SUIS: simplify the use of geospatial web services in environmental modelling

Ziheng Sun, Liping Di*, Juozas Gaigalas

Center for Spatial Information Science and Systems, George Mason University, Fairfax, VA, USA

Ziheng Sun - Email: zsun@gmu.edu; Tel: (+1) 703 993 6124; Mail: 4400 University Dr, MSE 6E1, Fairfax, VA, 22030; Fax: (+1) 703 993 6127; ORCiD: 0000-0001-9810-0023

Liping Di (Corresponding author) - Email: ldi@gmu.edu; Tel: (+1) 703 993 6114; Mail: 4400 University Dr, MSE 6E1, Fairfax, VA, 22030; Fax: (+1) 703 993 6127

Juozas Gaigalas - Email: juozasgaigalas@gmail.com; Tel: (+1) 207 318 0332; Mail: 4400 University Dr, MSE 6E1, Fairfax, VA, 22030; Fax: (+1) 703 993 6127

Ziheng Sun is a research assistant professor in CSISS.

Liping Di is a professor at George Mason University and director of CSISS.

Juozas Gaigalas is a Ph.D. student and research assistant in CSISS.

This work was supported by the National Science Foundation under Grant 1740693 and 1440294.

SUIS: simplify the use of geospatial web services in environmental modelling

Abstract: Today environmental scientists heavily rely on geospatial web services (GWS). However, many online facilities are under-utilized by the environmental modelling community because accessing the disparate service interfaces requires highly specialized technical expertise. This paper proposes a Simple Universal Interface for Services (SUIS) framework which is a client framework for accessing heterogeneous services via a single unified interface to simplify service access. The supported services including Open Geospatial Consortium (OGC), Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) services. SUIS relieves modellers from having to learn the details of service technologies such as protocols, bindings, and schemas. SUIS4j, a Java implementation of the SUIS framework, is developed and tested to combine multiple operational GWS to demonstrate geoprocessing workflows in agricultural drought monitoring and coastal ocean modelling. The results confirm the expected benefits. SUIS is demonstrated to support simplified use of geospatial cyberinfrastructure for ad-hoc environmental model integration.

Keywords: geospatial web service; interoperability; Earth scientific model; simplification; geoprocessing workflow.

Highlights

- New simple and universal service client framework (SUIS) is proposed for reducing complexity and engaging the full potential of geospatial web services.
- Understandable and descriptive interface for environmental modellers/scientists
- SUIS makes technical terminologies on network communications invisible to scientists
- SUIS enables simple and effective composition of web services to perform agricultural drought and coastal ocean modelling
- SUIS add negligible time cost (<10 milliseconds) into service performance

Software availability

Name of software: suis4j

Developer: Center for Spatial Information Science and Systems, George Mason University

Source language: Java

Contact Information: zsun@gmu.edu

Availability: The source code and application jar can be accessed via Github: https://github.com/CSISS/suis4j.

1. Introduction

Following the realization that the traditional personal computer oriented analysis workflows are hindering the use of large volume of geospatial data due to limited disk space and computing capacity (Wagemann et al., 2018), geospatial web services (GWS) appeared and brought great benefits to Earth scientists by providing web access to massive geospatial datasets and functionalities in an elastic manner (Hey and Trefethen, 2005; Richard et al., 2014; Vitolo et al., 2015; Wright and Wang, 2011). Driven by the idea of "e-Science" (Hey and Trefethen, 2005), tens of thousands of web services were developed and deployed to continuously serve millions of spatial records and datasets on a daily basis. More and more free and open source software (FOSS) for web applications and geographic information systems became available and used by data vendors to deploy their own thematic web services, allowing vendors to directly connect with stakeholders (Swain et al., 2015). These web services are the key tools that enable modellers to conduct data-intensive science (Hey and Trefethen, 2005). Today, researchers in the whole spectrum of Earth science domains including geography, geology, geophysics, oceanography, glaciology, atmospheric sciences and so on, frequently rely on GWS to search, download, visualize, analyze, and disseminate data. Powerful tools could definitely improve the conduct of environmental research (Hey et al., 2009). However, more powerful tools are usually more complicated, because simplicity is sacrificed in exchange for flexibility and generality. Unfortunately, many scientists are prevented from using the full power of GWS because the service client capacity is limited while GWS service interfaces are too complex. Scientists wishing to utilize these powerful services are forced to understand intricate technical details and processes (Fig. 1) that are not intuitive or easily comprehensible to users who lack computer and geospatial interoperability backgrounds. Each type of service takes a different approach to technical details such as operation names, parameter names, data types, formats, schemas, value options, special tokens, protocol headers, and exception codes. This breadth of options supports flexibility at expense of service adoption.

Figure 1. The word cloud of disparate interfaces in geospatial cyberinfrastructure (produced by word cloud generator from the terminologies from collected online documents of geospatial cyberinfrastructure standards, e.g. OGC, TC211, FGDC, and related web service blogs)

The growth of web-based resources in recent years has made this problem worse. Imagine a hypothetical seismic scientist who wants to combine an IRIS

(Incorporated Research Institutions for Seismology) REST service (Shapiro et al., 2005) that offers time series of waveform data with a UCAR (University Corporation for Atmospheric Research) Web Coverage Service (WCS) offering radar data and a Web Processing Service (WPS) (OGC, 2007) to perform re-gridding. After their datasets are processed they want to use a SOAP service (Clements, 2002) for data integration. Typically, they will install desktop client software such as ArcGIS (Institute, 2001) or QGIS (Team, 2013) for OGC services and find a Jupyter notebook (Kluyver et al., 2016) or write custom code for the SOAP and RESTful services. This researcher might spend days studying obscure web APIs (Application Programming Interfaces) and navigating unnecessary software functionality. Once they master accessing these varied GWS interfaces they still won't be able to programmatically chain the services together automatically to deliver the data in its final form. Instead, they will spend more time pre-processing the data manually or writing custom scripts. Although GWS that offer required pre-processing capabilities exist, and even though the researcher is already using some GWS to download the data and do the final data integration – they will avoid utilizing existing web GWS for pre-processing because to them those interfaces appear obscure and cumbersome to access. Instead of taking advantage of these powerful facilities, our hypothetical researcher will avoid learning confusing technical details and will continue to rely on inefficient and time-consuming but familiar procedures. This is a persistent problem and naturally many scientific communities have voiced their desire for simplified access to these powerful online facilities to reduce time spent performing manual data pre-processing (Kelbert, 2014).

This paper proposes to solve the problem by applying a simple universal interface for services (SUIS) framework to GWS clients. SUIS client framework bridges the disparate service interfaces with a single generic interface that carefully abstracts service technical details such as protocols, styles, bindings, schemas, and addresses. Only the intuitive information for each service like operation names, parameter names and data types are exposed to end users. That information is generally intuitive and easier for scientists to comprehend because it is directly related to the scientific requirements of specific research fields. Operations in SUIS are mapped to the actions in the original service interfaces. For each type of services SUIS provides a driver that accomplishes the simplified mapping automatically. This means that once users learn how to use SUIS they can access all standard-conforming online geospatial cyberinfrastructure.

The major benefits of SUIS are reduced barrier of entry and reduced risk of misunderstanding between endpoint consumers and service providers. With SUIS, endpoint users of GWS are separated from heterogeneous interfaces and access with all services via the same set of uniform processes. SUIS aims to lighten the burdens of learning about unnecessary software and to ease the pitfalls of coding clients to interact with complicated geospatial web service interfaces. Additionally, by alleviating problems of interface complexity and heterogeneity SUIS supports easier composition of GWS into workflows. Generic SUIS operations can be chained into geoprocessing models (Di, 2004; Yue et al., 2010) that map to executable workflows composed of the multiple services (Chen et al., 2009; Yu et al., 2012).

A Java library named suis4j was implemented to demonstrate and evaluate the SUIS concept. suis4j currently supports the basics of SOAP/WSDL (Web Service Description Language), RESTful/WADL (Web Application Description Language), WPS (version 1.0.0), WCS (version 2.0.0), WMS (version 1.3.0), and WFS (version 2.0.0). suis4j was tested and its performance was evaluated with various existing web services. The experiment shows that the SUIS approach shields users from overwhelming and unnecessary technical details and allows users to take advantage of GWS in their applications in a simple way.

The remainder of the paper is organized as follows. Section 2 introduces the background of this research and lists the existing related work. Section 3 describes the objectives and design principles underlying SUIS, while Section 4 presents SUIS framework in full detail. In Section 5, the implementation of suis4j is presented. Section 6 describes how suis4j was tested with prominent online infrastructures in the Earth science community. Section 7 discusses the pros and cons of SUIS framework. Section 8 concludes this research and maps out future work.

2. Related Work

This section talks about the circumstances and explains why simplification is an inevitable trend in geospatial cyberinfrastructure.

2.1 GWS Interfaces

The interfaces of GWS can be generally divided into three groups: SOAP (Simple Object Access Protocol) interfaces, REST (Representational State Transfer) style interfaces and OGC (Open Geospatial Consortium) standard-compliant interfaces (shown in Fig. 2). Their regimes could be overlapped because a service may belong to more than one group. For example, an OGC WPS could simultaneously provide both a SOAP endpoint and REST endpoint.

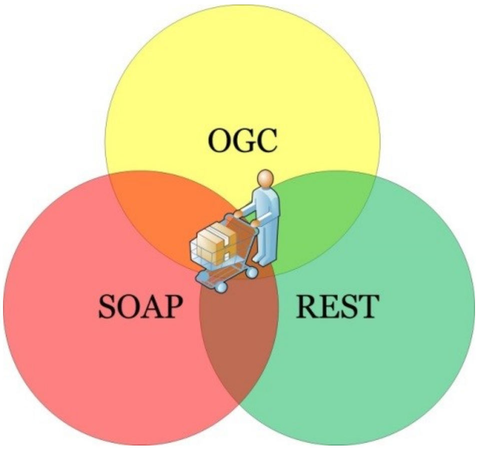


Figure 2. Three major categories of GWS on the market

W3C identifies a set of common core technologies for web services. The main ones are HTTP (HyperText Transfer Protocol), XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), WSDL, etc (W3C, 2015). SOAP messages are exchanged through XML payloads that are transmitted via HTTP POST. WSDL is used to describe SOAP web services interfaces (Chinnici et al., 2007; Christensen et al., 2001). Historically, the SOAP standard has maintained the monopoly position in service-oriented architecture (SOA), but as new standards have emerged SOAP has become one of several options in the market. SOAP is well regarded due to its domain independence and security. SOAP and WSDL services are employed in many B2B (business-to-business) and B2C (business-to-consumer) industries such as chemistry (Kim et al., 2015), travel planning, hotel booking (Dhara et al., 2015) and decision support (Demirkan and Delen, 2013).

REST is the newcomer and provides a lighter weight alternative to SOAP. It is an architectural style of web services and is NOT a standard. REST is used widely to develop World Wide Web applications. In REST, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs). REST requires that actions on the resources are limited to a small set of simple, well-defined operations. Usage of REST in web APIs has skyrocketed in the past decade because RESTful services are lightweight, highly scalable and maintainable. WADL is a schema format developed to describe RESTful applications (Hadley, 2006).

Both SOAP and REST are domain-independent. However, in many scenarios, target-specific interfaces are required to facilitate geospatial applications (Dietz, 2010). Organizations like ISO/TC211 (the International Organization for Standardization Technical Committee 211) and OGC (Open Geospatial Consortium) have developed a series of standards targeted to the specific requirements from the geospatial community. ISO focuses on standardization of geographic information and geo-informatics (Ostensen and Smits, 2002) while OGC majorly works on standardizing service interfaces and data models (Di, 2003). Some of their well well-known products are ISO19115:2003 (metadata), ISO19119:2005 (geographic information - services) (Percivall, 2002), WMS (WMS, 2004), WCS, WPS, WFS, SOS (Sensor Observation Service), SPS (Sensor Planning Service), CSW (Catalog Service for the Web) and OpenLS (Open Location Service) (Botts et al., 2008; OGC, 2016). To further advance interoperability ISO/TC211 and OGC hold a cooperative agreement that allows them to cite each other’s standards (ISO/TC211, 2009).

These standards have greatly improved the interoperability among geospatial web services. However, the proliferation of standards has greatly increased the heterogeneity of service interfaces. The more complicated the standards are, the greater barriers of entry they present to scientists. This complexity is one important reason for the low adoption rate of GWS by endpoint users.

2.2 Geospatial Cyberinfrastructure and Spatial Data Infrastructure

Cyberinfrastructure ideas have been gradually embraced by Earth science community and many teams have developed online digital systems to meet cyberinfrastructure needs that were previously unmet for years (Council, 2007; Richard et al., 2014; Sun et al., 2014). That has spurred new research which utilizes web-based instruments, sensors, high-powered computers, data storage capabilities, visualization facilities, and networks for communication and collaboration (Berman and Brady, 2005; Hofer, 2013). Spatial Data Infrastructure (SDI) is one type of cyberinfrastructure. In the early days of GIS, SDI was designed primarily for sharing geographic information in response to the high cost of information collection and maintenance. Later, SDI efforts have evolved towards the creation of shared, distributed, and interoperable environments through GWS (Davis Jr and Alves, 2005). In SDI, data providers register their services with a public server that scientists can use to search, select data services of their interest and reach those services through the Web (Table 1). SDI enables users to retrieve the latest version of the data products and simplifies the requirement for endpoint devices that can remain lightweight without the need for large local storage space. Despite advances in SDI, many geospatial scientists who recognize the need for cyberinfrastructure continue to hold a "wait and see" attitude rising from the concern that the systems will not be helpful without broader input from the communities they are meant to serve. Cyberinfrastructure community needs to engage the geoscience population to reach a consensus on what kind of cyberinfrastructures are the most suitable for the community (Mookerjee et al., 2015).

Table 1. The popular online geospatial cyberinfrastructures

Name	Searchable	Object	Server Interface	Portal	Provider
CWIC	✓	Data	CSW/OpenSearch	http://cwic.wgiss.ceos.org	CEOS
Unidata	x	Data	TDS	http://thredds.ucar.edu	UCAR
EOS	x	Data	HTTP	http://eospso.nasa.gov/	NASA
GCMD	✓	Data & GWS	HTTP	http://gcmd.nasa.gov/	NASA
GEOSS	✓	Data & GWS	CSW	http://www.geossregistries.info	GEO
U.S. Water	✓	Data	HTTP	http://water.usgs.gov	USGS
USGS Catalog	✓	Data	CKAN	https://data.usgs.gov	USGS
Data.gov	✓	Data & GWS	CSW/CKAN	https://data.gov	GSA
NOAA Catalog	✓	Data	CKAN	https://data.noaa.gov	NOAA
NCEI Ocean Archives	✓	Data	TDS/HTTP/FTP/DAP	http://data.nodc.noaa.gov/geoportal	NOAA
AWS Public Datasets	x	Data	HTTP	https://aws.amazon.com/datasets/	Amazon
FGDC Catalog	✓	Data	CKAN	https://cms.geoplatform.gov/data/	FGDC

2.3 Client Framework for GWS

Most web service frameworks offer client frameworks for users to embed the code calling web services into their application, e.g., Apache Axis (SOAP/REST), Apache CXF (SOAP/REST), gSOAP (SOAP), .NET Framework (REST), Yii (REST), Jersey (REST), Spring (REST), etc. Besides that, service consumer groups develop some independent frameworks for GWS, like ArcGIS, QGIS, gVSIG, and SAGA GIS developed their own embedded client framework which is usually hidden and specifically invoked by a plugin dialog. OGC has invested a lot of efforts in unifying OGC web service interfaces, e.g., OWS common (Whiteside and Greenwood, 2010), OWSLib (Kralidis, 2015), and GeoAPI (Custer, 2011). OWS common defines unified GetCapabilities operation and other minimum utilities and is the basis of most OGC service interface standards. GeoAPI and OWSLib are Java/Python client interfaces aiming to formalize the handling of the types defined in the OGC specifications. However, they are fully engaged with detailed technical terminologies in OGC standards and will cost a lot of time of the environmental scientists who don't want to invest too much time on web service. In industry, commercial service providers normally develop new client framework to interact with their own services, such as the Python/Java/Javascript/Go client library for Google Maps web services, interactive SDK for Bing Maps web control, MapKit JS client for Apple Maps, JavaScript API client for ArcGIS REST services, simple API client for OpenWeatherMap, javascript API of MapQuest, web/mobile SDKs for Here WeGo maps, etc. All these client frameworks are independent, very different, and require long-term engagement and interest, which is not realistic in environmental modeling. Scientists need to focus on environment models rather than various service client SDKs and a more universal and simple client framework will be of their interest.

2.4 Geoprocessing Workflow & Earth Science Modelling

Geoprocessing denotes processing geographical data and is the core part of geographic information system (Allen, 2011; Chen et al., 2009; FRISBIE, 1979; Goodchild, 1982; Kinzy, 1978; Mark, 1979; Roberts et al., 2010; Sun et al., 2012). A geoprocessing workflow is a chain of several atomic functions to achieve more complex tasks (Di et al., 2006). The available atomic processes differ among platforms. In ArcGIS, the processes are the tools in ArcGIS toolbox. In cyberinfrastructure, the processes are web services registered in centralized catalogs. Geoprocessing workflows are one of the major users of GWS and they greatly extend the capability scope that cyberinfrastructure can cover. But workflow approaches to modelling still struggle to advocate themselves within the community. Scientists have difficulties to leverage workflows in real science. They are supposed to ease the burden on scientists for processing data but eventually they leave users with another big burden of dealing with workflow. The workflows become even more complicated once they involve ontologies, provenance, inference-based automation, etc. The hard-to-use impression of GWS contributes to the unpopularity of geoprocessing workflows. In recent studies, integrated environmental modelling (IEM) has been identified as a structural way to develop and organize environmental models (Gao et al., 2019; Jakeman and Letcher, 2003; Kelly et al., 2013; Laniak et al., 2013). Geoprocessing workflow approach is listed as an option for constructing integrated models composed of heterogeneous atomic processes (Yue et al., 2015). The approach is promising but remains too complicated for Earth scientists without web service background.

2.5 Existing efforts for simplicity

Cyberinfrastructure community has recognized the complexity problem and has attempted to shield end users from some of the complexity. Initially, they studied the causes of complexity. Shen et al (Shen et al., 2007) concluded five types of heterogeneous issues among web services including semantic, parameter data type, parameter structure, parameter number, and parameter data unit. Many attempts have been made to extract common things among web services and create a generic interface to simplify the calling procedure on the client side. For instance, Schindler et al present a generic and flexible framework for building geoscientific metadata portals independent of content standards for metadata and protocols (Schindler and Diepenbroek, 2008). Kiehle et al built a generic service utilizing spatial standards of OGC, ISO, and W3C (World Wide Web Consortium) for providing common geoinformation capabilities in SDI (Kiehle et al., 2006). de Souza Munoz et al propose a generic approach called openModeller to handle different data formats and multiple algorithms that can be used in potential distribution modeling and make it easy in data preparation and comparison between algorithms using separate software applications (de Souza Muñoz et al., 2011). Trabant et al used a simple subset of RESTful concepts, common calling conventions, a common tabular text dataset convention, human-readable documentation and tools to help scientists learn how to use the web services from IRIS Data Management Center (Trabant et al., 2015). Burkon et al tried to develop and demonstrate the practical use of a generic model of service's interface that could be used as a basis for creation of a formal description of any service in any industry (Burkoň). Mackiewicz discussed the benefits of applying the generic interface definition (GID) of IEC 61970 to power system operations and industrial applications (Mackiewicz, 2006). Tristan et al introduced a generic service wrapper enabling the optimization of legacy codes assembled in application workflows on grid infrastructure (Glatard et al., 2006). Most research work focuses on the server side and attempts to unify the service interfaces. However, the current landscape is not favorable for unifying service interfaces across domains and industries because service providers have different business goals and different prior knowledge. Because it's not reasonable to expect the existing service providers will simplify their interfaces this work should focus on the client and create a simple client framework that can handle the disparate service interfaces and provide a universal calling interface for end users.

3. Objective and design principles

Our general objective is to hide scientifically irrelevant technical details of geospatial web services and to expose only application-related information to end users. A generic client framework is created to act as a system building-block that bridges scientific end-users and disparate geospatial web services (Fig. 3). The general objective is supported by three specific principles: a) keeping the processing interface simple, b) making GWS more composable in the environmental workflow, and c) seeking common ground to become a universal solution.

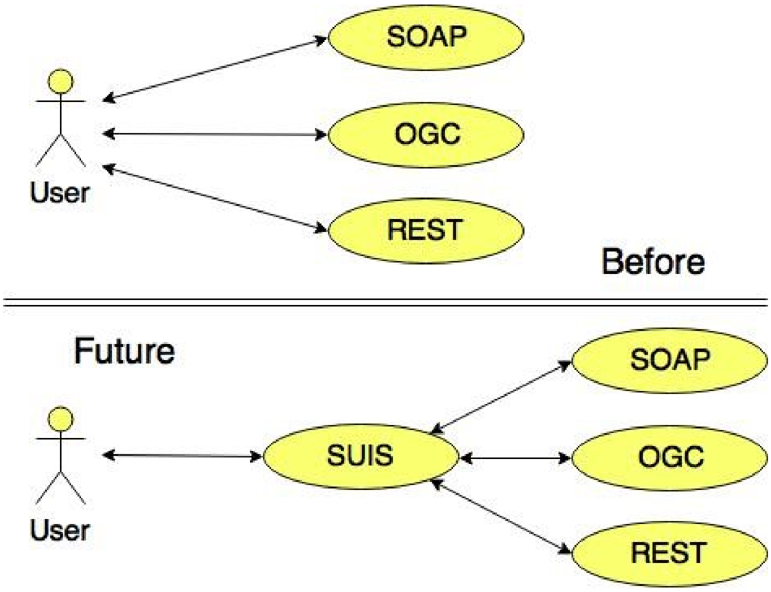


Figure 3. SUIS objective

3.1 Keeping It Simple

The geospatial process interfaces that take inputs and produce outputs are simple in traditional GIS, but have become complicated after being translated into GWS paradigm. The protocols for information communication have evolved concurrently with the long-term organic development of the Internet. The complex historical development of web service technologies has embedded specialized technological knowledge into the GWS interfaces. As case in point, every ordinary World Wide Web Consortium (WWW) service request-response exchange involves multiple layers of historical technical minutia that are irrelevant to web user's needs. The World Wide Web is successful because Web browsers engineers have artfully leveraged protocols without drowning web users in technical details – protocol details are hidden far away from the surface that end users see. Meanwhile, consumers of geospatial web service face complicated and frustrating client software that directly exposes GWS protocols to end users. Service consumers must deal with a full stack of engineering information – from data exchange to operation semantics. For instance, for a non-GWS-expert environmental scientist, the XML-formated messages with many redundant and deeply nested labels are very likely to produce confusion and frustration. Our objective requires that technical details about communication protocols are simplified and hidden away from the application logic.

Additionally, current GWS interface descriptions have too many layers. In WSDL, a service has bindings, a binding has port types, a port type has operations, an operation has input elements, an input element has messages, a message has schemes, and a scheme allows numerous compositions. It is normal to initially become lost while figuring the relationships among these terminologies. In most cases, those concepts are only meaningful to expert users. It is unreasonable to have every user encounter them. Although the layered architecture enhances engineering flexibility when building loosely coupled services, it correspondingly raises interface complexity barriers for consumer comprehension. To provide a geospatial service process interface that is as simple as GIS, SUIS removes those description layers that are not relevant to general users.

3.2 Making GWS Composable in Environmental Workflow

At present, the adoption of GWS in the workflow is much less than was expected when GWS were introduced (Lopez-Pellicer et al., 2012). Most scientists treat geospatial web services as simple tools for data access, which is just one aspect of the design goals of cyberinfrastructure. GWS permit a major interoperability breakthrough of computer and network technologies that can directly support and transform the conduct of scientific and engineering research and yield revolutionary payoffs by empowering individual researchers and by increasing the scale, scope, and flexibility of collective research (David, 2004). GWS are supposed to be chained into workflows for automation to practically help with most basic steps of the real scientific research and in the analysis of datasets of large-scale areas and extended temporal periods. There are already many successful experiments in using GWS into environmental model workflows in Lab. However, after these years of developments, those vision goals of GWS are never really been achieved in real-world environmental model workflows. To make GWS more composable in the workflow, simplification of the interfaces of GWS to make them usable in workflows is a prerequisite step.

3.3 Seeking for Common Ground to Become Universal

Unifying all GWS interfaces into a single universal interface is challenging because attempts to do that are obstructed by the extreme interface heterogeneity. The reasons for that are highly varied and involve factors like service purpose, operation granularity, nested tree structures, data formats, message schemas, context scenarios, design concepts, technical restrictions, and subjective provider preferences. There is enough idiosyncrasy to make it barely possible to precisely map to all GWS interfaces to a single model of API interface. Our experiences in transforming OGC web services into SOAP services have confirmed that. The famous precept to “seek for common ground while reserving the differences” (Bol, 1987) in this case states the basic rule for designing a universal solution. A universal client interface for all other GWS interfaces should center on the common ground and relegate differences to the background. We need to identify and classify identical or similar interface concepts and then organize them into a complete interface which is neat, consistent and easily intelligible for scientists. The outlying and disparate interface concepts that cannot be unified should be handled via hidden adapters or drivers.

4. SUIS Framework

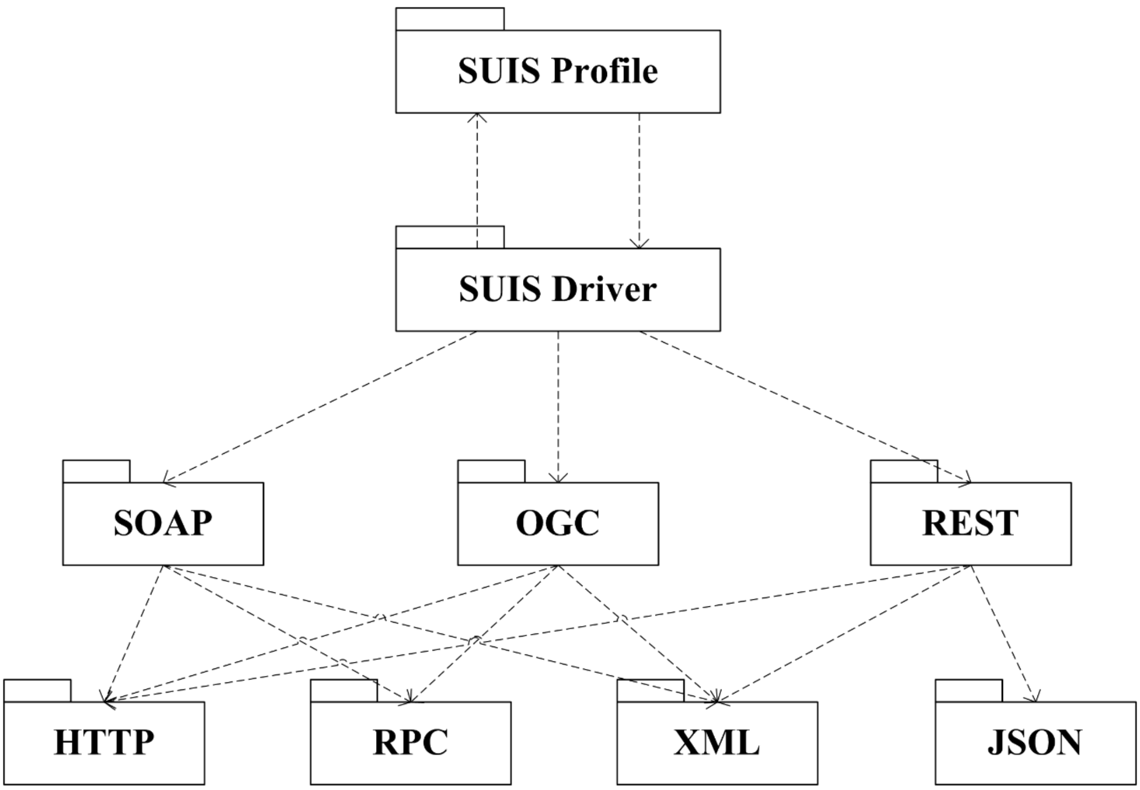


Figure 4. SUIS architecture

This section introduces the core model, architectural design and usage of SUIS. The core model has two major components: profile and driver (Fig. 4).

4.1 Profile

SUIS profile is a model representing the smallest functional unit of GWS. It is composed of three public interface classes (*Operation*, *Message,* and *Parameter*) and enumeration class *DataType* (Fig. 5).

(1) *Operation* denotes the action that the service provides. It includes operation name, input message, output message, and narrative description.
(2) *Message* is the payload exchanged between the client and server. It contains a content variable which could be any object such as JSON (JavaScript Object Notation), XML, and KVP (Key-Value Pairs).
(3) *Parameter* represents the variables that are inputs and outputs in operations. Parameter attributes are identity string, a data type, a parameter name, a parameter value, a description, and minimum and maximum occurrence limits. To support SUIS profile implementation in multiple programming languages, we define parameter value as an abstract object and give SUIS library developers the responsibility to determine the specific data type. Each parameter object must have an attribute referring to SUIS DataType enumeration.
(4) *DataType* has five named constants (Fig. 5) which represent basic data types common to the general database, GIS database, GWS, and general programming languages. The mapping between the conventional data types and SUIS data types is listed in Table 2. We combine similar data types to simplify the service profile. Three new types (BOOL, NUMBER, and DATE) are added to support logic description capabilities.

The DataType enumeration class describes the general types of data content communicated over the Internet using structured data exchange formats such as XML, JSON, KVP, Base64 (Josefsson, 2006), ASCII, Binary, etc. Although using different encodings and protocols, these parameters belong to the same type, FILE. Because non-expert users have no interests to know how the files are encoded or transferred in communication. The encoding and decoding, downloading and uploading details should be erased from the surface and processed automatically on the backstage.

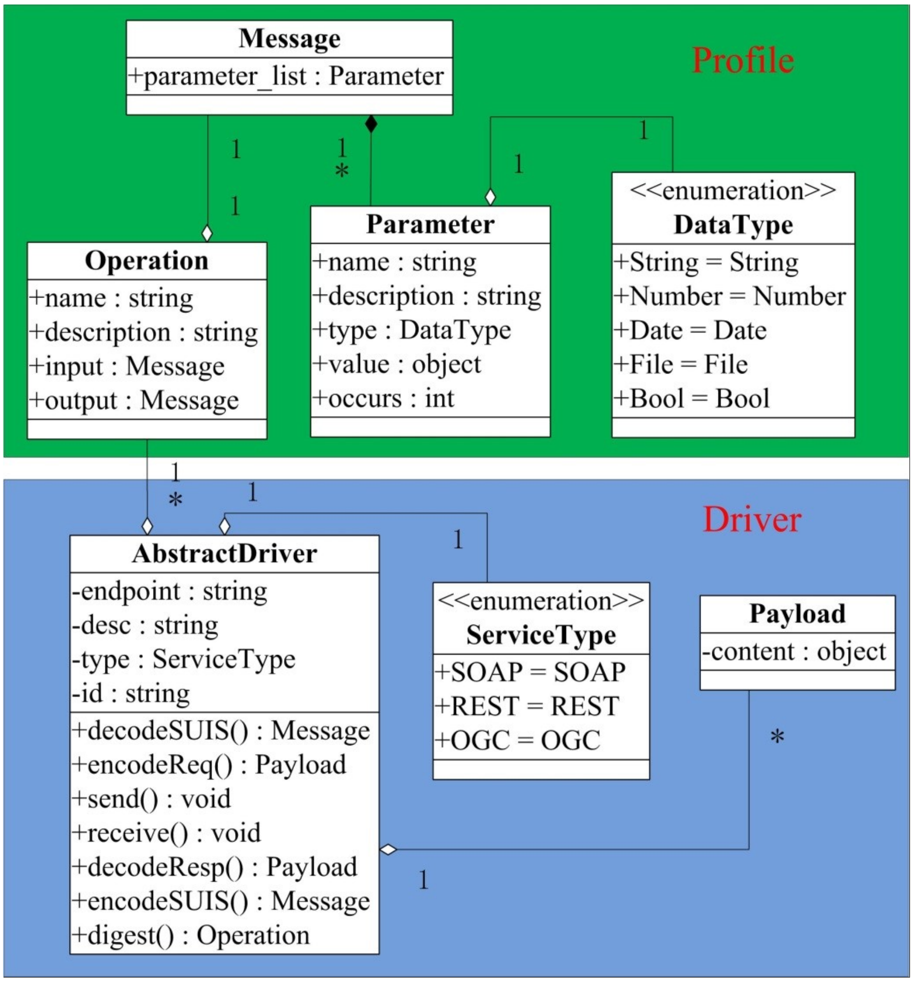


Figure 5. SUIS UML

Table 2. Data type mapping between GIS and SUIS

General & GIS	SUIS
Boolean	Bool
Short Integer	Number
Long Integer	Number
Float	Number
Double	Number
Text	String
Date	Date
BLOB	String/File
Object Id	String
Vector (geometry, point, linestring, polygon, multipoint, multiline, multipolygon, geometrycollection, etc)	String/File
Raster (grid, coverage, picture, etc)	String/File

4.2 Driver

The technical details of each GWS type are isolated and processed in a low-level container, called SUIS driver. The driver wraps the specific service interfaces with the SUIS profile and translates SUIS requests and responses to message formats compliant with service interfaces. Users only interact with the SUIS profile and are not required to understand technical details and complexity encapsulated in SUIS drivers. The SUIS driver backgrounds technical details and acts as a "gray box" which non-experts can treat as a black box while experts and power users can use it to leverage the backend service interface. The driver mechanism makes it easy to transform the existing services into SUIS style without sacrificing their unique capabilities. Each SUIS driver wraps one type of service interface. SUIS architecture allows new drivers to be created for other types of service interfaces that do not belong to the three groups discussed in Section 2. All SUIS drivers must implement a mandatory set of methods for decoding the SUIS requests and encoding SUIS responses as illustrated in Fig. 6: a set of methods that translate SUIS requests to service requests (yellow boxes) and a set of methods for translating service replies to SUIS responses (blue boxes).

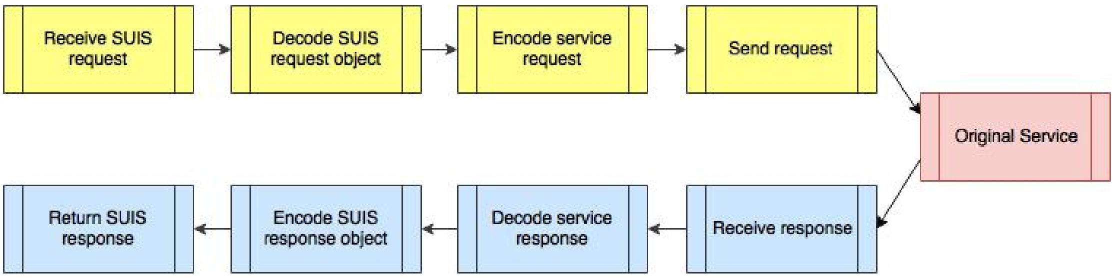


Figure 6. The work steps of SUIS driver

4.3 Mapping

The task of mapping the disparate GWS interfaces to a SUIS profile demands some subtle and challenging design decisions. It requires extracting incompatible operation semantics, identifying their essential information roles and grouping them most effectively using the categories provided by SUIS profile. Specific services allow multiple possible mappings – requiring careful consideration of overall semantics. For example, the common GetCapabilities operation of OGC services can be mapped to a SUIS operation or it can be merged into the initial method digest phase which retrieves service descriptions and initializes the driver. When multiple valid design choices are possible we evaluate each option against the general objective and goals of SUIS

(Section 3). Fig. 7 shows the mapping we created between the three service categories (SOAP, REST, OGC) and the SUIS profile. The mapping is not simple or direct because the ties lack fixed patterns such as one-to-one, one-to-many, or many-to-many. For example, a resource and one of its supported methods in REST interface are combined into a SUIS operation, while the GetCapabilities request is mapped to SUIS operations listing the provided assets. Taken together these complex mapping choices produce a simple and universal API model that represents capabilities of all GWS interface types. The specific level of simplifying on the SUIS interface depends on the acknowledged common requirements from environmental scientists.

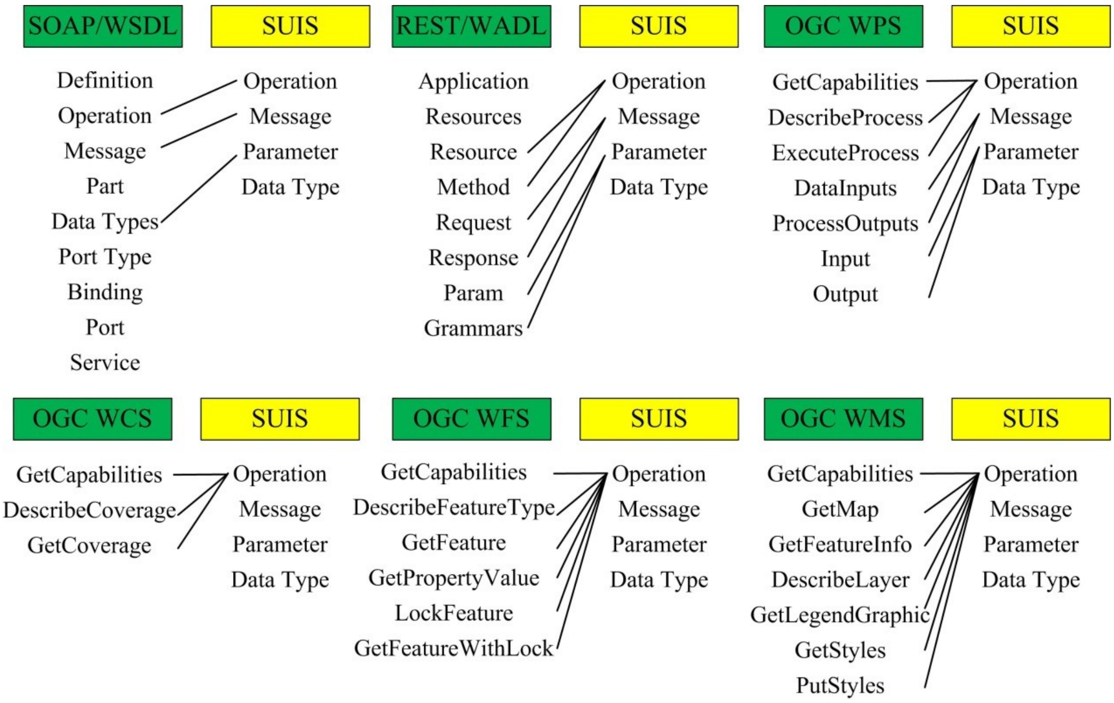


Figure 7. The mapping between existing service interfaces and SUIS profile

4.4 Payload

The data payloads transferred between the SUIS client and the GWS interfaces are automatically generated by SUIS drivers in accordance to the GWS interface schemas. Since the payloads encapsulate superfluous technical details, the SUIS architecture makes them invisible to scientific end users. SUIS users construct SUIS requests that are composed of parameter key-value pairs that represent the core service request information. SUIS drivers automatically decode and wrap SUIS requests into request payloads. In the same fashion, the drivers decode the response payloads and transform them into SUIS key-value pairs. To end users, the transformation from simple SUIS data model to complex payload structure is invisible. SUIS drivers provide two transmission methods, send and receive, for delivering and receiving service payloads. If GWS requires file inputs, the SUIS drivers are required to support at least one of the three ways to transfer files into or out of GWS: URL (simplest), HTTP POST multipart attachment (file size limited), or a third-party file uploading service (e.g., FTP) to turn local files into URLs.

4.5 Usage

SUIS is designed to permit flexible usage that adapts to multiple context scenarios. Scientists are free to choose from a variety of existing GWS facilities (such as mobile or real-time GWS) according to their application requirements. Customized SUIS drivers allow the inclusion of new message structures and formats. The SUIS data types allow users to input or receive either GIS datasets or literal values. In program code, input specification, process activation, and output retrieval tasks from diverse GWS are presented by SUIS in a uniform fashion. Both synchronous and asynchronous modes of operation in the distributed processing environment are supported (Fig. 8). The synchronous mode can be used for instantly responsive services, while asynchronous mode allows interaction with extended duration GWS processes. The SUIS Framework API can be expressed in all general-purpose programming languages such as Java, Python, and C/C++ thus allowing scientists to use SUIS with their preferred languages. The main steps of using SUIS to invoke GWS (Table 3) are:

(1) Initialize SUIS drivers to parse the capabilities of the service, such as the operations, parameters, data types. Capabilities information is used to configure the driver.
(2) Examine the supported operations (optional). Choose the required operation.
(3) Examine input and output parameters of the chosen operation (optional).
(4) Construct the request message by setting values of input parameters.
(5) Send the request and receive the response.
(6) Examine the returned messages (optional).

These steps could be altered to support complex application logic and to support program flow events such as exceptions, to use services that are missing service description file or to perform asynchronous requests. Scientists can skip the service examination steps if they are familiar with the operations. The async mode in SUIS is built because many web services don't support asynchronous requests, e.g. most REST services. For those services with async settings, e.g., WPS 2.0, SUIS driver developers are recommended to directly reuse their native async settings.

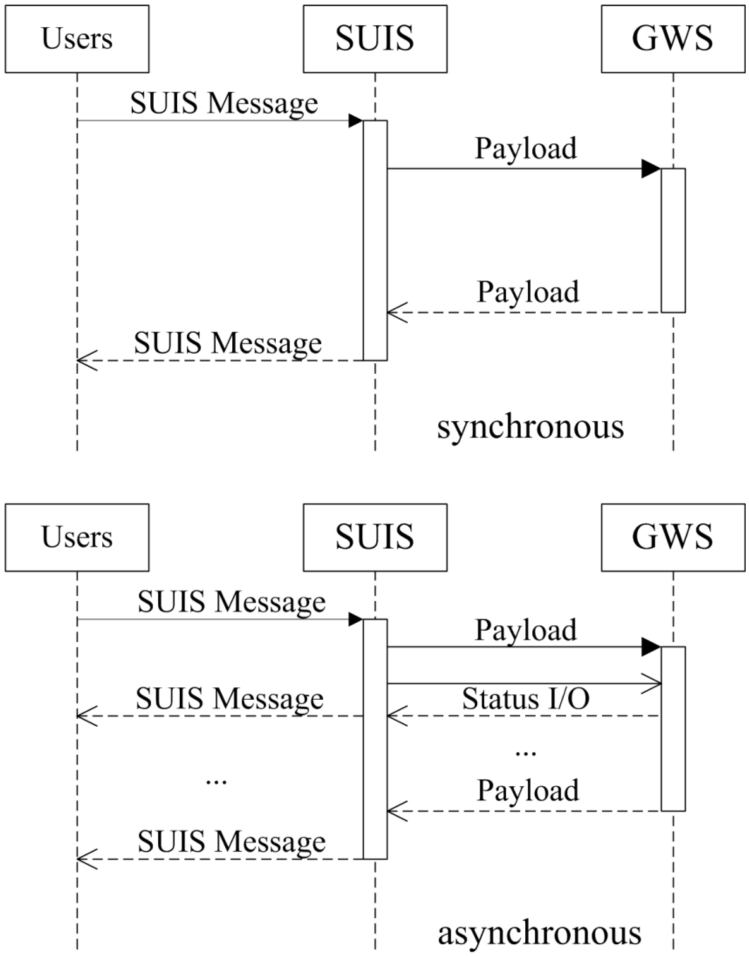


Figure 8. Two modes of using SUIS to call GWS

Table 3. An example of SUIS invoking IRIS REST service

```
//Step 1
SUISClient sc = new SUISClient.Builder()
   .initialize("https://service.iris.edu/irisws/timeseries/1/application.wadl",
   ServiceType.REST).build();
//Step 2
sc.listOperations();
Operation o = sc.operation("http://service.iris.edu/timeseries/1/version.GET");
//Step 3
sc.listInputParams(o);
sc.listOutputParams(o);
//Step 4
```

```
o.input().value("network", "IU")
        .value("station", "ANMO")
        .value("location", "00")
        .value("channel", "BHZ")
        .value("starttime", "2001-12-09T12:00:00")
        .value("endtime", "2001-12-09T12:20:00")
        .value("output", "plot");
//Step 5
sc.call(o);
//Step 6 - optional
sc.listOutputValues(o);
String filepath = o.output().value("return"); //get the data location
```

5. Implementation

The SUIS Framework should be implemented by SUIS developers of different programming languages (e.g., Java – suis4j, Python - suispy, etc). Each library will be maintained by the community of stakeholders who use the corresponding programming language. The client providers like ArcGIS and QGIS can contribute to the development and adopt the SUIS libraries in their software to avoid maintaining their own code to call GWS. Compatibility issues should be fixed by SUIS developers driven by the science user communities.

SUIS has been implemented as a Java library named suis4j. It utilizes several open source Java libraries to achieve SUIS functionality (Table 4). suis4j is available on GitHub (https://github.com/CSISS/suis4j) for downloading and sharing. suis4j development and maintenance follow standard Java ecosystem practices. GitHub issue tracking system is used for fixing bugs and planning enhancements. Apache Maven (Miller et al., 2010) is used to manage dependencies and to build releases. Maven allows developers to easily include suis4j as a dependency into their projects. The code structure is split into two major packages: the SUIS profile and drivers as described in the core framework model. A Client class provides the object-oriented interface for end users to access SUIS capabilities. The library has no dependencies to any complex GIS system and works with all standards-conformant GWS.

Table 4. suis4j dependencies

Library name	Functionality

SoapUI (Kankanamge, 2012)	Composing SOAP requests
JAXB	Parsing XML schemas
XMLBean	Parsing XML schemas
WSDL4J	Parsing WSDL
GeoTools Java Toolkit	OGC standard schema API

6. Experiments

To validate SUIS framework against its objectives we applied the suis4j library to two geospatial science use cases: agricultural drought modelling (Deng, 2013; Sun et al., 2017b) and FVCOM (Finite Volume Coastal Ocean Model) data processing (Chen et al., 2006), both of which involve a number of heterogeneous GWS, including GADMFS (Global Agricultural Drought Monitoring and Forecasting System) WCS (Deng, 2013), NWS (National Weather Service) REST, GeoServer, GeoBrain SOAP services (Di, 2004), and WPS. All the service calls in both workflows are made in synchronous mode to ensure the service outputs are ready as the inputs of other services.

6.1 Agricultural Drought

Suppose we are agricultural drought scientists and we have created a new index to monitor agricultural drought. The equation for the index is:

$$\text{DroughtIndex} = \frac{VCI+MP}{2} \qquad (1)$$

where *VCI* (vegetation condition index) represents the relative status of vegetation comparing to the historical records in the same period. *MP* (monthly precipitation) is derived from quantitative precipitation estimate (QPE) from NWS. The drought index supposes that vegetation status and precipitation are linearly correlated with drought. Remote sensing scientists are continuously searching for indices to accurately reflect observed conditions and this index represents a novel attempt in a realistic agricultural drought research scenario.

Multiple datasets must be combined to calculate the drought index and to do our study. We must retrieve VCI products from GADMFS[1] and then download MP

[1] http://gis.csiss.gmu.edu/GADMFS

products from the NWS AHPS (Advanced Hydrologic Prediction Service) website. Once data is obtained we use GeoBrain web services (Han et al., 2008; Li et al., 2010) to process the two products into the final drought index product. We employ suis4j to automate these tasks into a geoprocessing workflow. The workflow is shown in Fig. 9, where irregular shapes represent GWS, purple rectangles represent operations, dashed lines represent data flow, and solid lines represent SUIS calling web services. We utilize geospatial web services to re-project, clip, and calculate the final drought product based on our index equation. We use suis4j to call the required services in the required order and then link their inputs and outputs to form a chain. We apply the same workflow chain to different days in 2017 to generate a time series of drought products (shown in Fig. 10). Our results show that the long-narrow central part of California (the area between roads I-5 and CA-99) endures agricultural drought for almost the entire year and seasonally (from May to July) drought spreads to cover most places in California. In August, the drought starts to gradually dissipate. To present our results we select the April 23 drought index product and render that as a drought map by overlaying drought index on Google Maps.

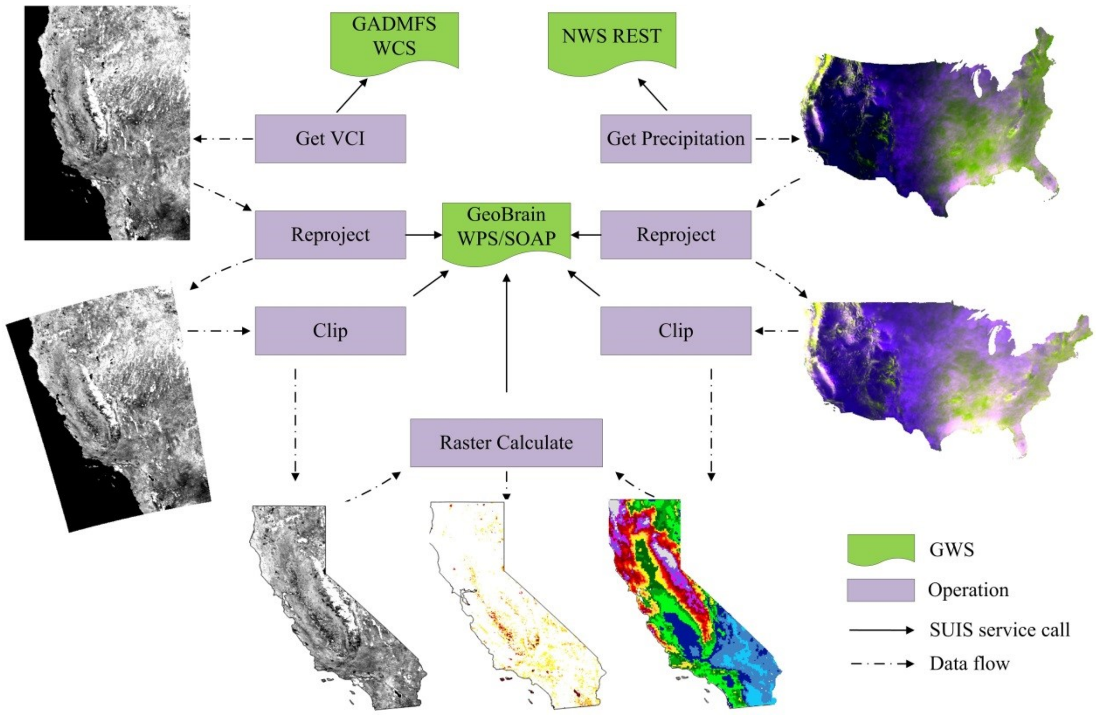
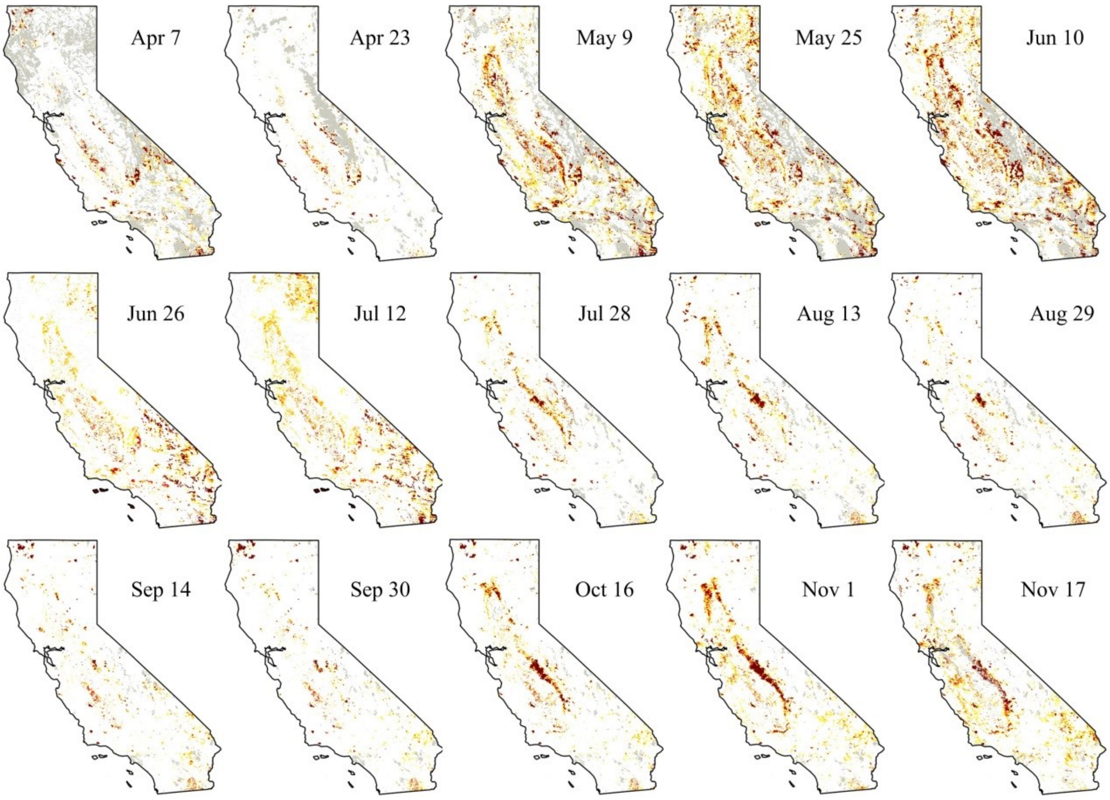


Figure 9. The use of SUIS in drought workflow



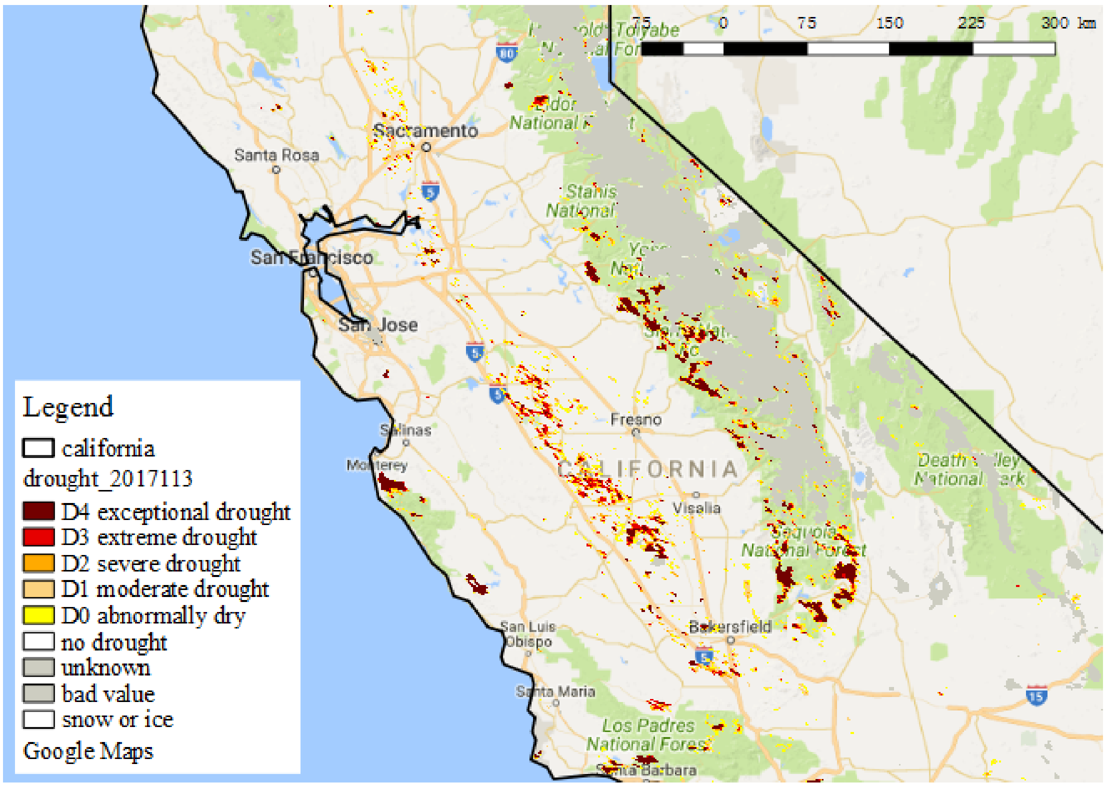


Figure 10. April 23 drought index of California in 2017, generated by suis4j (The base map is Google Maps © Google)

The finished experiment warrants discussion of technical results, especially those related to performance issues. The drought workflow uses web services from two categories: data services and processing services. Both types of services introduce network load. Processing services involve computational load on the server and wait time for the client. Application architecture can be used to address some performance challenges. For example, SUIS application might cache the outputted data from GWS to reduce both computational efforts and network load across multiple application runs. The particular caching strategy depends on SUIS driver developers. The recommended practice is to remember the paths of the files downloaded by users from GWS. Next time when users input the same parameters, SUIS will check the file paths and directly return files to users if they exist. The lifetime of the cached files is equal to the time the downloaded files exist in their cached paths. For time-sensitive requests, if the input parameters to GWS are different from the input parameters which produced the cache files, SUIS will resend the requests for new files; if the input parameters to GWS stay the same, SUIS will provide an option for users to force refresh the cache files by downloading new ones.

To decrease the long delays caused by slow network connections between client and GWS, SUIS supports easy switching between multiple GWS. For example, both GADMFS and NOAA STAR provide VCI products, and GADMFS serves the data via WCS while NOAA STAR uses FTP-based Shell scripts. Scientific users can quickly alter which service SUIS accesses by changing service endpoint and input parameters. Effective SUIS applications can preserve network resources by never downloading remote service data more than once. For example, in traditional usage, the WCS GetCoverage request will download data from the remote server to the local client. Then, as the next step, this data must be uploaded to another location from where it can be downloaded by the re-projecting service. SUIS can make this compound process more efficient by allowing service users to skip the download and upload steps and instead directly pass the WCS GetCoverage URL to the re-projecting service interface (Keens, 2007; OGC, 2007, 2017) (as shown in Fig. 11). No network load is generated as data streams directly from WCS to the re-projecting service without being repeatedly downloaded and uploaded. The fake call mechanism can save the large part of the total time cost and has the added benefit of making the workflow more concise. Furthermore, SUIS can prevent idle blocking while waiting for the result data to be received. Regardless whether a specific geospatial web service supports asynchronous operation semantics, SUIS provides its own asynchronous communication mode to minimize the time scientists spend idly waiting for processing results.

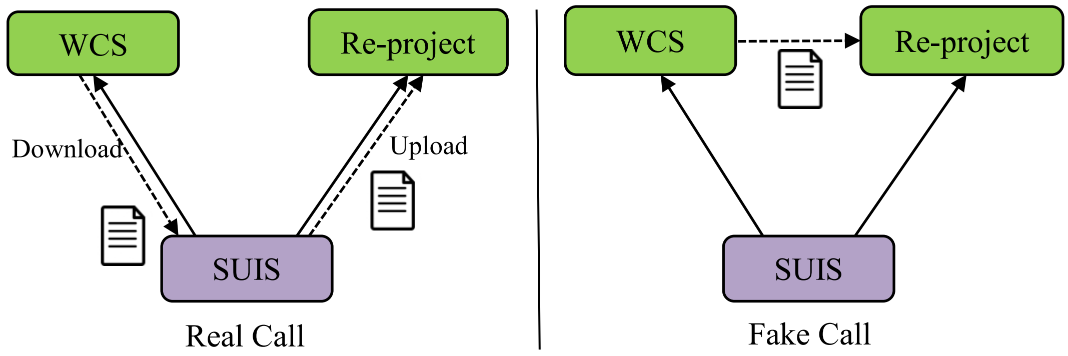


Figure 11. The direct streaming call with SUIS

To derive precise quantitative measures from the aforementioned performance issues, we recorded and evaluated the inputs, outputs, and the duration of each SUIS call. To calculate a representative workload scenario, we made simple assumptions concerning potential users and their behavior. We then derived average values such as inter-arrival times between incoming requests or the requested amount of data from the scenario. When SUIS and GWS exchange messages, each exchange causes extra delays that vary depending on the client and server machines' computing power. We compare the computational effort of subsetting and re-gridding coverages via WCS to the extra delay caused by SUIS wrappers and slow network connections. Fig. 12 gives the average allocation of time cost of the SUIS steps after 100-times repeated tests on GADMFS WCS. The experiments request 23.3 Mbytes of VCI covering the California area of 647,972 square kilometers. Fig. 12 shows that it costs 9.2% of the total time to receive and parse the WCS capabilities document to initialize SUIS. Sending GetCoverage requests and downloading the VCI image only takes 90.7% of the time which is 1.03 seconds on average. Meanwhile, SUIS own operations cost barely any time or computational power (overall less than 1 millisecond). The service description retrieving takes some time cost due to the complex structure of the capabilities document which makes automatic parsing slow. We can improve it by exporting the corresponding SUIS driver state to a local file and read it back when scientists want to use that web service next time thus avoiding repeating the work of parsing the capabilities of that services. Recreating a SUIS driver from a configuration file is much faster than creating a new one from OGC capabilities document.

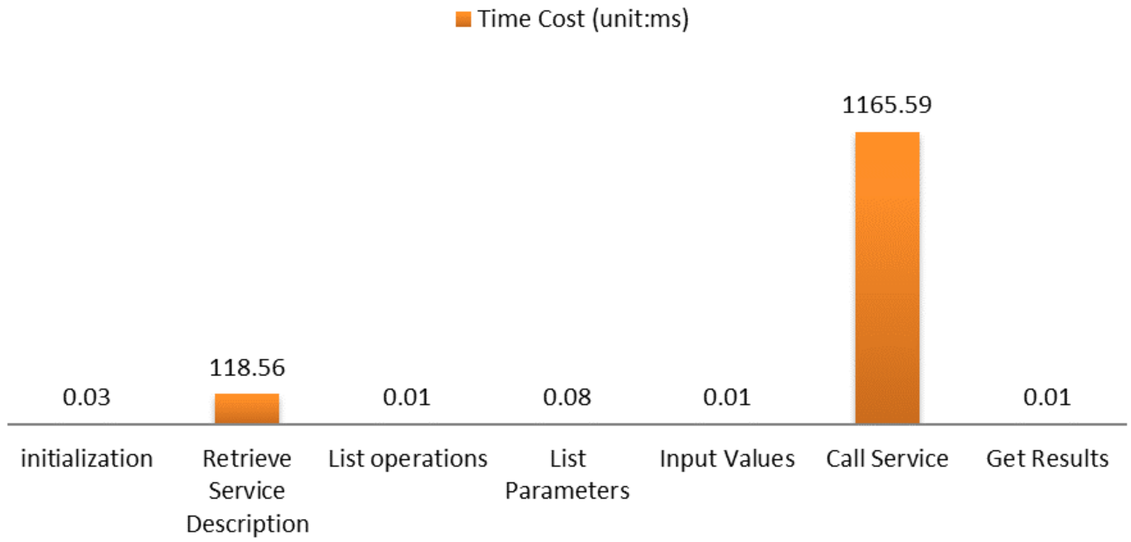


Figure 12. The average time cost of SUIS calling GADMFS WCS (SUIS own operations add negligible time costs)

The time cost of sending & receiving data will rise as the requested data becomes larger. Transmitting large binary datasets via web messages requires complex actions on both the server and the client. Protocols like SOAP allow multiple transmission options such as MTOM (W3C Message Transmission Optimization Mechanism), Base64, URL reference, FTP, etc.

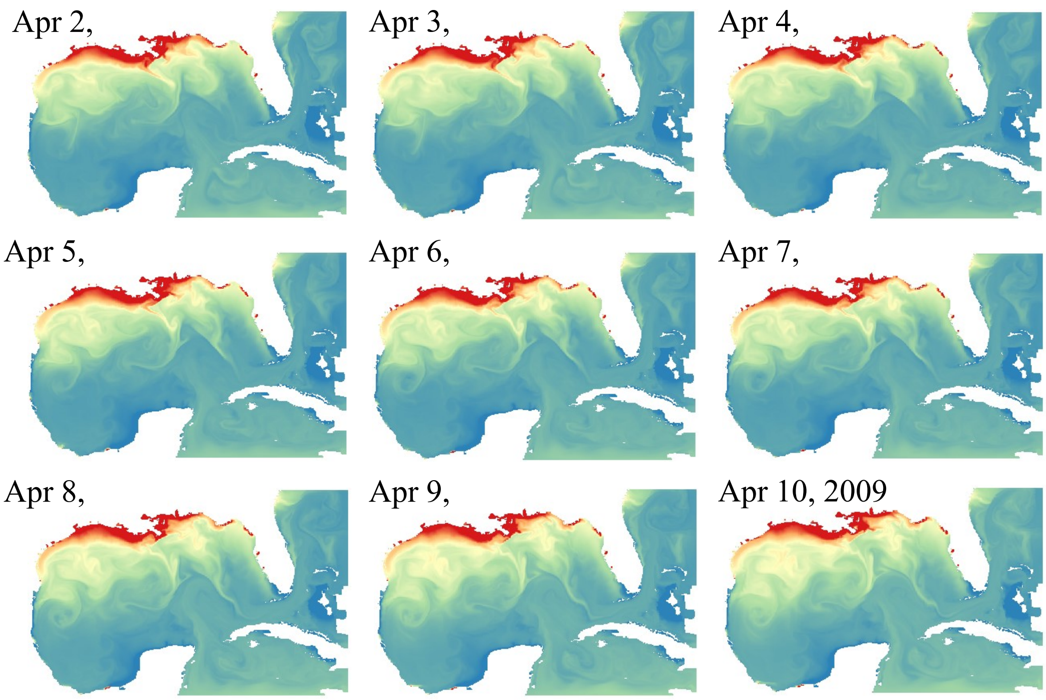
6.2 Coastal Ocean Modelling

To demonstrate that SUIS is a domain-independent tool, we also use suis4j in a coastal ocean modelling study based on FVCOM – an unstructured grid, finite-volume coastal ocean model (Chen et al., 2012). Our study area is the Gulf of Mexico and parts of the Atlantic Ocean. FVCOM requires input temperature and salinity data to be formatted into model-specific schemas. This data transformation task engages a substantial amount of oceanographers' time and they have voiced their need for automation of this work for a number of years. We excise suis4j to the preprocess water temperature and salinity data to use with FVCOM. A Java program[2] generating salinity condition grid to use as input for FVCOM was created and uploaded to GitHub to demonstrate another possible use of SUIS. This program uses services provided by the EarthCube CyberConnector project (Sun et al., 2017a). We access three services to download raw

[2] https://github.com/ZihengSun/suis4j/blob/master/src/suis4j/client/FVCOMTest.java

data, to interpolate it onto the FVCOM grid, and then finally to reformat it into a special model-ready format. suis4j invokes the three processes in sequence to produces a map of seawater salinity (Fig. 13). This experiment shows that SUIS enables instant automation to produce a time series of maps by making some minimal changes to the input parameters and rerunning the workflow sequence. This greatly relieves oceanographers from the repetitive, tedious and error-prone task of manually downloading and processing each dataset. Because SUIS vastly reduces the labor involved in using existing services, oceanographers are able to take advantage of EarthCube CyberConnector facilities that solve their specific data pre-processing problems. Without SUIS, these powerful facilities will remain under-utilized.

Besides the two case studies, we have actively engaged with our stakeholders in various communities including OGC, ESIP (Federation of Earth Science Information Partners), AGU (American Geophysical Union), and AMS (American Meteorological Society), and invited modellers and cyberinfrastructure developers to help test suis4j. We received some feedbacks which include many positive comments and also some suggestions for further improvements. Most of them confirm its necessity and simplicity, and supporting more languages such as python and R is the most priority thing for broad adoption.



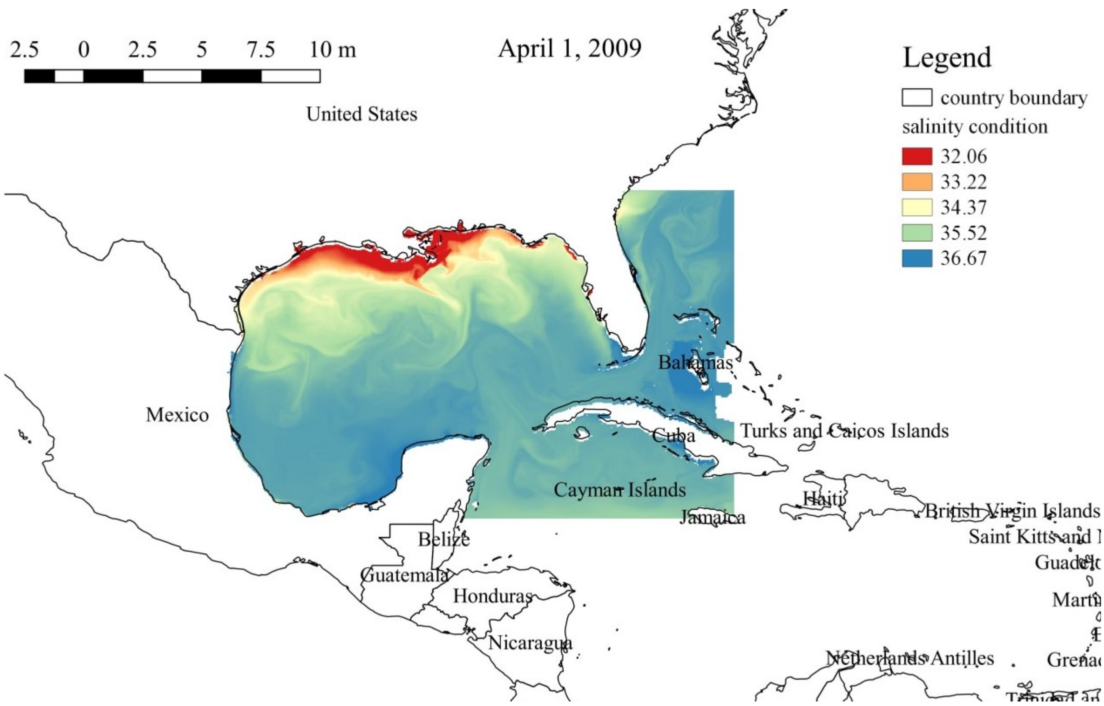


Figure 13. The result map of SUIS salinity workflow (generated by suis4j. The base layer is world country border.)

7. Discussion

This section discusses the advantages and disadvantages of SUIS from both engineering and scientific user's perspective.

7.1 Vendor Perspective

(1) *Scalability*: Scalability is strongly correlated with compatibility. SUIS has exceptional compatibility with existing GWS interfaces – it supports all generic GWS standards. SUIS framework is open and extensible – it is easy to create drivers to access service resources through new interfaces. One negative consequence of broad compatibility is that the greater variety of interfaces makes work to adapt all of them more complicated.

(2) *Interoperability*: The interoperability of a systems framework determines its level of flexibility and greatly impacts its future development (Thomas et al., 2007). SUIS supports two levels of interoperability: service and workflow. Service interoperability is provided by compatibility with the standard interfaces of geospatial web services. Workflow interoperability is supported through workflow language standard and workflow engine. SUIS workflows can be

translated to workflows in other workflow languages and systems like BPEL (Business Process Execution Language) (OASIS, 2007) or Taverna (Oinn et al., 2004).

(3) *Performance*: The resource overhead of SUIS own operation steps is small and negligible (Fig. 12). Most time cost within SUIS is spent on communicating with GWS – which is inevitable. The internal logic of SUIS does not incur significant time cost. The performance of SUIS applications is determined mainly by the network capacity, the client and server computational power and the workload.

7.2 Scientist Perspective

(1) *Simplicity*: SUIS is a clear lifesaver for users tired of interacting with varied and confusing web service interfaces. SUIS simplifies the calling procedures into a unified process which is easy to master for beginners. The disparate, unnecessary and complicated technical details are safely buried in the background.

(2) *Reliability*: SUIS will operate without interruption as long as the corresponding geospatial web service is up and running. SUIS itself won't interrupt the user logic unless it encounters a service-related exception and has to terminate the entire workflow. SUIS can run indefinitely without interruptions and suis4j library presents an easy and reliable introduction to all GWS.

(3) *Short learning curve*: SUIS exposes minimal little technical details and avoids obscure technical jargon in its API model and documentation. The terminology and concepts involved in understanding and using SUIS are as simple and understandable as possible. No technical knowledge of service details is required because SUIS separates its intuitive profile from the messy service binding details. As shown in Table 3, users are able to take advantage of the service without learning about service standards, web protocol, web service profiles, workflows, XML, etc. The GWS barrier of entry is substantially lowered by SUIS.

8. Conclusion

This paper proposes a novel framework called SUIS to simplify the usage of GWS in geospatial cyberinfrastructure, which has been under-utilized because of difficult and disparate interfaces. SUIS creates a universal profile for the major geospatial web service categories and builds a convenient bridge between the existing GWS and scientists in geospatial application domains. It severely decreases the complexity of using cyberinfrastructure service resources in and especially benefits scientists without GWS backgrounds. Simultaneously, the framework supports high scalability, interoperability and lower barriers of entry.

In the future, scientists from various communities will take advantage of SUIS to develop new scientific use cases. The SUIS workflow translation to standard workflow languages will be implemented. As snippets of knowledge, SUIS workflows can interconnect and form more advanced models to perform large and complex tasks such as global climate change simulation or global drought forecasting. We will continue to work on include SUIS in broader collaborative research that includes datasets and functionalities from a greater variety of sources and disciplines. Security and service documentation enhancement are another two important issues and will be studied in the next stage of work. In addition, SUIS drivers should enumerate and rank possible transmission protocols according to their network performances for a given volume of data and then select the most effective option. Dynamic selection of transmission channels can help SUIS adapt to different data volume scaling scenarios and choices of data formats. These methods can be utilized to reduce the time costs of the sending and receiving steps and avoid exceeding timeout limits or overloading the network infrastructure.

Acknowledgment

We sincerely thank the anonymous reviewers, the authors of the software, libraries, tools and datasets we have used in this work, and ESIP Lab. This study was supported by grants from the National Science Foundation (Grant number: AGS-1740693 & CNS-1739705; PI: Prof. Liping Di).

Disclosure

No interest conflict is claimed.

Reference

Allen, D.W., 2011. Getting to Know ArcGIS ModelBuilder. Esri Press.

Berman, F.D., Brady, H.E., 2005. NSF SBE-CISE workshop on cyberinfrastructure and the social sciences. National Science Foundation.

Bol, P.K., 1987. Seeking Common Ground: Han Literati under Jurchen Rule. Harvard Journal of Asiatic Studies 47(2) 461-538.

Botts, M., Percivall, G., Reed, C., Davidson, J., 2008. OGC® sensor web enablement: Overview and high level architecture, GeoSensor networks. Springer Berlin Heidelberg, pp. 175-190.

Burkoň, L., Generic Service Interface. prací účastník 12.

Chen, A., Di, L., Wei, Y., Bai, Y., Liu, Y., 2009. Use of grid computing for modeling virtual geospatial products. International Journal of Geographical Information Science 23(5) 581-604.

Chen, C., Beardsley, R.C., Cowles, G., 2006. An unstructured grid, finite-volume coastal ocean model (FVCOM) system. Oceanography 19(1) 78-89.

Chen, C., Beardsley, R.C., Cowles, G.W., Qi, J., Lai, Z., Gao, G., Stuebe, D.A., Xu, Q., Xue, P., Ge, J., 2012. An Unstructured-grid, Finite-volume Community Ocean Model: FVCOM User Manual. Sea Grant College Program, Massachusetts Institute of Technology.

Chinnici, R., Moreau, J.-J., Ryman, A., Weerawarana, S., 2007. Web services description language (wsdl) version 2.0 part 1: Core language, W3C recommendation. W3C, p. 19.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., 2001. Web services description language (WSDL) 1.1, W3C Standard, p. W3C Standard.

Clements, T., 2002. Overview of SOAP, Sun Develper Network.

Council, C., 2007. Cyberinfrastructure vision for 21st century discovery. National Science Foundation, Cyberinfrastructure Council.

Custer, A., 2011. GeoAPI 3.0 Implementation Standard. Open Geospatial Consortium (OGC).

David, P.A., 2004. Towards a cyberinfrastructure for enhanced scientific collaboration: providing its' soft'foundations may be the hardest part.

Davis Jr, C.A., Alves, L.L., 2005. Local Spatial Data Infrastructures Based on a Service-Oriented Architecture, GeoInfo, pp. 30-48.

de Souza Muñoz, M.E., De Giovanni, R., de Siqueira, M.F., Sutton, T., Brewer, P., Pereira, R.S., Canhos, D.A.L., Canhos, V.P., 2011. openModeller: a generic approach to species' potential distribution modelling. GeoInformatica 15(1) 111-135.

Demirkan, H., Delen, D., 2013. Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. Decision Support Systems 55(1) 412-421.

Deng, M., L. Di, W. Han, A. Yagci, C. Peng and Gil Heo, 2013. Web-service-based Monitoring and Analysis of Global Agricultural Drought. Photogrammetric Engineering & Remote Sensing (PE&RS) 79(10) 929-943.

Dhara, K.M., Dharmala, M., Sharma, C.K., 2015. A Survey Paper on Service Oriented Architecture Approach and Modern Web Services.

Di, L., 2003. The development of remote-sensing related standards at FGDC, OGC, and ISO TC 211, Geoscience and Remote Sensing Symposium, 2003. IGARSS'03. Proceedings. 2003 IEEE International. IEEE, pp. 643-647.

Di, L., 2004. GeoBrain-A Web Services based Geospatial Knowledge Building System, Proceedings of NASA Earth Science Technology Conference: Palo Alto, CA, USA, p. 8.
Di, L., Zhao, P., Yang, W., Yue, P., 2006. Ontology-driven automatic geospatial-processing modeling based on web-service chaining, Proceedings of the sixth annual NASA earth science technology conference. Citeseer, pp. 27-29.
Dietz, C., 2010. Geospatial Web Services, Open Standards, and Advances in Interoperability: A Selected, Annotated Bibliography. Coordinates Online Journal of the Map and Geography Round Table A(8).
FRISBIE, D., 1979. GEOPROCESSING FOR COMMUNITY CRIME PREVENTION PLANNING (FROM CRIMINAL JUSTICE INFORMATION AND STATISTICS SYSTEMS-INTERNATIONAL SEARCH SYMPOSIUM-PROCEEDINGS, FOURTH, 1979, BY JOHN W LAUCHER-NCJ-63648).
Gao, F., Yue, P., Zhang, C., Wang, M., 2019. Coupling components and services for integrated environmental modelling. Environmental Modelling & Software 118 14-22.
Glatard, T., Emsellem, D., Montagnat, J., 2006. Generic web service wrapper for efficient embedding of legacy codes in service-based workflows, Grid-Enabling Legacy Applications and Supporting End Users Workshop, pp. 1-10.
Goodchild, M., 1982. Accuracy and spatial resolution: critical dimensions for geoprocessing, In: Douglas, D., Boyle, A.R. (Eds.), Computer Aided Cartography and Geographic Information Processing: Hope and Realism: Ottawa, Canada, pp. 87-90.
Hadley, M.J., 2006. Web application description language (WADL).
Han, W., Di, L., Zhao, P., Wei, Y., Li, X., 2008. Design and implementation of GeoBrain online analysis system (GeOnAS), Web and Wireless Geographical Information Systems. Springer, pp. 27-36.
Hey, T., Tansley, S., Tolle, K.M., 2009. The fourth paradigm: data-intensive scientific discovery. Microsoft research Redmond, WA.
Hey, T., Trefethen, A.E., 2005. Cyberinfrastructure for e-Science. Science 308(5723) 817-821.
Hofer, B., 2013. Geospatial Cyberinfrastructure and Geoprocessing Web—A Review of Commonalities and Differences of E-Science Approaches. ISPRS International Journal of Geo-Information 2(3) 749-765.
Institute, E.S.R., 2001. What is ArcGIS?: GIS by ESRI. ESRI.
ISO/TC211, 2009. Standards Guide: ISO/TC 211 GEOGRAPHIC INFORMATION/GEOMATICS.
Jakeman, A.J., Letcher, R.A., 2003. Integrated assessment and modelling: features, principles and examples for catchment management. Environmental Modelling & Software 18(6) 491-501.
Josefsson, S., 2006. RFC4648: The Base16, Base32, and Base64 data encodings. Internet Engineering Task Force, Tech. Rep.
Kankanamge, C., 2012. Web services testing with soapUI. Packt Publishing Ltd.
Keens, S., 2007. Discussions, findings, and use of WPS in OWS-4. OGC Discussion Paper. OGC 06-182r1. Version 0.9. 1, 2007-05-10.
Kelbert, A., 2014. Science and cyberinfrastructure: The chicken and egg problem. Eos, Transactions American Geophysical Union 95(49) 458-459.
Kelly, R.A., Jakeman, A.J., Barreteau, O., Borsuk, M.E., ElSawah, S., Hamilton, S.H., Henriksen, H.J., Kuikka, S., Maier, H.R., Rizzoli, A.E., 2013. Selecting among five common modelling approaches for integrated environmental assessment and management. Environmental Modelling & Software 47 159-181.

Kiehle, C., Greve, K., Heier, C., 2006. Standardized geoprocessing–taking spatial data infrastructures one step further, Proceedings of the 9th AGILE International Conference on Geographic Information Science. Visegrád, Hungary.

Kim, S., Thiessen, P.A., Bolton, E.E., Bryant, S.H., 2015. PUG-SOAP and PUG-REST: web services for programmatic access to chemical information in PubChem. Nucleic acids research gkv396.

Kinzy, S., 1978. Geoprocessing System Planning. Data Resources and Requirements: Federal and. Local. Perspectives, Edited by Rolf R. Schmitt and Ronald E. Crellin, URISA 102-107.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J.B., Grout, J., Corlay, S., 2016. Jupyter Notebooks-a publishing format for reproducible computational workflows, ELPUB, pp. 87-90.

Kralidis, T., 2015. OWSLib documentation. Retrieved.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., 2013. Integrated environmental modeling: a vision and roadmap for the future. Environmental Modelling & Software 39 3-23.

Li, X., Di, L., Han, W., Zhao, P., Dadi, U., 2010. Sharing geoscience algorithms in a Web service-oriented environment (GRASS GIS example). Computers & Geosciences 36(8) 1060-1068.

Lopez-Pellicer, F.J., RenteríA-Agualimpia, W., Béjar, R., Muro-Medrano, P.R., Zarazaga-Soria, F.J., 2012. Availability of the OGC geoprocessing standard: March 2011 reality check. Computers & Geosciences 47 13-19.

Mackiewicz, R., 2006. The benefits of standardized Web services based on the IEC 61970 generic interface definition for electric utility control center application integration, Power Systems Conference and Exposition, 2006. PSCE'06. 2006 IEEE PES. IEEE, pp. 491-494.

Mark, D.M., 1979. Phenomenon-based data-structuring and digital terrain modeling. Geo-processing 1(1) 27-36.

Miller, F.P., Vandome, A.F., McBrewster, J., 2010. Apache Maven.

Mookerjee, M., Vieira, D., Chan, M.A., Gil, Y., Goodwin, C., Shipley, T.F., Tikoff, B., 2015. We need to talk: Facilitating communication between field-based geoscience and cyberinfrastructure communities. GSA TODAY 25(11).

OASIS, 2007. Web Services Business Process Execution Language Version 2.0.

OGC, 2007. OpenGIS® Web Processing Service. OGC: OGC Standard.

OGC, 2017. Testbed-12 Implementing Asynchronous Services Response Engineering Report, In: Pross, B. (Ed.), OGC 16-023r3.

OGC, I., 2016. OGC Standards and Supporting Documents.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., 2004. Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20(17) 3045-3054.

Ostensen, O.M., Smits, P.C., 2002. ISO/TC211: Standardisation of geographic information and geo-informatics, Geoscience and Remote Sensing Symposium, 2002. IGARSS '02. 2002 IEEE International, pp. 261-263.

Percivall, G., 2002. ISO 19119 and OGC Service architecture, FIG XXII International Congress: Washington, D.C. USA, pp. 1-12.

Richard, S.M., Pearthree, G., Aufdenkampe, A.K., Cutcher‐Gershenfeld, J., Daniels, M., Gomez, B., Kinkade, D., Percivall, G., 2014. Community‐Developed Geoscience Cyberinfrastructure. Eos, Transactions American Geophysical Union 95(20) 165-166.

Roberts, J.J., Best, B.D., Dunn, D.C., Treml, E.A., Halpin, P.N., 2010. Marine Geospatial Ecology Tools: An integrated framework for ecological geoprocessing with

ArcGIS, Python, R, MATLAB, and C++. Environmental Modelling & Software 25(10) 1197-1207.
Schindler, U., Diepenbroek, M., 2008. Generic XML-based framework for metadata portals. Computers & Geosciences 34(12) 1947-1955.
Shapiro, N.M., Campillo, M., Stehly, L., Ritzwoller, M.H., 2005. High-resolution surface-wave tomography from ambient seismic noise. Science 307(5715) 1615-1618.
Shen, D., Yu, G., Kou, Y., Nie, T., Zhao, Z., 2007. Resolving heterogeneity of Web-service composition in network manufacturing based on ontology. International Journal of Computer Integrated Manufacturing 20(2-3) 222-233.
Sun, Z., Di, L., Huang, H., Wu, X., Tong, D.Q., Zhang, C., Virgei, C., Fang, H., Yu, E., Tan, X., 2017a. CyberConnector: a service-oriented system for automatically tailoring multisource Earth observation data to feed Earth science models. Earth Science Informatics 11(1) 1-17.
Sun, Z., Di, L., Zhang, C., Fang, H., Yu, E., Lin, L., Tan, X., Guo, L., Chen, Z., Yue, P., 2017b. Establish cyberinfrastructure to facilitate agricultural drought monitoring, Agro-Geoinformatics, 2017 6th International Conference on. IEEE, pp. 1-4.
Sun, Z., Peng, C., Deng, M., Chen, A., Yue, P., Fang, H., Di, L., 2014. Automation of Customized and Near-Real-Time Vegetation Condition Index Generation Through Cyberinfrastructure-Based Geoprocessing Workflows. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 7(11) 4512-4522.
Sun, Z., Yue, P., Di, L., 2012. GeoPWTManager: a task-oriented web geoprocessing system. Computers & Geosciences 47(0) 34-45.
Swain, N.R., Latu, K., Christensen, S.D., Jones, N.L., Nelson, E.J., Ames, D.P., Williams, G.P., 2015. A review of open source software solutions for developing water resources web applications. Environmental Modelling & Software 67 108-117.
Team, Q.D., 2013. QGIS geographic information system. Open Source Geospatial Foundation Project.
Thomas, D., Khalsa, S., Nativi, S., Ahern, T., Shibasaki, R., 2007. Processes for achieving interoperability in GEOSS, AGU Fall Meeting Abstracts, p. 08.
Trabant, C., Ahern, T., Stults, M., 2015. Building web service interfaces to geoscience data sets: EarthCube GeoWS project activities at the IRIS DMC, AGU Fall Meeting Abstracts.
Vitolo, C., Elkhatib, Y., Reusser, D., Macleod, C.J., Buytaert, W., 2015. Web technologies for environmental Big Data. Environmental Modelling & Software 63 185-198.
W3C, 2015. Web of Services.
Wagemann, J., Clements, O., Marco Figuera, R., Rossi, A.P., Mantovani, S., 2018. Geospatial web services pave new ways for server-based on-demand access and processing of Big Earth Data. International Journal of Digital Earth 11(1) 7-25.
Whiteside, A., Greenwood, J., 2010. OGC Web Services Common Standard. Open Geospatial Consortium.
WMS, O., 2004. Web Map Service. Version.
Wright, D.J., Wang, S., 2011. The emergence of spatial cyberinfrastructure. Proceedings of the National Academy of Sciences 108(14) 5488-5491.
Yu, G., Zhao, P., Di, L., Chen, A., Deng, M., Bai, Y., 2012. BPELPower-A BPEL execution engine for geospatial web services. Computers & Geosciences 47(0) 87-101.
Yue, P., Gong, J., Di, L., Yuan, J., Sun, L., Sun, Z., Wang, Q., 2010. GeoPW: Laying Blocks for the Geospatial Processing Web. Transactions in GIS 14(6) 755-772.

Yue, P., Zhang, M., Tan, Z., 2015. A geoprocessing workflow system for environmental monitoring and integrated modelling. Environmental Modelling & Software 69 128-140.

Tables

Table 1. The popular online geospatial cyberinfrastructures

Name	Searchable	Object	Server Interface	Portal	Provider
CWIC	✓	Data	CSW/OpenSearch	http://cwic.wgiss.ceos.org	CEOS
Unidata	x	Data	TDS	http://thredds.ucar.edu	UCAR
EOS	x	Data	HTTP	http://eospso.nasa.gov/	NASA
GCMD	✓	Data & GWS	HTTP	http://gcmd.nasa.gov/	NASA
GEOSS	✓	Data & GWS	CSW	http://www.geossregistries.info	GEO
U.S. Water	✓	Data	HTTP	http://water.usgs.gov	USGS
USGS Catalog	✓	Data	CKAN	https://data.usgs.gov	USGS
Data.gov	✓	Data & GWS	CSW/CKAN	https://data.gov	GSA
NOAA Catalog	✓	Data	CKAN	https://data.noaa.gov	NOAA
NCEI Ocean Archives	✓	Data	TDS/HTTP/FTP/DAP	http://data.nodc.noaa.gov/geoportal	NOAA
AWS Public Datasets	x	Data	HTTP	https://aws.amazon.com/datasets/	Amazon
FGDC Catalog	✓	Data	CKAN	https://cms.geoplatform.gov/data/	FGDC

Table 2. Data type mapping between GIS and SUIS

GIS	SUIS
Boolean	Bool
Short Integer	Number
Long Integer	Number
Float	Number
Double	Number
Text	String
Date	Date
BLOB	File
Object Id	String
Vector	String/File

Raster	String/File

Table 3. An example of SUIS invoking IRIS REST service

```
//Step 1
SUISClient sc = new SUISClient.Builder()
   .initialize("https://service.iris.edu/irisws/timeseries/1/application.wadl",
ServiceType.REST)
           .build();
//Step 2
sc.listOperations(); //optional
Operation o = sc.operation("http://service.iris.edu/timeseries/1/version.GET");
//Step 3 - optional
sc.listInputParams(o);
sc.listOutputParams(o);
//Step 4
o.input().value("network", "IU")
           .value("station", "ANMO")
           .value("location", "00")
           .value("channel", "BHZ")
           .value("starttime", "2001-12-09T12:00:00")
           .value("endtime", "2001-12-09T12:20:00")
           .value("output", "plot");
//Step 5
sc.call(o);
//Step 6 - optional
sc.listOutputValues(o);
String filepath = o.output().value("return");//get the data location
```

Table 4. suis4j dependencies

Library name	Functionality
SoapUI (Kankanamge,	Composing SOAP requests

2012)	
JAXB	Parsing XML schemas
XMLBean	Parsing XML schemas
WSDL4J	Parsing WSDL
GeoTools Java Toolkit	OGC standard schema API

Figures