Microservice-Based System for Environmental Science Software Applications

Vinh Le, Connor Scully-Allison, Mitchell Martinez
Sergiu M. Dascalu, Frederick C Harris, Jr., Scotty Strachan, Eric Fritzinger

Computer Science and Engineering
University of Nevada, Reno
Reno, Nevada 89557, USA
vle@unr.edu, cscullyallison@email.arizona.edu, mitchell.martinez@nevada.unr.edu
dascalus@cse.unr.edu, fred.harris@cse.unr.edu, strachan@unr.edu,landale3@gmail.com

***Abstract*—When an environmental research project grows, technical concerns over system scalability, data exposure, and third-party application support are overlooked. This paper presents a system, the Microservice-based Envirosensing Support Applications (MESA), that provides a scalable environment and data infrastructure solutions for the NSF-funded Solar Energy-Water-Environment Nexus project. MESA can be broken into 4 major parts: a suite of microservices exposed over an API, an overarching service discovery, a series of tables replicated from an existing monolith, and the applications that MESA lends its support. In order to evaluate the capability of MESA, the features of this system were compared against three other existing microservice-based research systems. MESA features were more robust than two of the other systems, but was found lacking when compared to the last, as it does not lender support to advanced techniques like HPC or Machine Learning.**

***Index Terms*—Microservice, Distributed System, Data Management, Containerization, Data Analysis, Data Visualization, User Interface, Web Framework, Web-based Systems**

I. Introduction

When it comes to environmental research projects, a common approach to data storage is spinning up a monolithic system consisting of one or more databases, interwoven code, and a small suite of sensors streaming in data at specific intervals. However, as data begins to accumulate and more sensors are deployed as the project begins to grow, problems emerge from this ad hoc approach.

Querying speeds, management, access, and analytics all become affected by the increase in data volume. Usually in modern software engineering practices, this implies that it is time for a system reconstruction. Although appropriate, these overhauls exact a heavy toll in time and resources, commodities not readily available for most small-to-midsize environmental research projects.

To address these problems, this paper presents the Microservice-based Envirosensing Support Applications (MESA), a distributed support system build using a Microservice Architecture style that is tailored for the use in environmental research projects. As part of this research, MESA was implemented to support the NSF Track 1 Solar Energy-Water-Environmental Nexus project and it's data hub, the Nevada Research Data Center (NRDC) [4].

We evaluate the comparative merits of MESA against similar systems with a feature comparison. Using common features expected from software like this as a benchmark, MESA is shown to possess more functionality than two of the other systems. However, MESA is slightly deficient when compared to the last system due to the lack of infrastructure for Machine Learning and High Performance Computing. We address these deficiencies by indicating that these are key areas of future work further on.

The remainder of this paper is structured as follows: Section 2 presents the background of the NRDC and the three systems being compared against MESA, Section 3 provides details of the software specifications of MESA, Section 4 describes the various implementations of MESA's microservices, Section 5 provides a discussion on the feature comparison and other considerations, and Section 6 wraps up the paper with the conclusion, and planned future work.

II. Background and Related Works

A. Nevada Research Data Center

The Nevada Research Data Center (NRDC) is the central data hub of the Solar Energy-Water-Environmental Nexus project, where environmental sensor data from various research teams is collected and stored [4]. Unfortunately, the NRDC is the descendant of an older monolithic system and inherited its predecessor's rigidly interconnected approach [3]. Due to its interwoven nature, the NRDC system has great trouble even maintaining itself. As an example, the NRDC could miss several hundred data entries, which is especially disconcerting for the scientists who are expected to conduct research on the data. Furthermore, the NRDC has virtually no means to actively monitor its services' health which makes it hard for to tell if a functionality was even online. Fortunately, this paper is not the first time this problem was recognized, and there was prior work by the authors of this paper on proposing a more distributed reformation of the NRDC [11, 9].

B. Microservice Architecture

The term "Microservice" was traced back to a Microsoft Service Edge Conference presentation in 2005 by Dr. Peter Rodgers. Dr. Rogers referred to the concept of granular web

services that remained independent of each other as "Micro-Web-Services" [13]. These granular web services could then be mapped to a specific functionality and the inter-switching of them created a new option for modularity within the system. The eventual orchestration of these microservices would eventually lead to a functioning system architecture. However, the architecture has a failing in the form of it being incredibly difficult to implement. In order to deploy a microservice-based implementation, it would require the actual construction of such a system by using the concepts as a guideline. Very few software and packages exist out there that would streamline the process of creating a microservice-based system.

C. DIMMER Smart City Platform

Although not environmental in nature, a similarly developed research-oriented microservice-based system was created to support a smart city project in Europe. This system, dubbed DIMMER, collected sensor data, interfaced with a suite of applications, and used a service discovery as part of their DevOps [8]. DIMMER also featured an high performance computing (HPC) resources as part of their platform. However, because DIMMER both actively collects sensor data while also providing support to various applications, this can create significant network overhead for the researchers utilizing the system.

D. Generic Service Infrastructure for PEIS

Sharing many similarities with MESA, a microservice-based environmental research system, the Public Environmental Information System (PEIS), was established as part of a nation-wide environmental research project to provide infrastructure and application support to various client applications, such as sensor networks, web applications, and mobile devices [2]. As part of PEIS' design, the system also provides support to advanced research tools such as HPC and Machine Learning. On top of this, PEIS also features modern tool integration, in the form of containerization, continuous integration, and a service discovery. The approach adopted by PEIS during the implementation of PEIS involved the complete refactoring and rebuilding of a sensor collection system. During an ongoing environmental research project, such decisions could prove to be too expensive in time, money, and data lost.

E. OceanTEA: Exploring Ocean-Derived Climate Data

On a similar research scale, a support system, dubbed OceanTEA, was designed to aid researchers with processing data from a ocean monitoring system designed by the University of Kiel in Germany [7]. This system uses microservices to structurize the data presented to researchers based on specified criteria. OceanTEA then presents this information through an intuitive and responsive web interface. However, the design of OceanTEA gives off the impression that the system is tailored only towards to the structuring and presenting of data to researchers, rather than providing the groundwork for other future tool development.

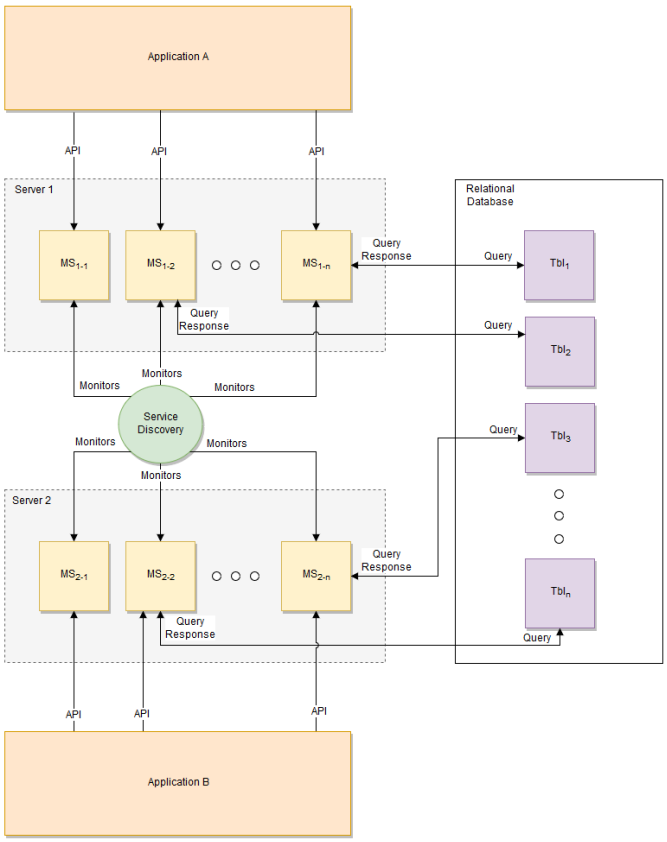

Fig. 1: A high-level view of the MESA system.

III. Software Specification

A. High Level Design

MESA as a system has four major components, as shown in Figure 1. At the center of MESA is the Service Discovery, which serves as a both registry and monitor for all of the microservices. The Service Discovery does not actively entangle itself with any services, aside from scheduled tests, and provides to the user necessary metadata regarding location and health. The next and most crucial portion of MESA are the microservices. The microservices run independent of one another and execute specific programs and tasks for the MESA system. These are often shuffled into servers associated with their specific functionality. Their functionalities are then made available on those servers via reverse-proxy to an application through their respective HTTP APIs. Moving on to the next component, the tables of the database are not only utilized as a general data abstraction between client applications, but are also called and used in certain microservices to perform complex calculations or data management operations. Finally, the last component is the multiple applications that interface with MESA. These applications are not limited only to web applications, but also include mobile phone apps and can even be separate systems.

B. Technology Utilized

The MESA system was developed using several common web technologies compatible with the database manager used

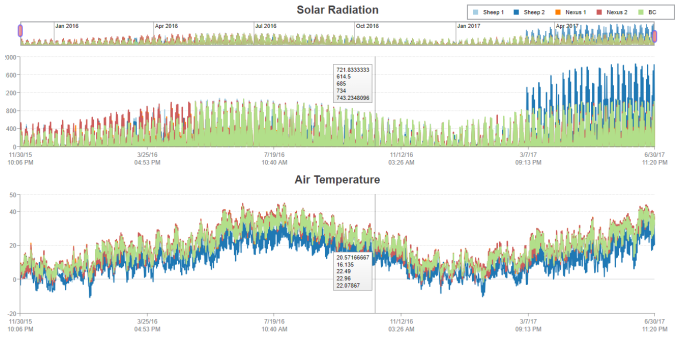

Fig. 2: Sample data visualization on the Lysimeter Data Display.

by the NRDC. The programming languages utilized include C# and Python, and the tools utilized were WCF and Flask. Windows Communication Foundation (WCF) is a toolset developed in the .NET Framework that specializes in implementing and deploying service-oriented architectures (SOA) [10]. Flask is a python-based micro-framework that supports the development of web services [1]. Database management for the NRDC is handled through Microsoft SQL Server (MSSQL). For its service discovery, MESA uses *Consul* [5]. Containerization is handled through Docker and Continuous Integration is managed with Jenkins.

IV. Use Cases

A. SEPHAS Lysimeter Visualization

The SEPHAS Lysimeter Visualization, as shown in Figure 2, is a web application developed to better visualize the environmental data gathered over the span of several years by the SEPHAS facility in Las Vegas [6]. The microservice support from MESA played a non-vital but significant role in the visualization of the lysimeter data. By using powerful frontend visualization libraries such as D3.js, the data file could be loaded and visualized.

However, this would cause almost unbearable lag times between actions issued by the user on the visualization. Although the visualization can operate without the need for a microservice, the usage of a microservice in this case was able to cut down virtually all of the lag time between the user actions and the visualization library. The Data Visualization microservice was utilized to handle all of the data loading and transformation operation, so the web client only needed to query the microservice for all of its needs. Once the client contacts the microservice, the service will then return limited amounts of data to only preserve the shape of the visualization. However, once the user explored further into the visualization, the microservice would then alter the range and the amount of data presented to match what the user viewed. This allowed the client to levy all of its intensive actions onto the server and provide an accurate, responsive, and swift visualization.

B. NRDC Quality Assurance Application

The NRDC Quality Assurance (QA) Application, or QA App for short, is an application developed by the Cyberinfrastructure component of the Nexus project to handle metadata [12]. As part of the project, Nexus technicians often trek out to research sites for maintenance, installation, and configuration of sensor tower equipment. Technicians would have to manually write down entries on a notebook and then transcribe those notes into a database sometime after. The QA App was developed for the express reason of alleviating the troubles faced by Nexus technicians. The application narrows down metadata on research sites specific to the user and allows them to alter entries or add new ones right at the tower. Since there is limited internet access at these towers, this application stores the changes locally and syncs them to the database when appropriate internet connection is made available.

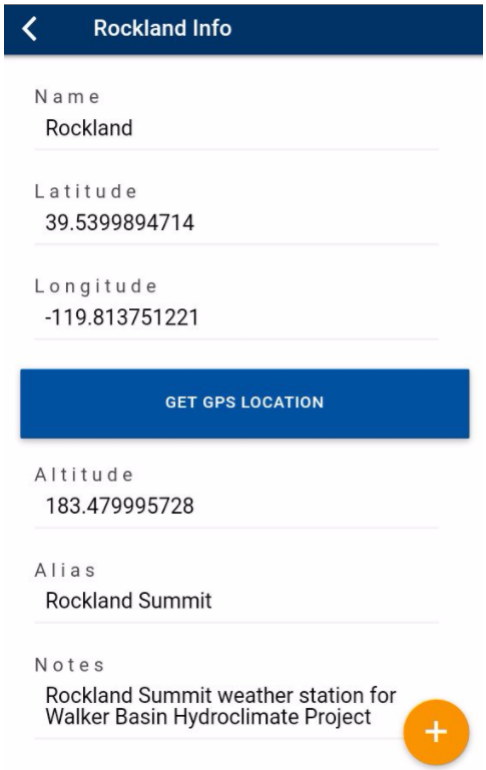

Fig. 3: The NRDC QA Application navigating through a site entry.

The microservices play a vital role as server backend for the QA application. The QA application upon initial activation calls upon each of the eight microservices to store a local copy of the relevant data entries within the metadata database. It is here where the microservices converts data from the NRDC and presents it to the QA application. When changes are made in the application and the sync button is pressed, the application then uses the eight microservices alongside the Conflict Management microservice to verify and submit the changes to the database. Should the Conflict Management microservice return a merge issue, the response from the microservice is parsed and then used to generate a merging interface. Additionally, the Imagery microservice is called when an entry features an image and handles the storage and retrieval of that image into the database. The Imagery microservice is also called when the a entry is viewed by the

user, where it retrieves a preview image instead of the original.

C. Conflict Management

Fig. 4: The conflict management functionality on the NRDC QA Application.

The Conflict Management microservice was created to resolve conflicts that result from multiple users enacting changes on the NRDC database with an application. Conflict Management was developed largely for the metadata application uploading multiple entries at one time. The main process operates in a similar manner as most version control software. When the submission of a data entry whose modification date is earlier than what is listed inside the database, a conflict is flagged. Much like version control software, the user is given the option to continue with their flagged copy or merge their version with the current canon. Once a selection is chosen, the microservices locates the database table in which the data resides and overwrites the entry with the selection made. This is then repeated for each of the multiple entries being uploaded by the application during that one transaction.

Calling the Conflict Management microservice requires sending a POST request consisting of a list of entries to submit to the database. Also inside the JSON, metadata is given to locate the entry's associated database table. The microservice then goes through each of the submissions and compares the modification dates. Should a conflicting modification date be found, the microservice appends that entry to a flagged list. Meanwhile, the passing submissions are added to their respective database tables via the appropriate microservices. At this point, the microservice will return a response detailing specific information about the conflict, and a copy of both what was sent and what currently exists within the database. The response returned provides the necessary information for a front end to create a conflict resolution interface. Once a finalized choice has been made, a POST request to another URI within the microservice allows for the overwriting or updating of what currently exists within that database entry.

D. Near Real-time Autonomous Quality Control

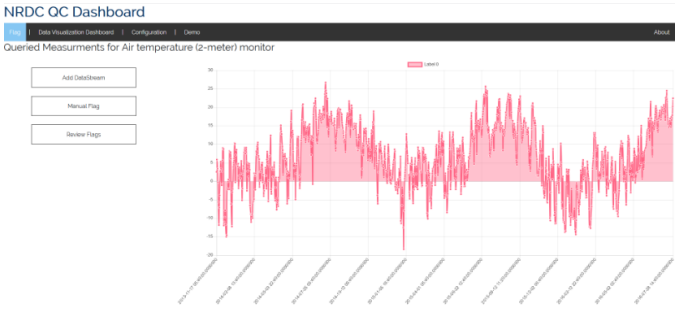

Fig. 5: The main visualization component of the NRAQC system.

Occasionally, sensor readings received by the NRDC from the remote research sites show signs of erroneous data. This can be missing values, values outside possible bounds, or even repeats of past values. To address these problems, the Near Real-time Autonomous Quality Control (NRAQC) System was developed for the NRDC [14]. NRAQC tests incoming data points logged autonomously at a research site to see if they meet the criteria of an invalid measurement. The system, with aid of user specified configurations, flags all invalid measurements with metadata that specifies the nature of the invalidity. The service provided by NRAQC is necessary for the production and distribution of a quality data product. A sample of NRAQC's data visualization is shown in Figure 5.

Much like the QA Application, the microservices play the vital role of server backend for the NRAQC system. NRAQC utilizes an intuitive web interface as the main client, but splits its main features into microservices that support it. These features includes the handling of differing data sources, enable autonomous flagging of measurements, interfacing with the client, enabling a data visualization, and formatting the results based on the user specifications. While the microservices deal with the computationally and memory intensive portions of NRAQC, they do not govern the entire system itself. The NRAQC client presents a number of features to the user and when the user selects a task, the NRAQC client then communicates with the microservices via HTTP.

V. Discussion

In the environmental field, microservice architecture is often used to drive software with narrow goals. Multiple microservices are usually developed and used to create web-based

TABLE I: Feature-based comparison table

Feature Description	MESA	DIMMER	OceanTEA	PEIS
Requires refactoring entire system		x	x	x
Support multiple applications	x	x		x
Oriented toward environmental research	x		x	x
Service Discovery features	x	x		x
Supports multiple databases	x	x		x
Uses Containerization	x		x	x
Uses Continuous Integration	x			x
Capable of High Performance Computing solutions		x		x
Machine Learning Capabilities				x

support for a singular application, such as the ones described in OceanTEA. For research outside the earth sciences, the microservice architecture is often used as platform for providing an abstract data layer between an application and the data source of the project, as described in the DIMMER Smart City Platform. Interestingly enough, both research inside and outside the Earth sciences share a common trend of refactoring systems into microservice architectures when the problem involves an existing monolith. This approach usually involves the decomposition of a monolith into requirements that are mapped to microservices in the hopes of enabling scalability. This conversion can especially be seen in the PEIS system mentioned above.

While both of the described trends showcase two valid and intended use cases of the Microservice Architecture, the complete refactoring of a system from monolithic to microservice-based brings about serious concerns. In environmental research projects, especially projects ranging from a single university to an entire state, monolithic system designs are common due to unexpected growth in a project or from a sheer lack in pooled technical knowledge. Many of these projects simply do not have the resources or people to simply halt progress and perform an entire system overhaul. Additionally, many environmental projects often autonomously collect data from sensor networks and shutting down these systems, even briefly, can cause detrimental effects on the research produced by the overall project.

It is through these concerns that brings to light the novelty of MESA as a microservice-based system. MESA is designed as a scalable application development platform to support critical systems, especially monoliths, without having the need to tear down the existing system. MESA connects to a regularly updated replication of the main NRDC database that houses copies of the incoming data, so it does not interfere with the monolith whom autonomously gathers data at set intervals. To clarify, this approach does not eliminate the option of full system migration, and provides the means to lessen the burden of the demands placed on developers until a solution is decided upon and implemented. Should a full migration be decided, MESA can be utilized to shoulder the burdens of inactive systems and can eventually become decommissioned once the migration is complete.

A feature comparison, shown in table I, was created to measure MESA's capabilities against the previously described microservice systems. MESA brings forward a unique contribution to Environmental Research in that it does not require a complete system refactoring in order to be used. Additionally, the MESA system carries with it many of the modern software features and industry practices that are present in microservice development. It is through these features that MESA is able to offer more functionality than two similar systems: OceanTEA and DIMMER. OceanTEA, although a very effective system in environmental data gathering, does not offer platform support, a service discovery, or the continuous integration features that MESA does. Similarly so with DIMMER, it does not provide as much features as MESA, lacking in areas like supporting multiple databases, containerization, and continuous integration features. However, MESA still is outperformed by microservice systems used by the larger environmental projects, like PEIS who is able to support advance features such as HPC or Machine Learning. Overall, MESA's abundance of features makes it a solid alternative to a development platform for small to medium scale environmental projects.

VI. Conclusions and Future Work

A. Conclusion

The system described in this paper, the Microservice-based Envirosensing Support Architecture, focuses on creating an alternative approach to cyberinfrastructure for environmental research projects without enforcing a costly system refactoring. The central idea of this system is to create an applications platform centralized around a regularly replicated data source without having to tear down a monolith. The option to completely refactor a system and break apart an active monolith is expensive and requires a massive amount of technical knowledge and time to execute. This application platform was created by using a series of microservices that break up business requirements into independent web services. The microservices answer to a central service discovery and are generally mapped to a feature within a client application.

MESA is a relevant and beneficial system to environmental research projects due to its ability to provide platform support to a field that is often limited by the technical aspects of software development. While the idea of switching to a distributed architecture, like microservices, can be attractive to growing environmental projects, the reality of the matter comes down to whether the project has the time to halt progress while development is made and if there are adequate resources available to achieve this result. So oftentimes, environmental

scientists are forced to choose between two extremes: a limited older system or an expensive new system. MESA brings to the Earth sciences a third choice that can bridge the gap between the previous two while incorporating industry practices, such as containerization and continuous integration.

B. Future Work

Work is currently underway to make the readings gathered by the sensor towers to be more readily available and accessible as datasets for machine learning. This is a larger focus for MESA, while HPC services are currently being provided by a collaboration with the state of Nevada and Switch, a global leader in data center technologies.

To prevent the interception of data and verify that the client has clearance to interact with data, most modern RESTful-based software practices token-based authentication. Unfortunately, MESA does not utilize this industry-standard practice as of yet. Currently, the services perform this actions without verification and are highly susceptible to being intercepted. This is due to MESA being a prototype to show a proof of concept and security additions are considered secondary features. In the future, a major enhancement to the MESA system would be the application of modern security practices.

Although MESA uses containerization technology in the form of Docker, MESA only has Docker containers operating on approximately a third of the active microservices. Docker has commonly been used for Linux environments and while it does have Windows versions, it requires the deft hand of a system administrator or a DevOps engineer. During the development of MESA's Windows-based microservices, this task was deemed secondary as to allow more focus on supporting environmental research applications. However, recent talks and advancements within the project have advised steering toward a migration onto a Kubernetes environment to host the Docker containers. This would allow MESA to achieve total containerization of microservices in future iterations of the project.

Acknowledgment

This material is based in part upon work supported by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] Ronacher Armin. Flask microframework. URL: http://flask.pocoo.org/. [Online; accessed February 17, 2022].

[2] Eric Braun, Thorsten Schlachter, Clemens Düpmeier, Karl-Uwe Stucky, and Wolfgang Suess. A generic microservice architecture for environmental data management. In *Environmental Software Systems. Computer Science for Environmental Protection: 12th IFIP WG 5.11 International Symposium, ISESS 2017, Zadar, Croatia, May 10-12, 2017, Proceedings 12*, pages 383–394. Springer, 2017.

[3] Sergiu Dascalu, Frederick C Harris Jr, Michael McMahon Jr, Eric Fritzinger, Scotty Strachan, and Richard Kelley. An overview of the Nevada Climate Change Portal. *7th International Congress on Environmental Modelling and Software*, 2014.

[4] Nevada EPSCoR. Solar Energy Water Environment Nexus in Nevada. https://solarnexus.epscorspo.nevada.edu/. [Online; accessed February 17, 2022].

[5] Hashicorp. Consul. https://www.consul.io/. [Online; accessed February 17, 2022].

[6] Desert Research Institute. Scaling environmental processes in heterogeneous arid soils (sephas). URL: https://www.dri.edu/sephas. [Online; accessed February 17, 2022].

[7] Arne Johanson, Sascha Flögel, Christian Dullo, and Wilhelm Hasselbring. Oceantea: exploring ocean-derived climate data using microservices. *International Workshop on Climate Informatics (CI 2016)*:24–29, 2016.

[8] Alexandr Krylovskiy, Marco Jahn, and Edoardo Patti. Designing a smart city internet of things platform with microservice architecture. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 25–30. IEEE, 2015.

[9] Vinh D Le, Melanie M Neff, Royal V Stewart, Richard Kelley, Eric Fritzinger, Sergiu M Dascalu, and Frederick C Harris. Microservice-based architecture for the nrdc. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1659–1664. IEEE, 2015.

[10] Microsoft. Windows communication foundation. URL: https://docs.microsoft.com/en-us/dotnet/framework/wcf/. [Online; accessed February 17, 2022].

[11] Rakhi Motwani, Mukesh Motwani, Frederick C Harris Jr, and Sergiu Dascalu. Towards a scalable and interoperable global environmental sensor network using service oriented architecture. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2010 Sixth International Conference on*, pages 151–156. IEEE, 2010.

[12] Hannah Munoz, Connor Scully-Allison, Vinh Le, Frederick C Harris Jr, and Sergiu Dascalu. A mobile quality assurance application for the nrdc. *Proceedings of the ISCA 26th International Conference on Software Engineering and Data Engineering (SEDE 2017)*:61–66, 2017.

[13] Peter Rogers. Service-oriented development on netkernel- patterns, processes & products to reduce system complexity. URL: http://www.cloudcomputingexpo.com/node/80883. [Online; accessed February 17, 2022].

[14] Connor Scully-Allison, Vinh Le, Frederick C Harris Jr, and Sergiu Dascalu. Near real-time autonomous quality control for streaming environmental sensor data. *22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, 2018.