International Journal of Digital Earth

ISSN: 1753-8947 (Print) 1753-8955 (Online) Journal homepage: http://www.tandfonline.com/loi/tjde20

Cyber-applications as gateway to data-rich Digital Earth systems

Javier Morales & Rolf A. de By

To cite this article: Javier Morales & Rolf A. de By (2014) Cyber-applications as gateway to data-rich Digital Earth systems, International Journal of Digital Earth, 7:7, 576-593, DOI: 10.1080/17538947.2013.788224

To link to this article: https://doi.org/10.1080/17538947.2013.788224

Accepted author version posted online: 21 Mar 2013.
Published online: 23 Apr 2013.

International Journal of Digital Earth, 2014
Vol. 7, No. 7, 576–593, http://dx.doi.org/10.1080/17538947.2013.788224

Taylor & Francis
Taylor & Francis Group

Cyber-applications as gateway to data-rich Digital Earth systems

Javier Morales* and Rolf A. de By

Department of Geo-Information Processing, Faculty of Geo-Information Science and Earth Observation, University of Twente, Enschede, the Netherlands

(*Received 18 July 2012; final version received 17 March 2013*)

The underlying vision of the Digital Earth (DE) calls for applications that can embed vast quantities of geo-referenced data and allow users to study and analyse of our planet. Since the declaration of this vision in the late 90s, a significant number of DE data-sets have been created by the industry, governments, non-governmental organisations and individuals. An overwhelming majority of the successful applications that use DE data-sets has its end-user applications running on the desktop. While these applications are great tools, they remain inaccessible to the community as a whole. In this paper, we present a framework for the development of cyber-applications. We define an abstract architecture for cyber-applications based on the model-view-controller paradigm, which allows the dynamic inclusion of functional and data components into its execution engine at run-time. We define the operational characteristics of cyber-applications. We also specify the interface of pluggable components to the architecture. Finally, we demonstrate the appropriateness of the abstract architecture by means of a case study.

Keywords: digital earth architecture; cyber-application; geobrowser; design framework; hardware independence

1. Introduction

The underlying vision of the Digital Earth (DE) calls for applications that can embed vast quantities of geo-referenced data and allow users to study and analyse our planet. Since the declaration of this vision in the late 90s, a significant number of DE data-sets have been created by the industry, governments, non-governmental organisations and individuals. DE data-sets encompass data about all phenomena that are of interest to humans and important or useful to monitor, and for which different information sources may be exploited. Such data typically and often involve location specifics. Developments in information and communication technology in recent years have resulted in significant changes in the way DE data-sets are collected, disseminated and exploited. First, standards on data storage and exchange formats have become popular and widely used, making most data-sets potentially exploitable by a wide range of applications. Secondly, data accessibility has improved by the implementation of application program interfaces (APIs), allowing data access via REST or SOAP service calls. Thirdly, data collection is no longer the prerogative of experts in a given field, but it has opened up so that other stakeholders, including individuals, can create or update data-sets as part of their

*Corresponding author. Email: j.morales@utwente.nl

day-to-day activity (Poore 2011). Lastly, data, these days, can be produced in near real-time, thanks to the widespread availability of sensors capable of registering a variety of *in situ* parameters.

Scientists, professionals and non-professionals are exploiting these developments and have started to generate spatial data in volumes that were unimaginable only a few years ago. It can be argued that we now have available data on all essential aspects of the physical and social environments in which we live, bringing us closer to a true digital earth (Craglia et al. 2012). Relevant data-sets produced either by institutions, communities or individuals can be found in a variety of themes: Finance & Budgeting, Transportation and Mobility, Education and Communication, Agriculture, Fisheries, Forestry, Economy & Industry, Environment, Population, and Health. A number of data repositories, also known as geo-portals, have been created to advertise and openly share such data-sets at the regional, national and local levels. These data repositories in general provide functionality to search and retrieve data, and in some cases some simple functions such as clipping and overlaying. Examples of data repositories include Europe's public data (Pulicdata EU 2012), the World Bank's data catalogue (World Bank 2012), Google's public data directory (Google 2012), OSGeo's public geospatial data project (OSGeo 2012), the GEOSS Common Infrastructure (GEOSS 2012), government portals like the UK Government's official open data portal (UKGDP 1979), the Open data portal of the Dutch government (Data Overheid 2012), and the Natural Earth public domain map data-set (Natural Earth 2012). In addition to thematic data, sources of base data also exist, for instance as provided by the OpenStreetMap initiative (OSM 2012), the GeoNames geographical database (GeoNames 2012), the Dutch public services on the map (PDOK 2012), the UK's OS OpenData portal (OS OpenData 2012), and the European Commission geoportal – INSPIRE (INSPIRE 2007). This is certainly not an exhaustive list, but it highlights the overwhelming scope, and thematic spread, of data available today.

Given the flood application case presented in Section 5, a relevant set of more specific data repositories include The Shuttle Radar Topography Mission (SRTM), which provides high-resolution digital elevation data on a near-global scale (SRTM 2012). Two basic elevation data-sets are available: elevation data sampled at one arc-second of latitude and longitude (SRTM1), and elevation data sampled at three arc-seconds (SRTM3). The Global Monitoring for Environment and Security program (GMES 2007) provides data from space and *in situ* sensors and covers six different themes: land management, marine environment, the atmosphere, emergency response, civil security and climate change. Meteorological data-sets are available via the European Climate Assessment & Dataset project (ECA&D 2012). European Climate Assessment disseminates two types of data: daily observations of 12 meteorological elements from stations around Europe and the Mediterranean (56 countries in total), and a gridded data-set containing precipitation, temperature and sea level pressure, known as the E-OBS data-set.

The most recent initiatives on provision of data-sets have incorporated APIs as a mechanism of data access. This greatly facilitates data exploitation as it enables software components to directly interact with the data. The most popular API among national and regional governments, companies and organisations is provided by the CKAN, the Comprehensive Knowledge Archive Network. The open-source data portal platform (CKAN) was originally developed by the non-profit Open

Knowledge Foundation to run a public registry of open knowledge data-sets (CKAN 2012). It is now widely used to run data portals including the European Union's public data portal (Pulicdata EU 2012), the UK Government's open data portal (UKGDP 1979) and the Dutch public data portal Data Overheid (2012). CKAN provides a REST-based API that can be used to retrieve, create and update data resources. Through this API, applications can programmatically access data from a variety of catalogues in popular formats such as CSV and JSON JSON (2002). Although a large number of data-sets exist, it remains a challenge to achieve high levels of integration between these data sources.

Interestingly, in spite of the advances in connectivity that we witness today, generally users are still expected to either order, ship or download desired (integral) data-sets into their local systems (Yang et al. 2011) or to simply interact with the data through predetermined ('canned') visualisations. The second option is especially prominent for spatial data-sets, for which users are always expected to use rather simplistic and tightly-coupled data viewers, where data and representation are combined into a single product.

One of the remaining hurdles, therefore, is in the creation of the fundamental ability to exploit, in real-time, the available data. For the most part, all existing data-sets are tightly coupled to specific user interfaces. This is only natural as content has traditionally been created with a target application in mind, and also because developers aim at creating applications that maximise the exploitability of well-known data-sets. Modern and flexible approaches to application engineering should dynamically create access to, and operation on disparate data-sets to address requirements unknown to the developers at design time. Such applications become possible when the required data and function components can themselves be harvested from the underlying infrastructure. We call these applications *cyber-applications*. A cyber-application, or cyber-app for short, is a data-intensive application that runs in a web environment or as a smart-app, and is capable of dynamically incorporating new data and functions into its execution engine.

This type of application appeals to use cases in and around the DE vision (Goodchild 2008). Consider a mature mobile application that is developed to inform about the risk of flooding and flood events, to be used anywhere in Europe by citizens like home owners and buyers, but also holiday-makers, by professionals in departments of public works, water boards and other governmental agencies, and also by professionals in the private sector, for instance, affiliated with the real estate sector or insurance companies. The application addresses floods of four types: river flooding, coastal flooding, surface water flooding and flash floods, and aims to inform both about risk from the historic perspective, as well as from the current perspective. An application of this kind needs to rely heavily on external sources, especially *data resources*, *computational models*, *metadata* about the first two, and since the computational models are often too data- and time-complex to run on a local device, also *compute resources* (i.e. compute servers elsewhere).

Even though INSPIRE's delivery of data across Europe aims to standardise many of the needed data resources, the application witnesses a highly diverse landscape over the spatial extent that it intends to serve: different locations require different constellations of models and input data. Data-sets are universally partitioned based on artificial administrative boundaries and institutional themes. User-generated data may be virtually absent in one place and be highly detailed and

up to date in another. For some rivers, level gauge data may be in the open domain, and ready to be used, while elsewhere such data may not be available at all. The heterogeneity in this data landscape directly affects the applicability of available flood models, and this is the reason why an extremely important function of our mobile flooding application is the strategic analysis and determination of the most appropriate models and input data-sets.

The creation of cyber-apps requires a robust design method that exploits service principles and applies a clear separation of concerns over content, visualisation and processing functions. To address these needs, we present an approach to cyber-app design based on the model-view-controller (MVC) paradigm (Trygve 1979).

The remainder of the paper is organised as follows. Section 2 reviews the principles of web application design; Section 3 discusses the rationale for and the usage of the MVC pattern; Section 4 discusses our proposal for a basic architecture of cyber-apps; Section 5 puts forward an example implementation and Section 6 concludes with a short discussion on key priorities.

2. Web application development

Web applications come in a variety of types and levels of complexity, from simple informational web sites to elaborate sites that try to mimic desktop applications. The evolution of web applications reflects the vision and context of the creating organisation and the available technology at the time of development, which means that there is a significant correlation between the chronology and type of application Kappel et al. (2006). Given their scope and complexity, web applications can be chronologically categorised as document-centric, transactional, workflow, portal-oriented, social (Web 2.0) and cyber-applications.

Document-centric applications allow users to retrieve static HTML documents upon request. The content of these documents typically does not require frequent editorial maintenance, and is often redundantly available at other locations. Examples include web sites of small and middle-sized enterprises, tutorials, intranets and webcasts. The response time of these applications is short, reflecting document size, and the presentation is often simple. *Transactional* applications are more involved and commonly contain forms, menus and buttons that allow users to retrieve or manipulate content by interaction. This type of application commonly involves a database back-end, used to dynamically generate content based on end-user interaction. Users can not only perform read-only operations, but also modify the underlying content. Examples of this type are travel planners, booking systems, online banking and shopping applications. *Workflow* applications execute business processes, within or between organisations, which involve user interaction. Workflow participants may include private companies, public authorities, their employees and other individuals. The objective of these applications is to streamline business processes and time involvement with customers, which makes them popular in the area of public administration. Examples include applications for permit requests, e-government, property transactions and mobile offices. Workflow applications are rather complex and face challenges as they require implementation of interoperable services, process monitoring and secure identification.

Social media applications, also known as Web 2.0 applications, focus on cooperation and sharing of content. Contrary to workflow applications, they operate

in unstructured operational environments. The main focus is to guarantee appropriate communication between participants. Content is generated by participants in collaboration, and the focus of content generation changes rather frequently. Contrary to the anonymity that originally characterised the Web, collaborative applications strive for identity sharing among users with similar interests. Collaborative applications include all the social networking sites, such as Twitter, Flickr and Foursquare. An interesting side-effect of collaborative applications is their potential to exploit generated content in all sorts of human behavioural studies. One such example is Livehoods (Livehoods 2012), in which analysis determines the nature of urban areas by the types of people who visit those areas. Livehoods harvests data from the social network Foursquare, and uses it to study the dynamics of cities (Cranshaw et al. 2012). *Portal-oriented* applications provide a centralised access to heterogeneous data resources. The most common use for them is the search for information resources. These applications vary from general purpose to specific themes, such as entertainment, information technology, health, or geographical data. This type of application includes search engines, and data clearinghouses.

Cyber-apps are one of the latest developments in the evolutionary path of the Web and its applications. The fundamental principle of cyber-apps is the seamless exploitation of interconnected resources to support application goals, whether in research, education or business activities. Exploitable resources may be computational services, computing platforms, network systems, data repositories, sensors or skills. Cyber-apps are not executed over the Web as we know it today. Instead, they are part of an execution platform known as a CyberInfrastructure (CI). The term 'CyberInfrastructure' was initially used by the United States National Science Foundation (NSF) to refer to an infrastructure based on distributed computing, information and communication technology (Atkins et al. 2003), which supports advanced data acquisition, storage, and management, and also provides advanced information integration and processing services that allow the derivation of new scientific theories to help modern society progress. On this principle, a CI focuses on efficiently connecting laboratories, data, computers and researchers to facilitate the creation of novel scientific theories and knowledge (Brodaric et al. 2009, Rod Blais and Esche 2008, Darema 2010, Wright and Wang 2011, Yang et al. 2010). Cyber-apps correspond to a layer of functionality that sits on top of the CI, to enable exploitation of resources in specific communities, projects and disciplines. In Section 3 we introduce the concept of cyber-apps and explain the level of functionality they provide.

Given the various types of web application and the continuous change in the supporting technology and user requirements, it is important to adopt a sound design method that minimises the impact of this change Esclaona and Koch (2004), Mendes et al. (2006). For the case of cyber-apps, the ideal approach must be based on models and transformations so that application components are flexible, and can be easily adapted, i.e. redesigned, to changing conditions.

Development approaches for web applications have been available for a long time, providing developers with steps to address the development process properly. A somewhat comprehensive list of such approaches was discussed by Fatolahi et al. (2012). Even today, only a few of these methods are actually used by web developers. The Web Site Design Method (WSDM) focuses on sites for the presentation of organisational data, based on a so-called user-centric approach (De Troyer and

Leune 1998). The WSDM method applies three design phases: user modelling, conceptual design and implementation. The result is an appropriate navigation structure through web pages (viewpoints) that present content relevant for specific user classes. The W2000 Baresi et al. (2006) method uses transformational models as its primary design tool, and provides an extension to UML notation for modelling web page elements. This method focuses on the identification of information content from user requirements, the definition of operations needed to appropriately present content and finally the generation of navigation structure over the information content. The W2000 method applies a metamodel and follows three design phases: requirements analysis, hypermedia design and service design.

The Web Modelling Language (WebML) is a specification language defined to develop hypermedia applications. It provides constructs to represent sets of linked hypertext pages encompassing content units and operations. This approach offers developers a way to organise a web site as a modular arrangement of content areas represented as site views (Ceri et al. 2003).

The UML-based Web Engineering (UWE) method is an approach to develop web applications with the Unified Process (UP). UWE Koch et al. (2008) specifies three views for the design of web applications: presentation, navigation structure and content. Development along these views follows the UP cycle. One special characteristic of UWE is its emphasis on the use of models and transformations, to allow fully automated generation of web systems from models.

Other web design methods such as Object-Oriented Hypermedia Design Method (OOHDM), Relationship Navigational Analysis (RNA), Navigational Development Techniques (NDT) and Scenario-Based Object-Oriented Hypermedia Design Methodology (SOHDM) focus more on the navigation aspects of web sites, which is critical in standard web sites, but not so central for cyber-apps.

The majority of web design methods concentrate on the design of the navigation of the application because most web applications consist of interlinked documents or web pages that users can traverse. The interlinked documents arise from data perspectives identified by the developers, derived from user requirements. Tracking the state of the navigation is also an important requirement, and requires proper management of navigation history. A cyber-app, in contrast, relies heavily on the manipulation of user events, which may result in the generation of data, or asynchronous access to a new data source.

3. Designing cyber-apps

We define a cyber-app as a highly interactive Web application that allows users to perform tasks within a certain problem domain, e.g. determine the flooding event impact on humans, and that provides flexibility to manipulate the application's internal computational model, and to add external data resources that might be exploitable by some task. Both computational and data resources can be obtained from the underlying CI. Cyber-apps are built using the latest Web technology, such as HTTP, HTML5, CSS3 and JavaScript, to maximise reach and compatibility over multiple computing systems.

Traditionally, the Web has been used as a medium to deliver content, mostly in the form of HTML documents. However, thanks to developments in Web technology it is now possible to use the Web for the realisation of complex tasks, which have

until now been the target of desktop applications. Web applications are therefore rapidly becoming commonplace, varying from email management, route finding, word processing to content management Governor et al. (2009). In spite of their growing popularity there are still critics, especially from the side of usability (Gitzel et al. 2007, Nielsen 2007, 2009), who doubt their potential as they add complexity, from the usability point of view, for both users and developers. For this reason, proper complexity management is a key factor to turn these apps into well-established societal tools. This is especially important since many of the development projects during the initial wave of Web applications did not achieve great success (Díaz and Aedo 2007).

From the design perspective, the challenges for the creation of cyber-apps include: (1) a method to guide the design process, (2) an underlying architecture that supports the dynamic and modular structure of cyber-apps and (3) the selection of a typical application paradigm with the appropriate primitives to assemble disparate services into an application. In this paper, we present a systematic approach to the design and operation of cyber-apps. We first describe the design principles, and then the design steps.

Our approach uses the principle of application partitioning into a set of manageable application modules, called appmods, and their corresponding interactions. To illustrate this partitioning principle, imagine a mobile application designed to deliver updates on the conditions of a river during a flooding event. The application aims at capturing photos in real-time, depicting river conditions at key locations. Such information is used to validate the flood forecasts, or to document the conditions reflecting a normal state of a river, for instance, for insurance claim purposes. The application can be divided into a primary appmod responsible for decision logic, and two secondary appmods, one responsible for positioning and one for sensor readings (in this case: photo collection). The primary appmod receives location criteria from an external source. The positioning appmod continuously tracks location to be compared with the criteria obtained by the primary appmod. When a match is made between location criteria and the location reported by the positioning appmod, the primary appmod makes a request to the sensor appmod to take the corresponding photo. The primary appmod collects the photos and delivers the results after locations have been surveyed. In this simple example, we see that the primary appmod handles the computational model and operates independently from the data sources. We also see that the secondary appmods use some underlying technology (e.g. a GPS unit or a camera, perhaps with GPS on board) to obtain data. The application resides in a cyber-node, which is a network-accessible system with computing capabilities, volatile or persistent storage capabilities, sensing capabilities or any combination of these. Although a cyber-node must exhibit one or more of these capabilities, we make no assumptions with respect to the actual level of computing power, storage capacity or communication capabilities of a cyber-node.

This application perspective allows us to state that a cyber-app is made up of a set of appmods with a clearly defined set of interactions between them. Every individual module exists at a given cyber-node in the underlying CI. Consequently, the three constituents of a cyber-app are cyber-nodes, appmods and interaction topology (see Figure 1). We can also argue that because this application is a cyber-app, the partitioned appmods need not be running on the same cyber-node, and as such we can physically separate the primary appmod from the secondary appmods over

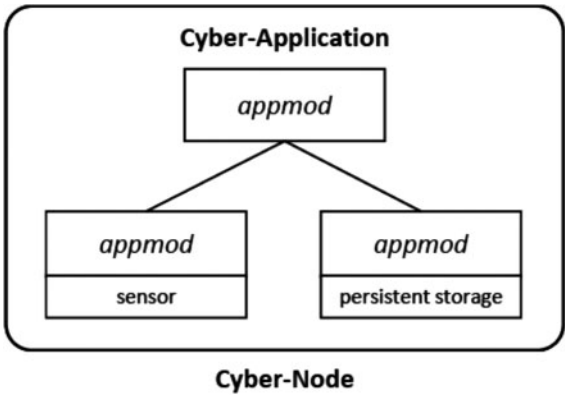

Figure 1. Structure of a cyber-application.

multiple cyber-nodes. A standardised theory for designing collaborative cyber-nodes can be found in de By et al. (2009).

The internal structure of a cyber-app consists of appmods. These modules encapsulate the application logic of the cyber-app and are the basis of the application's modularity and extensibility. For the identification of appmods, our approach uses the Model-View-Controller design pattern (MVC), whereby an appmod is either an M, a V or a C appmod. The MVC design approach Burbeck (1992), Hasan and Isaac (2011), Krasner and Pope (1988), Leff and Rayfield (2001), Trygve (1979) prescribes three different concerns in an application. Model elements take care of the information content of the application. View elements take care of presenting the contents of the model according to user needs; the decision as to which view element to use in a given case falls under the responsibility of controller elements. Controller elements realise the application behaviour by interpreting user-generated events, mapping them to specific actions (on model elements) and reporting back to the user through view elements. Controllers, if necessary, also manage the remote communication with other elements via HTTP requests. MVC is particularly suitable for the design of cyber-apps because it naturally enables the organisation and structuring of sets of interdependent appmods consistently based on their roles in the application. MVC also allows to expose the dependencies between appmods, making it possible to reliably substitute them or add new ones based on the specific requirements of the application. This characteristic is particularly useful when it comes to the incorporation of new (user) data-sets in a cyber app. Thus, MVC provides us with the flexibility needed so that cyber-apps can support complex applications in a web environment. An alternative design approach to MVC in this case is the widely popular WebForms approach (Microsoft 2012); however, for our objective it has some disadvantages. For example, WebForms controllers encapsulate the presentation, binding it with the data to create a combined view, for which the state is preserved. While this guarantees consistency in the presentation of the data, it makes it difficult to use an arbitrary representation structure, to combine multiple representations or to dynamically assign presentation structures to the data.

The interaction structure between appmods follows an event-based paradigm, in which actions take place in a sequential manner, and this sequence includes only

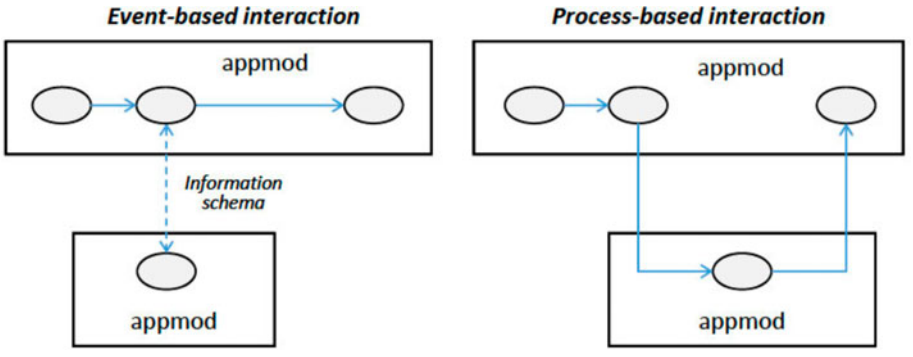

Figure 2. Event-based vs. process-based interaction.

actions that emerge as a direct result of the triggering event. An event here includes a user gesture, the arrival of a message, or the completion of a processing task. This means that there is no process notion in the interaction between appmods. This approach to the interaction between appmods brings two important benefits: first, it forces the designer to separate the required action (the action needed in response to the event) from the application module that eventually executes it. For example, an application module could issue a request, in response to an event, for a sensor reading (e.g. to determine location) and the actual reading could be handled by a different appmod, either a known appmod or one that is added to the cyber-app during run-time, which could be needed for locations about which the cyber-app does not have knowledge of such a sensor. Secondly, it also guarantees that the interaction stays local with respect to the participating appmods and does not form part of the process logic of the cyber-app. In the case of the sensor reading request, the appmod responsible for the sensor reading does not form part of the application workflow as would be the case in a process-based interaction. The difference is illustrated in Figure 2. The exchange of data between the primary and secondary appmods is based on a dedicated part of the cyber-app information schema.

With the design principles in place, we can now concentrate on the systematic approach to develop a cyber-app. The design process starts with the identification of data requirements and their formalisation by means of an information schema. That information schema captures the semantics of the problem area addressed by the application. The information schema has to be developed by a domain expert, who first, understands the application domain, and secondly, is knowledgeable on data modelling. The resulting schema must properly cover the application and be expressed with a data structure that can be realised using separate data-sets (see Listing 1).

Next, the appropriate data sources are identified from the pool of open data available in the CI (see Section 1). This identification also meets location-based criteria so that suitable data-sets are found for the various geographical regions covered by the cyber-app. As a consequence, more than one data-set could be associated with a single element of the information schema.

Then, the information schema has to be divided into a set of M-type appmods. This division process should be carried out in such a way that the resulting appmods

```
Ext.define('FloodApp.model.Obstacle', {
  extend: 'Ext.data.Model',
  fields: [
    { name: 'class', type: 'string' },
    { name: 'geom', type: 'geometry' },
    { name: 'roughness', type: 'real' },
    { name: 'code', type: 'string' },
    { name: 'layer', type: 'object' }
  ],
  idProperty: 'code'
});
Ext.define('FloodApp.store.Obstacle', {
  extend: 'Ext.data.Store',
  model: 'FloodApp.model.Obstacle',
  proxy: {
    type: 'ajax',
    api: {
      read: 'DFS/obstacles.json',
      update: 'DFS/users/obstacles.json'
    },
    reader: {
      type: 'json',
      root: 'obstacles',
      successProperty: 'success'
    }
  }
});
```

Listing 1. Obstacles information model.

support two objectives: allow for independent data exchange between interacting appmods and facilitate the computational tasks of the cyber-app. This step is revisited iteratively at the end of each of the remaining steps so that the two objectives of this division are properly supported.

Subsequently, the application events that may take place during execution are listed. This list has to be as exhaustive as possible since it is the basis for the structuring of communication patterns between appmods.

Next, the application workflow is specified, and it contains all the actions that are needed to realise the functional requirements of the cyber-app. The responsibility over the application logic is assigned to a specific C-type appmod (see Section 4), while the actions themselves are assigned to other C- or V-type appmods. The assignment of actions to appmods is made using the list of events that was previously identified. This results in a clear interaction structure between appmods. At this point, the partition of the information schema is revisited to verify its support for the interaction between the chosen appmods.

Finally, based on the reference architecture of cyber-apps, the distribution of work between permanent and volatile appmods is specified (we discuss this further in Section 4).

4. Cyber-app architecture

To achieve an adaptable architecture, the internal structure of a cyber-app consists of permanent and volatile appmods. Permanent appmods are those that form the backbone of the cyber-app and are common to all cyber-apps. Permanent appmods

carry the logic of the application, grant access to data resources and perform mediator functions between permanent and volatile appmods. Volatile appmods are those that are dynamically included in the cyber-app during run-time. Volatile modules are exclusively third-party data and computational services and are the result of the application development process. Since our design method follows the MVC paradigm, permanent and volatile appmods need to fall within one of the MVC types. In this context, we call M-type appmods information models, V-type appmods views and C-type appmods controllers. A cyber-app contains five essential controllers that are common to all cyber-apps (see Figure 3):

- data feed service (DFS),
- data interpreter (DI),
- module loader (ML),
- processing engine (PE), and
- core engine (CE).

The *data feed service (DFS)* appmod runs on the server as a proxy that sits between third-party data sources and cyber-apps. The role of the DFS is to retrieve the data and create data feeds that are compatible with one of the cyber-app information models. For the manipulation of these data feeds, the DFS uses the concept of *foreign tables*. This concept is based on the SQL/MED standard specification (Melton et al. 2002), which defines the mechanisms to access and operate on data that resides outside a database management system, while using regular SQL to operate on the

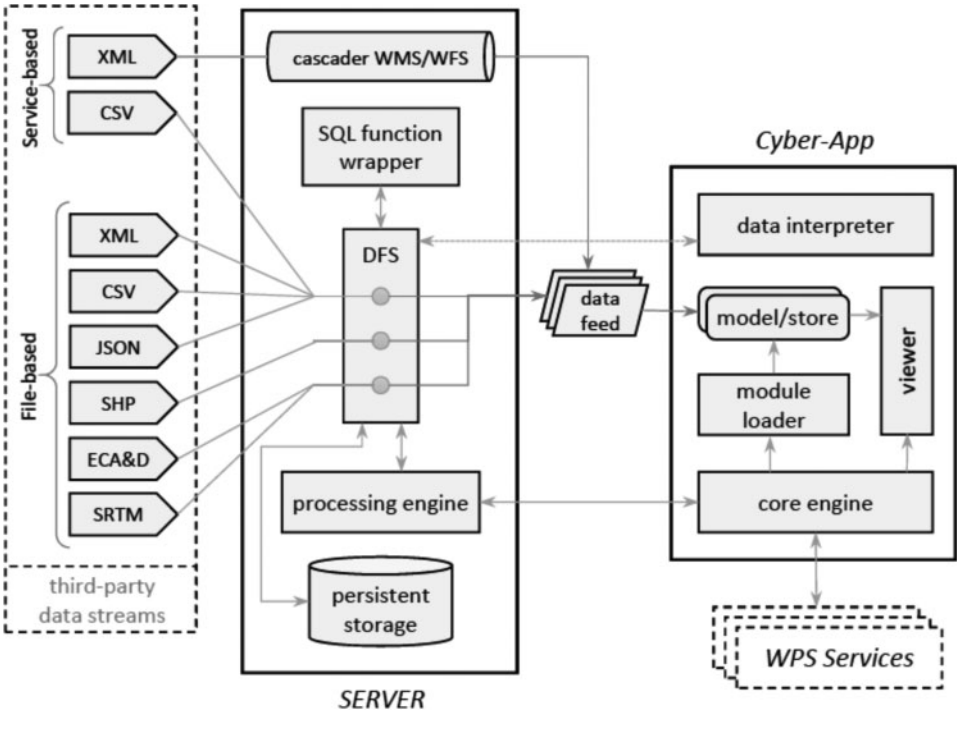

Figure 3. Cyber-app architecture.

data. This allows cyber-apps to manipulate third-party data and local data transparently using the same set of functions. The DFS also generates visual results in the form of WMS as response to requests from views.

The *data interpreter (DI)* appmod associates new data sources with the cyber-app. When the user of the cyber-app wants to include a new data-set in the cyber-app at run-time, the DI appmod obtains the metadata of this data-set to create a data-set schema, after which the corresponding mappings between this schema and the cyber-app's information schema are created. In some cases, multiple new data sources are combined to create a proper mapping onto the cyber-app information schema. The resulting mapping is then used by the DI appmod to create a volatile information model that wraps the new data source and makes it available to the cyber-app. This process takes place interactively and is initiated by the user. Once the new information model is created and loaded, the cyber-app can access this new data feed during its operation through the DFS appmod. Third-party data sources can be file- or service-based. For this process to work properly in real-time, the DFS is configured to work with the standard formats for distribution of open data. File-based formats are XML, CSV, JSON, Esri shapefile, HGT for SRTM data SRTM (2012) and compressed NetCDF for E-OBS meteorological data (NetCFD 2012). Service-based sources use a REST API or an OGC Web Service interface (OWS 2012). The usage of data sources in additional formats would require the addition of a corresponding interpreter appmod.

The *module loader (ML)* appmod is responsible for incorporating appmods into the application at run-time. A cyber-app starts with a minimum set of appmods, and includes new ones as the need arises during its execution. Newly added appmods can be permanent ones, which are part of the core engine, or volatile ones such as those created by the DI module. The ML incorporates not only the actual appmods, but also their metadata descriptions into the cyber-app.

The *processing engine (PE)* appmod takes care of the execution of functions involving complex and lengthy queries and operations, such as simulations or simultaneous read/write operations over multiple records.

The *core engine (CE)* consists of one or more appmods and their specific function is directly related to the purpose of the application. The core engine thus contains all the application logic. The CE controls the operations executed by the PE appmod and uses AJAX-based communication to request execution of tasks and retrieve processing results. The CE also coordinates the storage of user-generated content on the persistent storage component. Likewise, the CE is responsible for the interaction with third-party functions available over the CI as Web Processing Services.

5. Cyber-app example

In this section, we describe the cyber-app development process through a simple *app that provides flood information*, which is an application aiming at the provision of flood risk information for a variety of users within Europe. The flood information app allows users to determine whether an area of interest is at risk for flood. It uses data feeds from open government services. The application allows the use of different flood models to determine flood risk levels. The results are presented as levels of risk with respect to threat conditions. Users can also register a location to subscribe to

flood alerts. In the event of a flood, the user can also report conditions at his location in real-time. The flood information app can be accessed via a browser-based environment in a PC or smartphone.

The design process starts with the specification of the cyber-app's information schema. Since this is an illustrative example, we limit the scope to floods caused by rain in and around city areas. The quality of this schema is important, as the various appmods rely on it for the manipulation of data-sets, whether known or discovered during execution. The elements of the flood information schema are:

- *elevation*: heights of the area of interest are represented by a grid with a resolution of maximum 10 meters (typical width of European streets).
- *rivers*: geometries of the relevant watershed including widths and depths.
- *water bodies*: geometries and capacity of lakes, reservoirs, canals, etc.
- *water levels*: sensor observations over rivers and water bodies.
- *obstacles*: geometries of objects that have impact on the flow of water such as buildings, vegetation and dams. Each of these types of object also has a value associated that specifies its hydraulic roughness (the degree to which the obstacle displays resistance to water flow).
- *pathways*: roads and streets potentially form the main water flow path during floods. Roads can be used instead of buildings when determining the spread of a flood.
- *flood sources*: rain volume in millimetres over the area of interest.
- *area of interest*: the area to be analysed for flood risk.

Each of the elements on the information schema becomes an information model in the cyber-app. Listing 1 exemplifies the code for managing obstacle data. When this appmod is loaded, all controllers and views of the cyber-app can seamlessly access the data. The actual data that populates this information model could come from the cyber-app persistent storage or from data sources accessible via the DFS.

The second step in the design process is the specification of the core engine appmod. One must identify the correct application workflow and use it to create a partition to identify required appmods and distribute responsibilities over the components of the application.

For the flood information app, the main workflow consists of the following steps:

(1) *definition of the area of interest*; the user determines the particular zone for flood risk assessment, which is part of the flood plain used in the computation.
(2) *selection of data sources*; the cyber-app checks, given area of interest, the required data-sets and presents an overview to the user. The user may accept to compute with the predetermined data-sets, may change and provide own sources or might have to provide sources if these are not available. Based on the data sources that are available for the cyber-app to operate with, a list of computational models is presented to the user. Some of the flood models listed below may therefore be filtered out.
(3) *selection of the computational model*; the choice of computational model depends on the scale of the problem, the computational resources available and the needs of the user. The different types of flood model may require

slightly different inputs and have different performance characteristics. The set of possible computational models that are realistic for the flood information app include the following:

- `0D` Flood model, in which a predicted water level is estimated and then potential water levels are computed by subtracting the ground elevation form the water level plane. The execution time is in the order of magnitude of seconds.
- `1D` Flood model, in which the water flow is treated as linear features in the down-valley direction. This is suitable for narrow and confined flood plains. This flood model produces water depths and average water velocity, and the execution time is measured in minutes.
- `2D` Flood model, in which the water flow is treated as a plain with two axes. This flood model is suitable for a flood plain with complex topography. It produces flood extent, water levels, average water velocity and downstream outflow. The execution time for this case is measured in hours.

Presenting an exhaustive list of flood models and their variations falls outside the scope of this paper, but a detailed analysis of such flood models can be found in (Asselman et al. 2009).

(4) *Accessing real-time values*: before the actual computations take place, the cyber-app allows the inclusion of sensor data and user-generated content, such as elevation, water levels and potential path flows. The cyber-app uses the DI appmod to create the appropriate schema to be used. XML is then used as the carrier of the real-time content.
(5) *Flood computation*: given the choices and inputs specified by the user, the cyber-app makes use of the processing engine to realise the computation. For this, the cyber-app requests the processing engine to execute the required computation. The processing engine retrieves the required input data from its persistent storage or via the DFS appmod, and carries out the computation. On completion, the processing engine sends a notification to the cyber-app, which in turn uses its views to retrieve and display results.
(6) *Presentation*: with results available, the core engine hands over control to the V-type appmods. Multiple views are available so that the results can be presented in different ways. Presentation could take the form of an animation, a map, a graph or a table. Each of the views involved makes its data requests independently to the processing engine to present the results.

Figure 4 shows the distribution of responsibilities among appmods in the realisation of the *flood information app*. The process logic depicted as numbers in the figure corresponds to the enumerated workflow steps listed above.

One of the critical success factors of a cyber-app is the documentation associated with its various appmods. To make possible for an arbitrary user, either non-expert or expert, to choose one computational model over another, the cyber-app provides access to documentation associated with its various computational models. These descriptions present details on the purpose of the model and the conditions under which it can be used. The descriptions also detail the requirements in terms of data

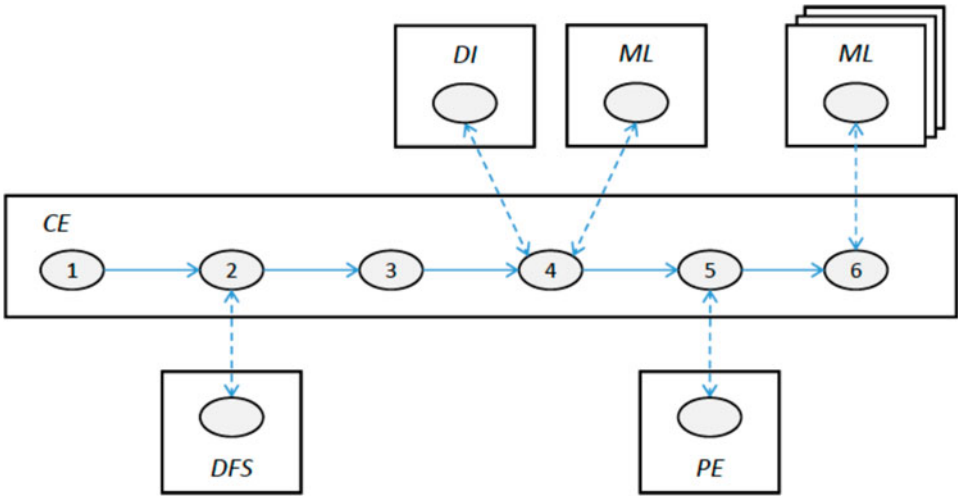

Figure 4. FloodApp realisation structure.

inputs to the flood model, and whether or not information models compatible with the flood model are known to the cyber-app.

6. Conclusions

This paper introduces the concept of cyber-app as the preferred mechanism to construct web systems to realise the vision of DE. Cyber-apps are device-independent software components that are based on browser technology, such as HTML, JavaScript, CSS and XML. Cyber-apps take advantage of today's technological possibilities to exploit the vast number of resources available online. To this end, a cyber-app is conceived as a container that utilises data and computational resources available in the underlying infrastructure, such that cyber-apps become an exploitation layer that adds application-based functionality to existing service repositories (geo-portals). The paper presents the appropriate architecture that accommodates the seamless exploitation of resources. To that end, the internal structure of a cyber-app follows a strict model-view-controller architecture, in which different components have discrete responsibilities in only one of the three areas. As a result, components are considered to be one of three types: M-, V- and C-Type, being information models, views and controllers, respectively. Cyber-app components are called appmods. The communication patterns between appmods follow an event-based design principle in which only the core component of the application, known as the core engine is process-aware. The participation of all other appmods is restricted to specific events that occur during execution. Cyber-apps are composed of permanent and volatile appmods, with permanent appmods being common to all cyber-apps, and volatile appmods specific to individual cyber-apps.

The paper also presents an extensive list of open data resources, which can be exploited via cyber-apps. Given these data resources, the paper presents a realistic but not yet implemented cyber-app example addressing the needs of end-users in the assessment

of flood risks. The flood information app is designed under the principles highlighted in the paper. This results in the identification of suitable appmods and proper distribution of execution responsibilities. Given the location of the user, the cyber-app identifies the set of required data resources, and presents a choice of potential computational models to the user. The example also illustrates how a cyber-app incorporates new data resources in real-time, such as sensor data and user-generated content.

Cyber-apps are the next step in the evolution of the web applications that allow to take advantage of the exceptional range of free and open data resources in combination with the ever-increasing capabilities of modern web browsers.

Acknowledgements

We are indebted to the anonymous reviewers, whose critical comments and constructive criticism helped us to improve an earlier version of the paper. Rolf de By's work on this publication was in part supported by the Dutch national program COMMIT (2012).

References

Asselman, N., P. Bates, S. Woodhead, T. Fewtrell, S. Soares-Frazão, Y. Zech, M. Velickovic, et al. 2009. *Flood Inundation Modelling: Model Choice and Proper Application*. Technical Report T08-09-03. FLOODsite: Integrated Flood Risk Analysis and Management Methodologies. http://www.floodsite.net/html/publications2.asp?by=documentTitle.

Atkins, D. E., K. K. Droegemeier, S. I. Feldman, H. Garcia-Molina, M. L. Klein, D. G. Messerschmitt, P. Messina, J. P. Ostriker, and M. H. Wright. 2003. *Revolutionizing Science and Engineering through Cyberinfrastructure*. Technical report. The National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure.

Baresi, L., S. Colazzo, L. Mainetti, and S. Morasca. 2006. "W2000: A Modelling Notation for Complex Web Applications." In *Web Engineering*, edited by Emilia Mendes and Nile Mosley, 335–364. Berlin: Springer.

Brodaric, B., P. Fox, and D. L. McGuinness. 2009. "Geoscience knowledge representation in cyberinfrastructure." *Computers & Geosciences* 35 (4): 697–699. doi:10.1016/j.cageo.2009.01.001.

Burbeck, S. 1992. *Applications Programming in Smalltalk-80(TM): How to Use Model-View-Controller (MVC)* [online]. Accessed February 27. http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.

Ceri, S., P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. 2003. *Designing Data-Intensive Web Applications*. The Morgan Kaufmann Series in Data Management Systems. San Francisco, CA: Morgan Kaufman Publishers.

CKAN. 2012. *The World's Leading Open-Source Data Portal Platform* [online]. Accessed February 27. http://ckan.org.

COMMIT. 2012. *A Public-Private Research Community* [online]. Accessed February 27. http://commit-nl.nl/about-commit.

Craglia, M., K. de Bie, D. Jackson, M. Pesaresi, G. Remetey-Fülöpp, C. Wang, A. Annoni, et al. 2012. "Digital Earth 2020: Towards the Vision for the Next Decade." *International Journal of Digital Earth* 5 (1): 4–21. doi:10.1080/17538947.2011.638500.

Cranshaw, J., R. Schwartz, J. I. Hong, and N. Sadeh. 2012. "The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City." In *Proceedings of the 6th International AAAI Conference on Weblogs and Social Media (ICWSM-12)*, June 4–7. Dublin: The AAAI Press.

Darema, F. 2010. "CyberInfrastructures of Cyber-Applications-Systems." *Procedia Computer Science* 1 (1): 1287–1296. doi:10.1016/j.procs.2010.04.143.

Data Overheid. 2012. *Open Data Portal of the Dutch Government* [online]. Accessed February 27. http://data.overheid.nl.

de By, R. A., R. Lemmens, and J. Morales. 2009. "A Skeleton Design Theory for Methodical Construction of SDI Nodes and SDI Networks." *Earth Science Informatics* 2 (4): 299–313. doi:10.1007/s12145-009-0034-7.

De Troyer, O. M. F., and C. J. Leune. 1998. "WSDM: A User Centered Design Method for Web Sites." *Computer Networks and ISDN systems* 30 (1–7): 85–94.

Díaz, P., and I. Aedo. 2007. "Towards Efficient Web Engineering Approaches through Flexible Process Models." *Journal of Systems and Software* 80 (8): 1375–1389. doi:10.1016/j.jss.2006.10.042.

ECA&D. 2012. *The European Climate Assessment & Dataset project* [online]. Accessed February 27. http://eca.knmi.nl.

Esclaona, M. J., and N. Koch. 2004. "Requirements Engineering for Web Applications – A Comparative Study." *Journal of Web Engineering* 2 (3): 193–212. http://www.pst.informatik.uni-muenchen.de/~kochn/KochEscalonaJWE-rev.pdf.

Fatolahi, A., S. S. Some, and T. C. Lethbridge. 2012. "A Meta-Model for Model-Driven Web Development." *International Journal Software Informatics* 6 (2): 125–162. http://www.ijsi.org/ch/reader/view_abstract.aspx?file_no=i117.

GeoNames. 2012. *The GeoNames Geographical Database* [online]. Accessed February 27. http://www.geonames.org.

GEOSS. 2012. *The Global Earth Observation System of Systems Project* [online]. Accessed February 27. http://www.geoportal.org/web/guest/geo_home.

Gitzel, R., A. Korthaus, and M. Schader. 2007. "Using Established Web Engineering Knowledge in Model-Driven Approaches." *Science of Computer Programming* 66 (2): 105–124. doi:10.1016/j.scico.2006.09.001.

GMES. 2007. *Global Monitoring for Environment and Security* [online]. Accessed February 27. http://www.esa.int/Our_Activities/Observing_the_Earth/GMES/Services_overview.

Goodchild, M. F. 2008. "The Use Cases of Digital Earth." *International Journal of Digital Earth* 1 (1): 31–42. doi:10.1080/17538940701782528.

Google. 2012. *Google's Public Data Directory* [online]. Accessed February 27. http://www.google.com/publicdata/directory.

Governor, J., D. Hinchcliffe, and D. Nickull. 2009. *Web 2.0 Architectures: What Entrepreneurs and Information Architects Need to know*. Sebastopol: O'Reilly Media.

Hasan, S. S., and R. K. Isaac. 2011. "An Integrated Approach of MAS-CommonKADS, Model-View-Controller and Web Application Optimization Strategies for Web-Based Expert System Development." *Expert Systems with Applications* 38 (1): 417–428. doi:10.1016/j.eswa.2010.06.080.

INSPIRE. 2007. *Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 Establishing an Infrastructure for Spatial Information in the European Community (INSPIRE)* [online]. Accessed February 27. http://inspire-geoportal.ec.europa.eu.

JSON. 2002. *The JavaScript Object Notation* [online]. Accessed February 27. http://www.json.org.

Kappel, G., B. Pröll, S. Reich, and W. Retschitzegger. 2006. *Web Engineering the Discipline of Systematic Development of Web Applications*. Heidelberg: John Wiley & Sons Ltd.

Koch, N., A. Knapp, G. Zhang, and H. Baumeister. 2008. "HumanComputer Interaction Series, UML-based Web Engineering: An Approach Based on Standards." In *Web Engineering: Modelling and Implementing Web Applications*, edited by Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina, 157–191. London: Springer.

Krasner, G. E., and S. T. Pope. 1988. "A Cookbook for Using the Model-View Controller User Interface Paradigm in Smalltalk-80." *Journal of Object-Oriented Programming* 1 (3): 26–49. http://dl.acm.org/citation.cfm?id=50759.

Leff, A., and J. T. Rayfield. 2001. "Web-Application Development Using the Model/View/Controller Design Pattern." In *Proceedings of the 5th IEEE International Enterprise Distributed Object Computing Conference, EDOC'01*, September 4–9, edited by F. M. Titsworth, 118–127. Seattle, Washington DC: IEEE Computer Society.

Livehoods. 2012. *Using Machine-Learning to Study Cities* [online]. Accessed February 27. http://livehoods.org.

Melton, J., J. E. Michels, V. Josifovski, K. Kulkarni, and P. Schwarz. 2002. "SQL/MED: A Status Report." *ACM SIGMOD Record* 31 (3): 81–89. doi:10.1145/601858.601877.

Mendes, E., N. Mosley, and S. Counsell. 2006. "The Need for Web Engineering: An Introduction." In *Web Engineering*, edited by Emilia Mendes and Nile Mosley, 1–27. Heidelberg: Springer.

Microsoft. 2012. *Web Application Development with WebForms* [online]. Accessed February 27. http://www.asp.net/web-forms.

Natural Earth. 2012. *Natural Earth Public Domain Dataset* [online]. Accessed February 27. http://www.naturalearthdata.com.

NetCFD. 2012. *Network Common Data Form* [online]. Accessed February 27. http://www.unidata.ucar.edu/software/netcdf/docs.

Nielsen, J. 2007. *Web 2.0 Can Be Dangerous* [online]. Accessed February 27. http://www.useit.com/alertbox/web-2.html.

Nielsen, J. 2009. "What Is Usability?" In *User Experience Re-Mastered*, edited by Chauncey Wilson, 3–22. Burlington, MA: Morgan Kaufmann Publishers.

OS OpenData. 2012. *Ordnance Survey OpenData Portal* [online]. Accessed February 27. http://www.ordnancesurvey.co.uk/oswebsite/products/os-opendata.html.

OSGeo. 2012. *OSGeo's Public Geospatial Data Project* [online]. Accessed February 27. http://wiki.osgeo.org/wiki/Public_Geospatial_Data_Project.

OSM. 2012. *The OpenStreetMap Project* [online]. Accessed February 27. http://www.openstreetmap.org.

OWS. 2012. *OGC Web Service Common Implementation Specification* [online]. Accessed February 27. http://www.opengeospatial.org/standards/common.

PDOK. 2012. *Dutch Public Services on the Map* [online]. Accessed February 27. http://www.nieuwsinkaart.nl/pdok.

Poore, B. S. 2011. "Users as Essential Contributors to Spatial Cyberinfrastructures." *Proceedings of the National Academy of Sciences of the United States of America PNAS* 108 (14): 5510–5515. doi:10.1073/pnas.0907677108.

Pulicdata EU. 2012. *The European Unions Public Data Portal* [online]. Accessed February 27. http://publicdata.eu.

Rod Blais, J. A., and H. Esche. 2008. "Geomatics and the New Cyberinfrastructure." *Geomatica* 62 (1): 431–443. http://dspace.ucalgary.ca/jspui/handle/1880/46638.

SRTM. 2012. *The Shuttle Radar Topography Mission (SRTM): The Mission to Map the World* [online]. Accessed February 27. http://www2.jpl.nasa.gov/srtm/index.html.

Trygve, M. H. 1979. *The World Bank's Open Data Catalog* [online]. Accessed February 27. http://heim.ifi.uio.no/ ~trygver/themes/mvc/mvc-index.html.

UKGDP. 1979. *UK Governments Official Open Data Portal* [online]. Accessed February 27. http://data.gov.uk.

World Bank. 2012. *The World Bank's Open Data Catalog* [online]. Accessed February 27. http://data.worldbank.org/data-catalog.

Wright, D. J., and S. Wang. 2011. "The Emergence of Spatial Cyberinfrastructure." *Proceedings of the National Academy of Sciences of the United States of America PNAS* 108 (14): 5488–5491. doi:10.1073/pnas.1103051108.

Yang, C., R. Raskin, M. Goodchild, and M. Gahegan. 2010. "Geospatial Cyberinfrastructure: Past, Present and Future." *Computers, Environment and Urban Systems* 34 (4): 264–277. doi:10.1016/j.compenvurbsys.2010.04.001.

Yang, C., H. Wu, Q. Huang, Z. Li, and J. Li. 2011. "Using Spatial Principles to Optimize Distributed Computing for Enabling the Physical Science Discoveries." *Proceedings of the National Academy of Sciences of the United States of America PNAS* 108 (14): 5498–5503. doi:10.1073/pnas.0909315108.