Put your models in the web

Less painful

Nils Brinckmann (GFZ)
nils.brinckmann@gfz-potsdam.de

Massimiliano Pittore (Eurac Research)
massimiliano.pittore@eurac.edu

Matthias Rüster (GFZ)
matthias.ruester@gfz-potsdam.de

Benjamin Proß (52° North)
b.pross@52north.org

Juan Camilo Gomez-Zapata (GFZ)
juan.camilo.gomez.zapata@gfz-potsdam.de

EGU Sharing Geoscience Online 2020

Session ESSI2.19
Management and integration of environmental observation data
Version: Work in progress

Context

Why should we share our models in the web?

Context

Complexity

Context

Very challenging for single domain experts to handle all aspects

Context

Need for modular, distributed frameworks

Context

To share expertise and work together

Context

Our context: RIESGOS project

Context

https://www.riesgos.de/en/

Context

Multi-hazard risk assessment

Context

Part of it:

Lahars (TU Munich)

Tsunamis (Alfred Wegener Institute)

…

Earth quakes & floods (GFZ)

Context

Each institute in their domain

Context

So work together

Context

And make it easy to exchange

Context

Using standards

WPS

WPS

WPS

OGC-Standard

WPS

Allows to constrain and work with various data formats

WPS

Yet flexible to model all possible computations

WPS

And accessible from our GIS

WPS

Sometime still smells like a dead horse

WPS

And not that easy to integrate for all common used scientific programming languages

WPS

Java?

WPS

52°North implementation

https://github.com/52North/WPS

WPS

But, Java is not so common in science

WPS

Python?

WPS

PyWPS / Geoserver via WSGI

https://github.com/geopython/pywps

https://docs.geoserver.org/latest/en/user/community/scripting/py/index.html

WPS

C?

WPS

Zoo implementation

http://zoo-project.org/site/

WPS

R? Fortran? Shell scripts? Octave? ...

WPS

Nothing so far

WPS

But we can call that from Java / Python

Wrapper

We just need some wrapper

Wrapper

And then we need them again for the next service

Wrapper

And again

Wrapper

And again…

Wrapper

So lets build a wrapper framework

Wrapper

On top of the server and the work from 52°North

https://github.com/bpross-52n/quakeledger

Wrapper

To integrate all kind of command line programs

Wrapper

Allowing input as cmd parameters

Wrapper

Or via stdin

Wrapper

And input files

Wrapper

And handling output

Wrapper

From output files

Wrapper

And stdout

Wrapper

And sometimes even from stderr

Wrapper

Able for proper error handling

Wrapper

So we have wrapper functionality

Wrapper

Now lets write code for each process we will integrate

Wrapper

Using the wrapper functions over and over again

Wrapper

...

Wrapper

:-(

Wrapper

We are lazy

Wrapper

We want to have one base process

Customization

That is customizable

Customization

Without writing code

Customization

```
{
   "title": "QuakeledgerProcess",
   "workingDirectory": "/usr/share/git/quakeledger",
   "commandToExecute": "python3 eventquery.py",
   "exitValueHandler": "logging",
   "stderrHandler": "pythonTraceback",
   "input": [
      {
         "title": "mmin",
         "abstract": "minimum magnitude",
         "useAs": "commandLineArgument",
         "type": "double",
         "default": "6.6"
      },
      // ...
```

Customization

And we can just add services by adding a json file

Encapsulation

And we want to split server code from service code

Encapsulation

Like using Java for server work

Encapsulation

And others for the scientific processes (Python, R, ...)

Encapsulation

And we want to capsulate the code of the services from each other

Encapsulation

Why?

Encapsulation

Different languages

Encapsulation

Different dependencies

Encapsulation

Different handling of temporary files

Encapsulation

So use containers to split

Encapsulation

Docker

Encapsulation

```
{
    "title": "QuakeledgerProcess",
    "abstract": "This is the description of the quakeledger process.",
    "imageId": "quakeledger:latest",
    "workingDirectory": "/usr/share/git/quakeledger",
    "commandToExecute": "python3 eventquery.py",
    // ...
```

Encapsulation

Handles dependencies

Encapsulation

Handles isolation of different runs (even with temporary files)

Encapsulation

Handles integrity of the service itself

Benefits

So having one base process that is configurable gives us benefits

Benefits

Like caching

Benefits

For all integrated services, regardless which language

Benefits

That‘s what we build

Our code

https://github.com/gfzriesgos/gfz-command-line-tool-repository

Our code

Used by several partners in RIESGOS

How to use it

So what do you have to do to use it?

How to use it

Start with a server

How to use it

You can even use our docker image:

https://hub.docker.com/r/gfzriesgos/riesgos-wps

How to use it

Take your command line program

How to use it

Write a dockerfile

How to use it

Write a json configuration

How to use it

And put your model in the web

How to use it

There is even some documentation

https://github.com/gfzriesgos/gfz-command-line-tool-repository/tree/master/doc

How to use it

We did ~~all~~ some of the painful stuff for you

How to use it

So check it out

Future

And give us your feedback

Future

Help us to get that into broader use

Future

And to go on with development

Future

Like:

Future

A web site to register processes

Future

Or keeping track of identifiers of input parameters and output parameters that will allow us to refer to the exact WPS output and all of the processing later

Future

Or testing singularity instead of docker

Future

Or whatever ideas you may have

Future

To push the WPS

The end

Thank you