CS461 SENIOR CAPSTONE

NOVEMBER 21 - FALL 2017

COVERAGEJSON RESPONSE HANDLER FOR OPENDAP

TECHNOLOGY REVIEW

PREPARED BY

COREY HEMPHILL

GROUP #55

Abstract

This document compares and contrasts several potential technology choices that can be used for the CoverageJSON Response Handler for OPeNDAP project. The three technologies discussed here are data access protocols, scientific data formats, and UI data models. For data access protocols, File Transfer Protocol (FTP), OpenGIS Consortium (OGC) Web Coverage Service (WCS), and Open-source Project for a Network Data Access Protocol (OPeNDAP) are considered. For scientific data formats, Extensible Markup Language (XML), Climate Science Modeling Language (CSML), and CoverageJSON (CovJSON) are considered. And lastly, for UI models, domain, application, and task models are considered.

1 Technology: Data Access Protocols

1.1 Overview

The NASA Jet Propulsion Laboratory (JPL) podaac website allows scientific data access via several different servers and data access protocols. This project requires the use of a data access mechanism to access scientific data from NASA JPL servers and databases. We will be modifying this method to implement a data response handler for a new scientific data format. This section will discuss methods of data access, and will consider their attributes against the established criteria.

1.2 Criteria

1) Machine-to-machine interoperability
2) Ability to work with heterogeneous data sets
3) Is an open-source software

1.3 Potential Choices

1.3.1 File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is the most widely used data access protocol in existence, and is often used in tandem with secure shell (SSH) connections. FTP facilitates the transfer of files between a client and a server on a given network, and does so using separate connections for both data and control connections. The specification for FTP was written back in the early 70's, and there is a long list of FTP clients that are freely available for use.

1) In terms of machine-to-machine interoperability, FTP is system agnostic, and can even handle file transfers between different operating systems. Generally speaking, if you need a file transferred, it can almost certainly be done via FTP.
2) FTP can handle almost any file/data type you throw at it: image files (.jpg, .gif, etc.), executables (.exe), documents (.xlsx, .docx, .pdf, etc.), archive files (.zip, .gz, etc.), and more.
3) FTP isnt an open-source software, per se, however, there are a number of open-source software which provide FTP solutions. Some examples would be FileZilla, WinSCP, and Cyberduck.

1.3.2 OpenGIS Consortium (OGC) Web Coverage Service (WCS)

OpenGIS Consortium (OGC) Web Coverage Service (WCS) is a protocol for accessing multi-dimensional coverage data over the internet. WCS is especially useful for handling gridded data sets, and other coordinate based data sets. In order to use WCS, coordinate based data must have complete information.

1) Since OGC WCS is a protocol based on HTTP, it is very reliable between machines as it is a network based protocol that functions in a request-response format.
2) OGC WCS is rather limited in the data types it can handle. In comparison to FTP which can handle almost every data type except for scientific data formats, WCS only handles scientific data formats; more specifically, WCS likes gridded data sets.
3) Unfortunately, OGC WCS is not an open-source project that can be added to. The OGC has a GitHub site which primarily deals with test suites for testing WCS functionality, but does not currently provide its source code openly.

1.3.3 Open-source Project for a Network Data Access Protocol (OPeNDAP)

OPeNDAP is an open-source network data access protocol developed by engineers at NASA JPL. The DAP2 protocol provides scientists and developers a discipline-neutral means of requesting and receiving data over the internet. OPeNDAP already contains request handlers for NetCDF, HDF, FreeForm, NcML scientific data formats.

1) OPeNDAP aims to be a core component of specialized systems that provide machine-to-machine interoperability.
2) OPeNDAP can handle a wide variety of scientific data types such as NetCDF, HDF, and much more.
3) OPeNDAP is a completely open-source project that is presented freely to the public. The purpose of the project is to provide an avenue for sharing data access with users of all disciplines, as well as a variety of services.

1.4 Discussion

Although FTP is a reliable and widely used protocol, it is not a good candidate for integrating a response handler. FTP has good machine-to-machine interoperability, handles a wide variety of data types, but is not really an open-source project which can be contributed to. OGC WCS also has good machine-to-machine interoperability due to being based off of HTTP, but is fairly limited in that it really only handles a small number of data types, primarily gridded coordinate data sets, which is good for coverage data. However, OGC WCS is not currently an open-source project, which makes it a poor project candidate. Lastly, OPeNDAP is committed to being a primary component in making machine-to-machine interoperability easier, handles a wide variety of scientific data types like NetCDF, HDF, and FreeForm, and is a completely open-source project.

1.5 Conclusion

OPeNDAP is the best candidate for data access of the three choices. It is a perfect selection for adding a new scientific data response handler.

2 Technology: Scientific Data Formats

2.1 Overview

The NASA JPL podaac website provides a number of scientific data formats for scientists and developers to utilize in their research and development. These formats include NetCDF, HDF, MD5, and more. This project requires a scientific data format that deals almost exclusively with satellite coverage data. We will be implementing a response handler for this scientific data format, and we will be integrating this handler with a data access method from the previous section. This section will discuss different data formats, and will consider their attributes against the established criteria.

2.2 Criteria

1) Ability to process and handle satellite coverage data
2) Format is simple and easy to use
3) Incorporates metadata

2.3 Potential Choices

2.3.1 Extensible Markup Language (XML)

XML is a common markup language and is used in a wide number of applications. XML is popular due to its ease of use and its readability. In short, XML encodes data into a structured format which includes both metadata and the corresponding data.

1) Although it may be possible to use XML to handle satellite coverage data, it was not necessarily designed for the task. This could be problematic in the future.
2) XML can be used to convey relatively simple formats, and it is pretty easy to learn and use.
3) XML is a metadata based markup language.

XML Example:

```
<PLANT>
    <COMMON>Bloodroot</COMMON>
    <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$2.44</PRICE>
    <AVAILABILITY>031599</AVAILABILITY>
</PLANT>
```

2.3.2 Climate Science Modeling Language (CSML)

CSML is a modeling language that specializes in handling both atmospheric and oceanographic data sets, which can be very similar to coverage data sets. Data types in CSML are generally defined primarily based on geometric and topologic data structures which describe discrete locations in space and time.

1) CSML is not explicitly described as a language for handling coverage data sets, however, it could likely be used for the task as atmospheric and oceanographic data sets can be described with similar data.
2) CSML is no more or less easy to use than XML, however, features in CSML tend to be a little more complicated in terms of structure.
3) CSML is a metadata based modeling/markup language.

CSML Example:

```
<Measurement>
    <Instrument>RADISON DE</Instrument>
    <measuredParameter>TEMPERATURE</measuredParameter>
</Measurement>
<SondeProfile>
    <measuredParameter>TEMPERATURE</measuredParameter>
</SondeProfile>
```

2.3.3 CoverageJSON (CovJSON)

CovJSON is a markup language format that encodes geospatial coverage data types such as grids, time series, and vertical profiles, and is based on JavaScript Object Notation language (JSON). CovJSON is object oriented, and can contain a combination of domains, parameters, ranges, and metadata as object members.

1) CovJSON was specifically designed to process and handle a number of different coverage data types.
2) Although there is some inherent complexity in representing coverage date, CovJSON is relatively simple to read and use.
3) CovJSON is a metadata based markup language.

CovJSON Example:

```
{
  "type" : "Coverage",
  "domain" : {
    "type": "Domain",
    "domainType": "Grid",
    "axes": {
      "x": { "start": -179.5, "stop": 179.5, "num": 360 },
      "y": { "start": -89.5, "stop": 89.5, "num": 180 },
      "t": { "values": ["2013-01-13T00:00:00Z"] }
    },
    "referencing": [{
      "coordinates": ["x","y"],
      "system": {
        "type": "GeographicCRS",
        "id": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
      }
    }, {
      "coordinates": ["t"],
      "system": {
        "type": "TemporalRS",
        "calendar": "Gregorian"
      }
    }]
  },
  // Some lines omitted due to length
}
```

2.4 Discussion

XML is a very simple to use metadata based markup language that caters to a wide variety of data formats, however, its downfall is that it wasnt necessarily designed to be used with coverage data. For the purpose of this project, we require a more specialized language geared towards handling coverage data. CSML is a more specialized metadata based markup language that is very similar to XML in its simplicity and ease of use, but it was also not designed to handle more complex coverage data. Although its a better candidate than XML, CSML still has some shortcomings that could be problematic in the future. Finally, CovJSON is a slightly more complicated, but simple to use, object-oriented markup language specifically for encoding coverage data, and it is metadata based, which makes it the perfect candidate for the project.

2.5 Conclusion

Of the three choices listed, CovJSON is clearly the best candidate for the projects data format.

3 TECHNOLOGY: UI MODELS

3.1 Overview

In order to establish a mechanism for the use of our data access protocol to obtain scientific satellite coverage data, which are concepts discussed in prior sections of this document, a number of possible graphical user interface (GUI) models should be considered for use. This section will discuss different UI models, and will consider their attributes against the established criteria.

3.2 Criteria

1) Defines user interaction, or is usage-centered
2) Is able to represent the commands and data that the application requires
3) Aids in simplifying user interaction

3.3 Potential Choices

3.3.1 Domain Model

The domain model is a model that defines what objects a user sees, and what they can interact with on a given UI. Typically, a domain model also includes a data model that defines what data a user can see as well.

1) A domain model defines user interaction for a given UI in that it identifies the objects in the domain in which a user can interact.
2) A domain model does not necessarily represent the commands available to the user, however, it does often come with a data model which defines what data the app requires/provides.
3) A domain model does not aid in simplifying user interaction.

3.3.2 Application Model

The application model is a model that defines the commands and data that a given application provides to the user.

1) An application model strictly defines user interaction through the definition of the commands a user can use.
2) An application model defines both the commands and data that the application provides.
3) An application model does not inherently aid in simplifying user interaction, but defining a very limited number of commands to the user could make interaction simple enough.

3.3.3 Task Model

The task model is a model that defines the tasks that a user can perform, and determines what interaction capabilities must be designed for the application.

1) The task model defines what capabilities the user should have within the domain of the application.
2) A task model neither explicitly defines the commands the user has access to, nor does it define what data is available to the user.
3) Task models can be very complex and can define entire workflows; they are intended to capture more complex user interactions.

3.4 Discussion

All three of the UI models discussed here could likely suffice as a UI model for this project, but there is one model that fits best; the application model. Since the aim of the project is to enhance the usability of coverage data, the primary concern is keeping interaction simple. The application model strictly outlines user interaction by defining the commands that user has access to. By defining a limited number of commands available to the user, we can aim to simplify user interaction. Also, since the purpose of the project revolves around handling data, our UI model should define the data the user can interact with, which the application model does.

3.5 Conclusion

The application UI model is the candidate that fits best with the projects intent.