code folder to the internet. It currently supports deployments in Amazon Web Services (AWS) instances, and the library is in charge of setting up the instance, obtaining the IP, uploading the code, and, if necessary, configuring the instance and launching the application. In addition, it currently has methods for doing so on Debian machines that use SSH.

## 3 CONCLUSIONS

In this paper, we have presented a tool that allows a user with no web development knowledge to generate a web-based GIS from a set of files containing geographic information, more specifically, shapefiles. In addition to generating an application that contains a map viewer with the selected geographic data, it also allows deploying this application automatically on the machine that the user selects, whether it is a Debian machine, its local machine, or on an AWS machine.

In future work, we want to be able to extend the customization of these products. Thanks to the use of the DSL as a product specification definition model, a user with little knowledge and understanding of the DSL syntax could make changes in this generated DSL to make the generated product better adapted to his needs: eliminating attributes of certain entities, creating layers and maps through the DSL, etc. In addition, it would be of interest to extend the formats of the files containing the geographic data and to support others such as GeoJSONs, as well as to extend the services in which it is possible to deploy the product, such as Microsoft Azure.

[10]https://nodejs.org/
[11]https://www.npmjs.com
[12]https://www.antlr.org

## ACKNOWLEDGMENTS

Partly funded by: MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, FLATCITY-POC: PDC2021-121239-C31; MCIN/AEI/10.13039/501100011033 EXTRACompact: PID2020-114635RB-I00; Xunta de Galicia/FEDER-UE GRC: ED431C 2 021/53; TED2021-129-245B-C21 (PLAGEMIS): partially funded by MCIN/AEI/10.13039/5-01100011033 and "NextGenerationEU"/PRTR; MCIN/AEI/10.13039/501100011033 MAGIST: PID2019-105221RB-C41; CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Department of Culture, Education, Vocational Training and Universities and the Galician universities for the reinforcement of the research centers of the Galician University System (CIGUS).

## REFERENCES

[1] Suilen H. Alvarado, Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira, and Ángeles S. Places. 2020. Developing web-based geographic information systems with a DSL: proposal and case study. (2020), 167–194. https://doi.org/10.13052/jwe1540-9589.1923

[2] Alejandro Cortiñas, Miguel R. Luaces, and Oscar Pedreira. 2022. spl-js-engine: a JavaScript tool to implement Software Product Lines. In *International Systems and Software Product Line Conference (SPLC)*. ACM, 66–69. https://doi.org/10.1145/3503229.3547035

[3] Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira, and Ángeles S. Places. 2017. Scaffolding and in-browser generation of web-based GIS applications in a SPL tool. In *International Systems and Software Product Line Conference (SPLC)*. ACM, 46–49. https://doi.org/10.1145/3109729.3109759

[4] Alejandro Cortiñas, Miguel R. Luaces, Oscar Pedreira, Ángeles S. Places, and Jennifer Pérez. 2017. Web-Based Geographic Information Systems SPLE: Domain Analysis and Experience Report. In *International Systems and Software Product Line Conference (SPLC)*. ACM, 190–194. https://doi.org/10.1145/3106195.3106222

## A DEMO

The presentation of the tool during the conference will encompass the following key points:

(1) The presentation will start with a very brief explanation of what a GIS application is and the difficulty of developing one.

(2) Then GIS-Publisher will be introduced, the functionalities it incorporates, and the problems it solves.

(3) Once the tool has been introduced, the slides will be left to focus on the demo, downloading the tool, and showing the files that will be used as examples.

(4) Once the objective we want to achieve has been defined, the tool will be executed in front of the audience and each of the steps it performs will be briefly explained, although the execution of the tool will be stopped after a short time, since the last steps take more time than the time available for the presentation.

(5) With the result of a previous execution of the tool before the presentation, the code generated by the tool from the files given by the user and how the result could be executed locally will be shown.

(6) After the presentation of the source code, the final result, a geographic information system automatically generated from a series of shapefiles only, will be shown on the screen.

(7) Once the result has been presented, we will return to the slides to discuss some conclusions and future work.

The demonstration of the tool, similar to the one to be performed at the congress, is available in the video shared in https://zenodo.org/record/8116611.

In the demonstration booths, it will be possible to show the execution of the tool in a more detailed way according to the interests of the attending audience. Some points are listed below:

- Installation of the tool on the local machine, using npm.
- Execution of the examples from a terminal. Through the command line, users will be shown how to run the tool through a previously prepared example, as well as the different options offered by the tool, such as generating a product without deploying it to a machine, etc.
- Demonstration of the intermediate results of the execution of the tool, such as the intermediate Domain Specific Language that is generated to subsequently derive the product.
- In-depth demonstration of the functionalities of the generated product, such as layer sorting, style modification or map export.