Enhancing Integrated Environmental Modelling by Designing Resource-Oriented Interfaces

Carlos Granell (European Commission, Joint Research Centre, Ispra, Italy)
Laura Díaz (Institute of New Imaging Technologies, Universitat Jaume I, Castellón, Spain)
Sven Schade (European Commission, Joint Research Centre, Ispra, Italy)
Nicole Ostländer (European Commission, Joint Research Centre, Ispra, taly)
Joaquín Huerta (Institute of New Imaging Technologies, Universitat Jaume I, Castellón, Spain)

Abstract

Integrated environmental modelling is gaining momentum for addressing grand scientific challenges such as monitoring the environment for change detection and forecasting environmental conditions along with the consequences for society. Such challenges can only be addressed by a multi-disciplinary approach, in which socio-economic, geospatial, and environmental information becomes interconnected. However, existing solutions cannot be seamlessly integrated and current interaction paradigms prevent mainstream usage of the existing technology. In particular, it is still difficult to access and join harmonized data and processing algorithms that are provided by different environmental information infrastructures. In this paper we take a novel approach for integrated environmental modelling based on the notion of inter-linked resources on the Web. We present design practices for creating resource-oriented interfaces, driven by an interaction protocol built on the combination of valid linkages to enhance resource integration, accompanied by associated recommendations for implementation. The suggested resource-oriented approach provides a solution to the problems identified above, but still requires intense prototyping and experimentation. We discuss the central open issues and present a roadmap for future research.

Keywords*:* Integrated environmental modelling (IEM); Resource-oriented architectures; Representational State Transfer (REST); RESTful interfaces; Environmental Information Infrastructures; Spatial Data Infrastructure (SDI)

This paper can be cited as:

C. Granell, L. Díaz, S. Schade, N. Ostländer, J. Huerta. **Enhancing Integrated Environmental Modelling by Designing Resource-Oriented Interfaces**. *Environmental Modelling & Software*, 39: 229-246, 2013, ISSN 1364-8152. http://dx.doi.org/10.1016/j.envsoft.2012.04.013

1. Introduction

In a 2009/2010 survey that involved over 1000 scientists from 85 countries, the International Council for Science (ICSU) identified five scientific priorities, or Grand Challenges, in global sustainability research. Among others, the Grand Challenges include: (i) the development of observation systems needed to manage global and regional environmental change; (ii) the improvement in the utility of forecasts for future environmental conditions and their consequences for humans; and (iii) the investigation of institutional, economic and behavioural responses that can enable effective steps toward global sustainability (ICSU, 2011). A next generation of web-based environmental information infrastructures and services has been proposed as the required tool, which provides more dynamic systems, distributed sources of information, and stronger capacities for integration (Craglia et al., 2008).

One key aspect to unlock the full potential of environmental information infrastructures is the development of scientific research and technology for *advanced environmental monitoring and integrated environmental modelling* (IEM[i]). In the early 1990s, Dolk (1993, p. 250) claimed the need to ‘go beyond strictly representational issues to consider how models may be linked, or integrated, with one another’. With the improvement of Information and Communication Technologies (ICT), Harris (2002) highlighted the shift in doing scientific work from smaller, independent research teams to massive multi-disciplinary research groups addressing global problems. This shift, according to Harris (2002, p. 5), posed several challenges such as ‘finding solutions to the problems of communication between a wide range of disciplines’ as well as ‘the task of building and maintaining models of various kinds’. Parker et al. (2002) agreed on common issues to be tackled in the future to enhance IEM. Among others, the authors suggested the importance of ‘open, honest and transparent modelling processes’ (p. 216) as well as the need of ‘new tools to achieve science and knowledge integration’ (p. 216). Rizzoli et al. (2008) reviewed the current status of IEM and encouraged ‘the development of open standards for the exchange and reuse of modelling knowledge, including datasets and models in order to facilitate improved communication among integrated modelling frameworks’ (p. 103). The authors envision that the future of IEM will be based on component-based solutions in combination with distributed computing technologies to enhance the sharing and reuse of environmental models. In such a setting, environmental information infrastructures may become key technological enablers for building IEM applications based on distributed, web-based technologies.

Environmental information infrastructures share capabilities and drawbacks with Spatial Data Infrastructures (SDIs), which are advocated as the primary means for geospatial data and services integration and sharing on the Web (Nebert, 2004; Masser, 2005). The concept of SDIs involves the data itself, integration technologies, policies, institutional arrangements, and people, in order to avoid geospatial data and services remaining hidden in silos (Masser, 2005). Good progress has been made in the development of agreements, policies, and open standards addressing the required data models, metadata, service interfaces in addition to sharing philosophies (Rajabifard et al., 2006; Goodchild et al., 2007). However, the SDI – and equally the environmental information infrastructures – ecosystem should be thought of as a network of inter-linked *infrastructure nodes* which have not completely fulfilled the integration requirements. Every single node comprises a set of geospatial data and services aligned with the SDI principles grouped commonly by geographic or thematic criteria. For example, many European countries deployed national SDI, whereas the Infrastructure for Spatial Information in Europe (INSPIRE) (European Parliament and Council, 2007) has been put into place for integration on the continental level. In such a context, individual infrastructure nodes have been incorporated into common solutions. In other words, small to medium scale integration has been achieved, but the SDI community does not (yet) address the large scale, including stakeholders such as environmental scientists, socio-economic analysts, policy makers and citizens.

Non-SDI experts cannot easily access and explore geospatial content of potentially high value over distributed nodes due to: (i) the inherent complexity of some geospatial data standards (Tamayo et al., 2012); (ii) the lack of support for proper connections and linkages between geospatial data and services; and (iii) the diversity of interaction paradigms. On one hand, the diversity of interaction paradigms and complexity of some data standards makes the development of geospatial applications a challenging task. On the other hand, current data and services seem to still be disconnected. For instance, it is difficult for a user to jump from a given service or dataset in one infrastructure node to alternative services deployed in other nodes. In practical terms this impedes the full potential of operational SDIs as a network of inter-linked nodes.

In order to avoid building similar silos again, we envision a solution in which geospatial and environmental resources can be accessed and combined in a straight forward manner using a uniform interaction paradigm, independent of the type of resources deployed in an infrastructure node. The proposed modelling approach abstracts from diverse information infrastructures for environmental resources and suggests a general level integration approach. It includes a new way of exploiting the access, reuse, and linkages between environmental resources. Particularly, we address the ICT perspective because we consider technology as a central driver for innovation and development. One of the future targets is a simpler interface with environmental modelling capacities in order to combine these features with socio-economic simulations.

In the following section we briefly introduce an integrated modelling use case to be used throughout this paper. Section 3 provides a larger perspective of EIM approaches related to our work. Section 4 motivates the use of the resource-oriented approach[ii] in environmental studies. We then discuss design practices for creating resource-oriented interfaces and application for actual EIM in Section 5. The applicability of the resource-oriented approach to the use case is described in Section 6 as a set of recommendations for its implementation. The two final sections summarize the key features of our approach, on-going work, and a possible roadmap for IEM on the Web.

2. Use case

In order to assess our approach in a more illustrative manner, in this section we introduce a use case based on a scenario proposed by McInerney et al. (2012) to monitor and assess the impact of forest fires in protected areas in Europe. The proposed use case enables us to:

- describe how the same use case can be modelled from different viewpoints, namely, those described in Sections 3 and 4; and
- demonstrate on a more practical basis the application of our resource-based approach for an integrated modelling use case (Sections 5 and 6).

Advance monitoring and assessment of changes, like those provoked by the impact of forest fires in protected areas, is necessary for effective decision-making (McInerney et al., 2012). The European Forest Fire Information System (EFFIS[iii]) provides users with data and tools to monitor forest fires in Europe. Among others, EFFIS provides models which reflect the spatial distribution of forest fires within protected areas. Such results may be used in other disciplines such as environmental health to assess, for instance, the forest fire's emissions impact on air quality and its influence on the health of humans settled in the surroundings.

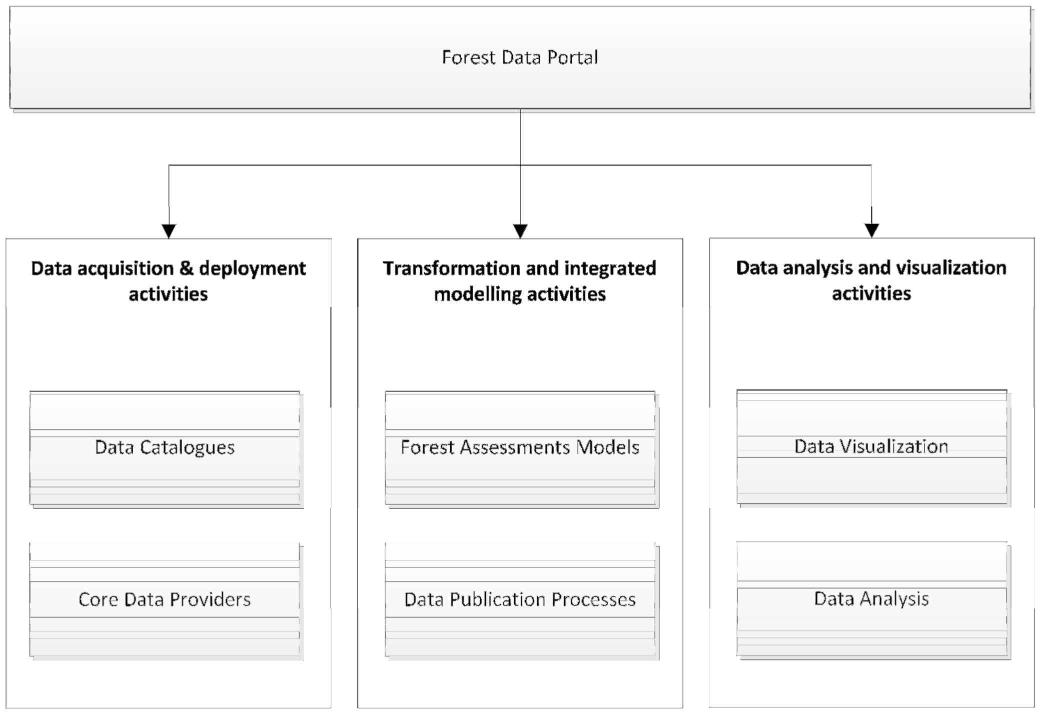

Forest Data Portal
Data acquisition & deployment activities
Transformation and integrated modelling activities
Data analysis and visualization activities
Data Catalogues
Forest Assessments Models
Data Visualization
Core Data Providers
Data Publication Processes
Data Analysis

Figure 1. Conceptual components to monitor the impact of forest fires in protected areas in Europe

Figure 1 illustrates a simplified architecture with the main components of the forest assessment use case. The first group of components addresses data collection and cataloguing. For example, the inputs data sets needed could be daily burnt areas coming from the European Forest Data Center (EFDAC[iv]), which provides up-to-date and reliable data on the state of European forest resources, and protected areas from biodiversity data repositories. The second group deals with integration, transformation and modelling activities. Components and models to assess the impact of forest fires in protected areas are required. Reichman et al (2011, p. 703) estimated that 'less than 1% of data results of modelling activities in ecological fields are accessible'. To this respect, McInerney et al. (2012) propose also an additional data publication step, where users are allowed to publish the (final and intermediate) results of the modelling process to the corresponding data catalogues and core data service components (left side of Figure 1) through the *Data Publication Process* (central part of in Figure 1). The third group provides users with components to visualize the spatial distribution and composition of forest fires for analysis and decision-making (right part of Figure 1).

The forest use case in Figure 1 is meant to introduce our resource-oriented approach in comparison with other modelling approaches described in Section 3. Figure 2, however, is further extended in Section 6 to describe how it can be modelled by means of resource-oriented interfaces. Figure 2 provides an overview of a workflow to monitor the impact of forest fires in protected areas. It is composed of two components from Figure 1. The first component, the *Impact Forest Fire* (IFF) model, takes several datasets as inputs such as the EFFIS burnt areas, protected areas, and the area of interest on which to compute the impact analysis. The result is an impact map in the chosen area accompanied with some statistics. The second component is the Data Publication Process (DPP) which permits publishing such an impact map in associated data services and data catalogues. This last step is meant to improve data sharing.

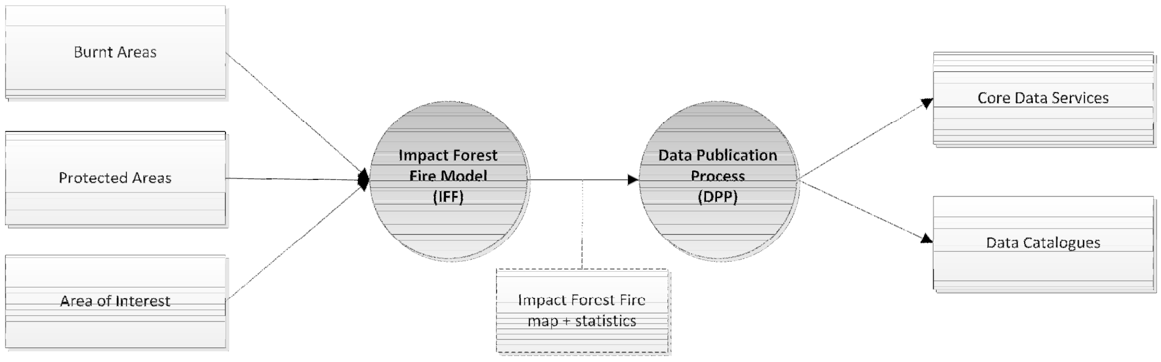

Burnt Areas
Protected Areas
Area of Interest
Impact Forest Fire Model (IFF)
Impact Forest Fire map + statistics
Data Publication Process (DPP)
Core Data Services
Data Catalogues

Figure 2. Example of integrated modelling to monitor the impact of forest fires in protected areas

3. Research Context

In a broader sense, IEM technologies encompass software engineering concepts and techniques to simplify the integration and programming efforts required by scientists and modellers to properly combine environmental data and models. From the ICT point of view, the adoption of software engineering techniques ensures that systems and applications are developed using a systematic approach (Verweij et al., 2010). Thus, environmental research communities have paid much attention to well-established software engineering solutions, such as *component-based* and recently to *service-based approaches*. Both put special emphasis on characteristics such as modularity and reusability to build integrated modelling tools and applications (Harris, 2002; Granell et al., 2010; Alexandrov et al., 2011). In this section we introduce component-based and service-based modelling approaches to build IEM solutions. We briefly describe how the previous use case could be modelled using each of these modelling strategies.

3.1 Component-Based Modelling

A component is an abstraction unit that allows modellers to share and encapsulate any given functionality and legacy code. Such components can be combined into integrated models or processes. Component-based modelling frameworks address the specific needs of integrated models, namely robust execution environments, full user control, and data-intensive processes. Component-based integrated models are then designed, deployed and executed locally by these rich frameworks. Argent (2004) provided an overview of component-based modelling approaches for integrated environmental modelling, and recently Jagers (2010) complemented it with an updated overview of some frameworks and systems such as Kepler (Ludäscher et al., 2006) and Taverna (Oinn et al., 2006). Although these frameworks share the same principles of modularity and decomposition, they also differ in their technological basis, which thereby impedes integrated modelling across different framework implementations (Rizzoli et al., 2008).

One of the most relevant initiatives for coupling components in IEM is the Open Modelling Interface (OpenMI) standard. OpenMI was designed to overcome interoperability issues among component-based platforms. It enables integrated modelling between third-party components and models that are implemented as local, OpenMI-compliant components (Gregersen et al., 2007). In short, OpenMI-based modelling consists of two phases. First, third-party models are converted into OpenMI-compliant components by inheriting from OpenMI base interfaces. Then, these OpenMI-compliant components are configured and combined to form integrated models by using the OpenMI Configurator Editor. This front-end application allows workflow modellers to construct IEM in a similar manner as Taverna and Kepler graphical tools do for scientific workflows. OpenMI has been successfully applied to various underlying models and component-based frameworks (Knapen et al., 2009). The authors report some examples of how components within ESMF and CCA frameworks have been coupled with OpenMI in hydrological applications. Castronova and Goodall (2010) proposed a semi-automated process for creating OpenMI-compliant components for modelling hydrologic processes. However, migrating local models and components to OpenMI interfaces remains a time consuming task because the technological skills and knowledge required is still a burden for many scientists.

Returning to our use case, from the component-based modelling perspective, OpenMI would be suitable to couple the components needed to assess the impact of forest fires in protected areas. Involved components and models could be transformed into an OpenMI-compliant component as commented earlier. By using the OpenMI Configurator Editor, scientists could link each input dataset with the corresponding components to assemble the IEM solution. Several strategies can be applied for coupling data sets to components (Castranova and Goodall, 2012). One such strategy would be to merge the required input datasets (e.g., burnt and protected areas) with the associated assessment components to create one integrated OpenMI-complaint model. The other strategy would be to exchange data among the OpenMI-compliant components at runtime. Regardless of the coupling strategy chosen, the set of data sets should be available locally for execution. Results of the model run, i.e., the spatial distribution of forest fires within protected areas could be locally stored for future runs.

The provision of a robust and complete control over the integrated model run is the advantage of adopting component-based modelling frameworks such as Kepler and the OpenMI Configurator Editor. It means users actively control each run of a model step, go back and forth within the workflow, and may manipulate state variables of an integrated model as required. Component-based frameworks gain robustness as integrated models are executed locally. Conversely, compared with service-oriented solutions (see below), component-based frameworks are less flexible in terms of integrating distributed components. For instance, third-party components should be first adapted to the corresponding interface of the target component-based framework. This extra transformation step may be difficult in some cases and limit the reuse of existing components by other component-based frameworks.

3.2 Service-Based Modelling

Service-based modelling implies that data and processing capabilities are exposed as network-accessible services via standard interfaces (Lee and Percivall, 2008). Service-oriented architectures (SOA) are used to develop collaborative, distributed web applications. Friis-Christiensen et al. (2009) define SOA as “open and interoperable environments based on reusability and standardized components and services”. Such services are then the building blocks for performing successful collaborative and multi-disciplinary research (Pearlman et al., 2011).

The previously mentioned SDIs follow the SOA paradigm and offer the possibility to access distributed, heterogeneous geospatial resources through a set of standards-based services that (in principle) allow one to connect a network of infrastructure nodes in an interoperable way. Most geospatial web services implement standard interfaces specified by the Open Geospatial Consortium (OGC) for serving, visualizing and processing data. An example is the OGC Web Processing Service (WPS) specification that has become the service interface of choice for exposing any geospatial processing capacity as a web service. Its *getCapabilities* and *describeProcess* methods offer service and process metadata, while the *execute* method triggers a concrete process. In this way, WPS-based services can be accessed remotely, reused and shared in different scenarios.

To meet complex user demands, SOA relies on best practices for service composition (Papazoglou and van den Heivel, 2007). Figure 3 depicts the main phases involved in service composition and workflows in SOA. Workflows are often created as a set of high-level tasks to achieve a certain goal. At this stage, workflow designers try to gather together all the pieces needed without entering into low-level implementation details. For instance the identification of relevant models and databases may be required to support certain socio-economic policies and constraints at this phase. This type of early modelling is called abstract workflow. Rather than being directly executable, abstract workflows capture information flow patterns, requirements, and constraints needed for conceptualizing and documenting workflows at a high level of abstraction.

Workflow instantiation means to transform abstract workflows into executable workflow descriptions (Figure 3). A key actor in describing executable workflows is the WS-BPEL (Jordan and Evdemon; 2007) specification that, under the OASIS[v] auspices, has turned into the de facto business process description language for web service workflows. WS-BPEL is supported by various workflow enactment engines that are either commercial like ActiveVOS[vi] or open source like Apache ODE[vii].

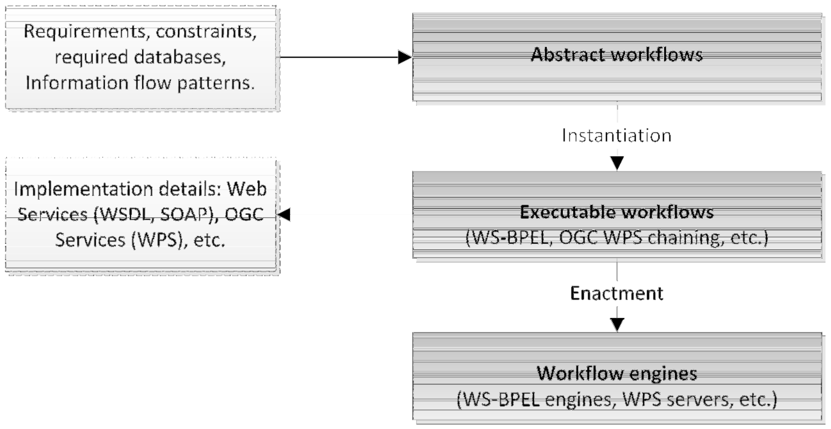

Requirements, constraints, required databases, Information flow patterns.
Abstract workflows
Instantiation
Implementation details: Web Services (WSDL, SOAP), OGC Services (WPS), etc.
Executable workflows
(WS-BPEL, OGC WPS chaining, etc.)
Enactment
Workflow engines
(WS-BPEL engines, WPS servers, etc.)

Figure 3. Phases involved in modelling geospatial workflows and processes in service-based modelling

Many attempts to integrate OGC services into service-based geospatial workflows have been proposed within the geospatial and environmental domains (Friis-Christiensen et al. (2009), Müller et al. (2010), Li et al. (2010), and Granell et al. (2010)). All these approaches aim at integrating and sharing geospatial content in terms of distributed web services. The use of OGC data and processing service interfaces demonstrates the flexibility and reliability of such services in operational geospatial workflows. This includes use cases from ecosystems and biodiversity modelling (Fook et al., 2009; Auer et al., 2011) to geospatial decision-making and map generalization (Brunner et al., 2009; Foerster et al., 2010).

McInerney et al. (2012) proposed a service-based modelling approach to build the forest use case. For instance, the required data sets such as EFFIS burnt areas and protected areas are available as features types from associated core data services (e.g., available data services deployed in EFDAC). Such features types are discoverable using catalogue services. The forest assessment models are wrapped as OGC WPS processes which can process assessments both at local and national scales. The results in this case may be automatically published in catalogue services and deployed in the associated core data services. Since the publication process is also wrapped as a WPS process (Díaz, L., Schade, 2011) it may be easily integrated into a geospatial workflow as mentioned earlier.

Service-based solutions, however, face some limitations within IEM: the lack of mechanisms to control interactively each step of a service composition run. Environmental modellers and scientists often re-run a given process once input variables and parameters have been adjusted or calibrated. As mentioned in Section 3.1, robust execution environments and interactive execution control are commonplace in component-based solutions.

3.3 Hybrid Approaches

Since reusability and scalability are desirable features in IEM (Section 1), some component-based modelling frameworks attempt to support remote web service access. The *Actor* concept of Kepler (Ludäscher et al, 2006) is a notable example. It provides mediation functionality to overcome interoperability issues such as different data encodings and structure (Wiederhold, 1992), and thereby enables the integration of external components and services into a common workflow. Users can select components via specialized *Actors* (e.g. *Web Service Actor*) and add them into Kepler-compliant workflows. Pratt et al. (2010) explore how Kepler-based workflows can be exposed as OGC WPS services and point to several limitations, which are caused by the disparate declaration of input and output parameters. Similarly, Barseghian et al. (2010) suggest the potential integration of OGC Sensor Web Enablement (SWE) within Kepler workflows, so that scientists can benefit from these standards and protocols to access to environmental observations and measurements in IEM solutions.

The latest release of OpenMI (version 2) also provides significant changes to simplify the underlying set of OpenMI base interfaces (Donchyts et al., 2010) and facilities the adoption of third-party components. Some authors already pointed out the need of exposing OpenMI-compliant components as web services in order to realize the full potential of service-based modelling. Goodall et al. (2011) obtained inspiration from the OGC WPS specification to design a web service interface built upon OpenMI. However, WPS compliance has been dropped. The authors were able to anticipate the recent agreement[viii] between OGC and OpenMI in promoting open geospatial standards related to IEM. This kind of collaborations that fosters interoperability between complementary standards will undoubtedly speed up the next generation of hybrid approaches. In essence, component-based frameworks are slowly moving towards a common, basic interface standard (e.g. OGC OpenMI 2.0) that would allow for generic single wrapper implementations optimized for the particular target environment in which the components are used.

4. A Resource-Oriented Approach

The analysis of the related work yields some interesting conclusions. Both component-based and service-based modelling techniques provide distinct advantages and limitations because both are optimized for specific application domains and scenarios. Service-based modelling techniques focus on reusing distributed services to create flexible business workflows, while component-based modelling frameworks offer robust mechanisms and environments for constructing interactive, data-intensive integrated models.

Both share the principle of communication interfaces. Standards-based interfaces have been successfully applied to combine a limited set of models and components in specific domains (Knapen et al., 2009). OpenMI provides the ability to enhance integrated modelling whenever backend components migrate to OpenMI-compliant interfaces. However, this kind of solution bears several limitations. Above all, components developed or adapted for a specific system are rarely compatible in practice with other frameworks (Rizzoli et al., 2008). Second, even for components deployed with the same framework, the task of continuously adapting components interfaces to new versions and emerging trends in communication protocols and technologies, limits considerably scalability and interoperability. For instance, new releases of the OpenMI interfaces or any other component interface will certainly require a tailoring of the contained components to updates in interface descriptions such as changes on attribute names or deprecated method names. Although OpenMI 2.0 interfaces, when implemented in a non-invasive manner, should not impact the underlying component, and tight-coupled interfaces in general may impede the re-use of services and components in IEM solutions.

In essence, the interface of components cannot evolve independent of the frameworks in which they are contained nor from client applications because such components, frameworks and client applications are tightly coupled to specific interfaces and communication protocols. Returning to our use case (Section 2), a migration from OpenMI 1.4 to version 2 would lead to changes in adapting the components for protected areas assessment. Similarly, from the service-based modelling perspective, the migration from the WPS 1.0 interface specification to the on-going WPS 2.0 would also require the modification of the interface descriptions for protected areas assessment services and client applications (e.g. Data Forest Portal). Furthermore, scalability and reuse get limited because of the need to continuously adapt the description of component interfaces to new changes.

The very principles of the Web provide an alternative to tightly-coupled interfaces. Since its inception, the Web has been continuously expanding to become the largest ever information infrastructure and integration platform. It is a repository made up of disparate resources such as documents and images, as well as services, processes, and models. Yet at the same time the Web provides easy mechanisms to enable users the creation, publication and connection of resources across different application domains. The content-centric aspect of the Web manifests itself in the growing popularity of Web 2.0 services and social networking services that enable citizen participation and user-content generation (Díaz et al., 2011).

Based on our past experiences in modelling geoprocessing service compositions for hydrological models (Granell et al, 2010), architectures for distributed geoprocessing services (Friis-Christiensen et al., 2009; Díaz et al., 2011), and the new requirements posed by sensor web in future environmental modelling (Havlik et al., 2011), we are now asking *whether the same architectural principles and mechanisms that shape the Web may serve to design uniform, resource-oriented interfaces to enhance integrated (environmental) modelling solutions*. The idea of exposing every piece of functionality (e.g., component, service) used in IEM solutions with a uniform interface is appealing enough that it is worth exploring.

Resource-oriented and standards-based communication interfaces share the essential idea but differ on the approach used. Both aim at easing the access and reuse of pieces of functionality in integrated models. Yet, in the latter case, every single framework (e.g., OpenMI, Kepler, etc.) provides a specific communication interface (also called Application Programming Interface, API) on top of a transport protocol (usually HTTP for remote access) to enable access, discovery, and execution of components. In the former case, resource-oriented modelling leverages HTTP itself as the application protocol, i.e., HTTP takes the role of an API, to access and manipulate any type of resource. As the HTTP protocol is omnipresent in distributed computing on the Web, it seems reasonable to use it directly rather than adding overlapping APIs on top of it. Our suggestion is to exploit the benefits of HTTP as a uniform communication interface for component-based and service-based modelling frameworks.

In the following section, we explore the design of such a resource-oriented solution in detail, suggest some implementation strategies, and highlight the importance of using uniform interfaces for enhancing integration of multi-disciplinary resources at any granularity level. We stick to a scenario where data providers, such as scientific institutions, expose their own environmental resources through an infrastructure node, following any of the technologies described previously. Such components and services may be used and accessed locally or remotely by third parties. Nevertheless, in such a scenario, building integrated models that involve components and services from different data providers remains a challenging task.

5. Design of RESTful Interfaces

After we discussed the motivations for the resource-oriented approach in the previous section, we now explain the way to develop a sustainable solution based on the Representational State Transfer (REST) architectural style (Fielding, 2000). REST imposes a set of constraints on the communication between clients and servers that guide software architects in designing concrete distributed systems and applications. In this section we look into some common aspects that must be considered for designing RESTful interfaces (Richardson and Ruby, 2007) and project those to enhance IEM. Where possible, each subsection follows a similar structure. A specific interface aspect from the REST viewpoint is introduced first, followed by a discussion on its application within the context of IEM. Recommendations on the implementation applied to our use case are presented in Section 6.

5.1 Specification of Resources

One of the first aspects in designing resource-oriented interfaces is to define the notion of a resource in respect to the application domain. Initially, any information that can be of interest is a resource. In other words, any informational entity may be regarded as a resource in the target RESTful application.

Design practices. In practice, any type of entity involved in IEM may be abstracted as a resource. From the engineering perspective, this is an important enhancement since the issue of dealing with different abstractions (component, service, process, models, etc.) adds an additional level of complexity to the design of IEM solutions. However, there are certain degrees of freedom in this statement, since the entities of interest in a specific domain may not directly match the set of exposed resources in the final application. Whereas components and services are regarded as coarse-grained entities, resources are usually considered fine-grained entities. Accordingly, a component or service in a given domain may get de-composed into a set of resources, depending on the particular application requirements. For instance, a model can be transformed into a set of resources (e.g., *process*, list of *inputs*, list of *outputs*, *state*, *synchronous execution*, *asynchronous execution*).

The previous example may give the initial impression that the number of involved resources tends to be much larger than the number of components and services. However, it is a common design practice to accommodate varied levels of granularity of resources by organizing resources into collections and composites (Allamaraju, 2010). A composite resource may thus provide a single, logical view of a set of interrelated resources, without hindering the independence of the contained single resources. Each resource may be used as an individual entity or as part of a composite resource. However, a collection groups resources of the same nature. Figure 4 illustrates the mapping of a single model into a set of resources. The *model* resource is a composite resource (in dark) that gives a proxy to inner resources such as *inputs*, *outputs* and *state*, which are not hidden by the *model* composite resource but may be also accessed individually. Similarly the *inputs* collection resource groups the list of *input* resources. In practice, finding the right granularity level of the target resources becomes a critical design decision since the remaining REST principles are all based on this concept. This reinforces the statement that collaboration among all parties, which are involved in designing resource-oriented interfaces for integrated modelling, is as critical as in the development of environmental models (Jakeman et al., 2006).

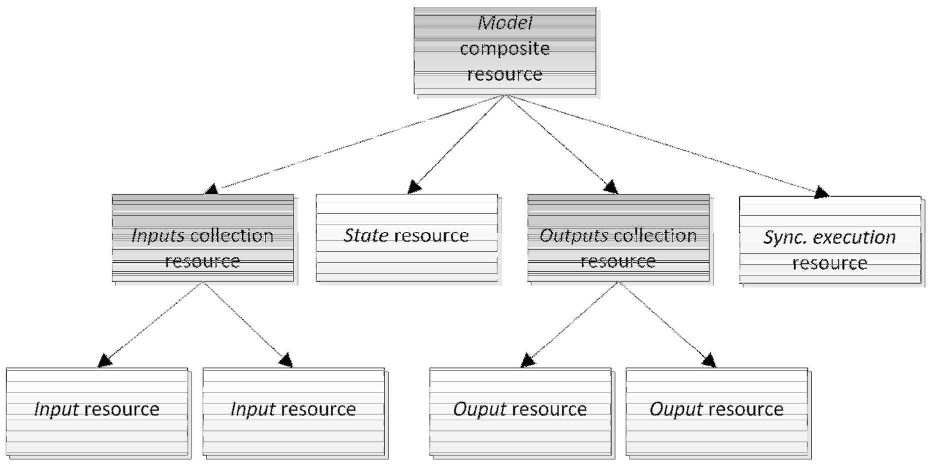


Figure 4. Use of composite and collection resources (in dark) to deal with multiple granularities

5.2 Resource Identifiers

A second aspect in REST is that every single resource must be uniquely identifiable. In general, a resource has to be correctly identified before it can be re-used. The Web uses the Uniform Resource Identifier (URI) (W3C/IETF, 2001) mechanism to name every resource. Apart from the purpose of identification, the use of URIs provides an extra benefit because resources are dereferenceable through their HTTP URI (Jacobs and Walsh, 2004). This means that client applications use URI identifiers for accessing the resource itself. In this way, the same mechanism is used for identification and physical access to the resource.

Design practices: Users and client applications do not require prior knowledge of the set of URIs for accessing specific resources such as *models*, *outputs* and *input* resources in Figure 4. In other words, each resource provider may choose different URI naming conventions without breaking with existing client applications. They just have to provide a global identifier for each resource, regardless of the concrete URI syntax used. For this reason, URI naming conventions are transparent to the HTTP protocol provided that such a URI is well-formed. The World Wide Web operates in the same manner. Users follow links within a given web page to move ahead rather than memorizing or constructing every URI by themselves.

Furthermore, attempts of standardizing the syntax of URIs just provoke the opposed effect, because a resource and its URI identifier become tightly coupled. When the URI naming conventions are uniquely imposed by the provider and hidden to client applications, then those become more independent and robust to broken links. Changes to URI naming policies can be easily implemented. For example, Web users suffer a recurrent case when they bookmarked a given HTTP URI and later fail to reach the concrete page due to slight changes in the URI naming conventions. Otherwise, letting client applications build a URI to access resources is an error-prone practice. Whenever possible, client applications should be given with the exact URI to access such resources.

The recommended way to access the full range of resources in a given server is via the canonical or root URI (Richardson and Ruby, 2007; Allamaraju, 2010). This is the unique URI that should be made available in public catalogues and registries. The role of canonical URIs is similar to one-stop geo-portals (Bernard et al., 2005; Goodchild et al., 2007), which allow efficient access to spatially distributed geospatial data and services offered by an infrastructure node. Client applications are then encouraged to bookmark the canonical URI instead of each individual resource's that might change over time. Changes may be not only due to URI syntax but also to the resource organization (e.g., a given model is provided with a new input resource). Accessing through the canonical resource ensures that client applications always get an up-to-date view of the plethora of resources behind the scene.

5.3 Uniform Interface to Manipulate Resources

Complementing the resource identification mechanism of HTTP URIs, the uniform interface constraint in REST establishes a standard way to access and manipulate resources regardless of their application domain. The uniform interface is driven by the principle of generality. Each resource has the same interface, which is derived from the standardised HTTP methods (Fielding et al., 1999). Most prominently, the GET method is for retrieving resource representations, POST method for creating new resources, PUT for updating resources, and finally, DELETE is meant to eliminate a given resource. In doing so, REST raises HTTP to the level of an application protocol.

Design practices: The use of HTTP as the unique application-level protocol means that HTTP methods become a kind of universal access API for any resource-oriented application. This fact has some important consequences. First, the semantics of the access interface is made explicit, due to the wide use and standardization of the HTTP protocol. Following any of the access interfaces described in Section 3, a client needs to understand every single access interface. If REST is used instead, regardless of the nature of resources in the server side, any client knows how to access them based on the semantics of the HTTP methods. As the interface remains invariable, underlying resources no longer have to be updated to new specific interface versions. Each resource evolves internally while it publicly exposes a uniform interface.

The second consequence of using HTTP methods is that the access interface (HTTP uniform interface) becomes decoupled from the resource representation (See section 5.4). The representation of a given resource is irrelevant for the action of issuing a HTTP GET request against that resource. Conversely, this does not hold for most component-based and service-based modelling solutions. Thinking of a WPS-based service: the response of a *getCapabilites* request, which contains the list of processes and supported operations, is described using the WPS schemas. Such a response is used by client applications to interact with a particular process (e.g., *describeProcess* request). In other words, the interaction protocol is not made explicit and is embedded to the description format used. A client thus discovers the details of the interaction protocol as it processes response descriptions. In REST, the access interface and the resource representation are independent from each other. Every single resource in Figure 5 (e.g. *mode*l, *inputs*) exposes the same uniform interface (set of HTTP methods). Most importantly, that access interface does not depend on the representation format of the resource. Examples of the use of the uniform interface are provided in Section 6.3.

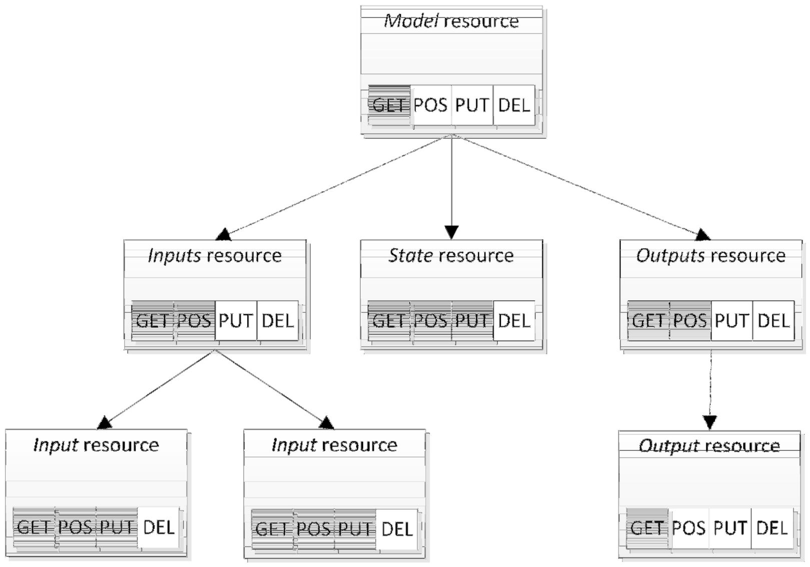


Figure 5. Inter-linked resources modelled as collections and members with uniform interface. For the HTTP methods, grey boxes denote allowed actions and white boxes unsupported actions (an extension of Figure 4)

5.4 Resource Representations

Resources are entities that may be regarded as being a set of attributes and properties which are accessible and editable. For instance, a model resource in Figure 5 is on one hand a piece of software functionality and, on the other hand, a list of input and output parameters, keywords, and textual descriptions. The resource representation is actually the unique part being manipulated through the HTTP-based uniform interface by client applications. That is, issuing a GET request against the *model* resource does not download the computational model itself but a representation of that *model* resource. A resource representation is then an informational view of a resource at a given time. Considering a resource that provides real-time sensor data as an example, issuing various HTTP GET requests against that resource at different time periods would most likely lead to different measurement values. In this way, REST disjoins abstract resources (e.g. a sensor, an algorithm, model) and concrete and manageable representations through the fixed set of HTTP methods. Resource representations are then manageable documents serialized according to specific media types (also called MIME types), such as HTML, XML and JPEG.

Design practices: The separation between *resources* and *resource representations* is not a new but a recurrent design principle, which is also applied to Web services and service-based solutions (Alonso et al., 2004; Papazoglou and van den Heuvel, 2007). A software component might provide a public OpenMI interface and a service a WPS-based interface description. Nevertheless, REST provides an additional benefit in that it enables a one-to-many relationship between a resource and their representations because the access interface (HTTP methods) and the resource representation (description) are separated from each other. A client application may request the inputs resource (Figure 6) and it may have different needs in terms of representation formats. In one case, it may expect results in HTML format for display in a web browser (e.g., documentation on evaluation and use of the collection of model inputs). In another case, the client may expect the *inputs* resource representation in JSON for automatic processing (e.g., building a data collection web form).

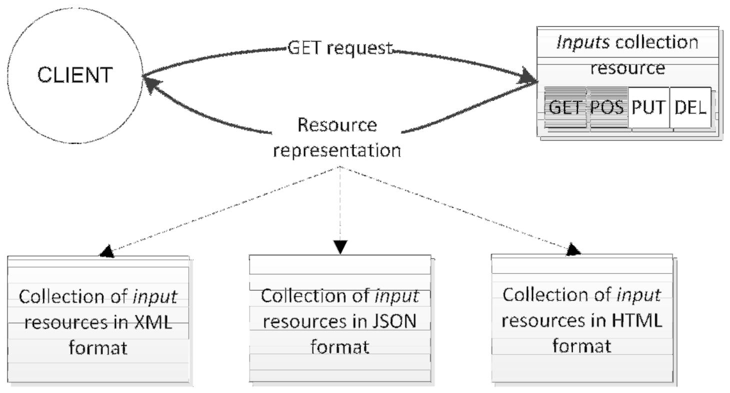

CLIENT
GET request
Inputs collection resource
GET
POS
PUT
DEL
Resource representation
Collection of input resources in XML format
Collection of input resources in JSON format
Collection of input resources in HTML format

Figure 6. Three different resource representations (XML, JSON, HTML) for the same *Inputs* resource

The ability to provide distinct media types (formats) for resource representations is an important design aspect when defining RESTful interfaces. The use of widely-used media types (e.g. HTML, XML and JSON) is recommended because they are universally understood by any client (Webber et al., 2010). MIME types may be used by clients to anticipate the expected format of a resource without the need to access the resource representation. Selecting one MIME type from various alternatives is then a desired capability since clients may not support every MIME type or they are just interested in one of them. Examples on the use of MIME types are provided in Section 6.4.

5.5 Resource State

REST relies on stateless communications between clients and services. No shared session is maintained or stored elsewhere (Fielding, 2000). This means that each request to the server must contain all of the information necessary so that a server fully understands the meaning of the request, without referring to any shared context. The same occurs for server responses to clients. Stateless communication means that every single request is independent from its predecessor and successor; that is, every request is self-containing and autonomous.

However, stateless communication does not imply stateless applications. The notion of state in RESTful applications might cause confusion. A few observations between application state, resource state, and internal state may provide clarification. Any resource encapsulates an *internal state* as is done with components and services. This internal state is maintained by the resource provider (right part of Figure 7) that decides which part of the resource's internal state is made public to clients. This public view of the resource's internal state is the *resource state* (central part of Figure 7). Resource representations play an important role to embody resource state because clients and servers communicate with each other by exchanging specific representations. For instance clients access the current state of a resource by retrieving its representation as response to a GET request.

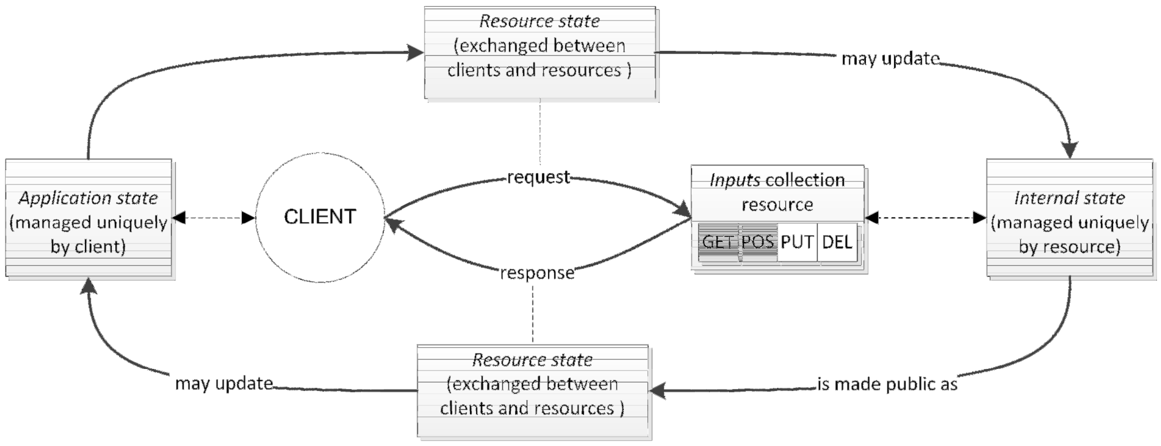

Resource state
(exchanged between clients and resources)
may update
Application state
(managed uniquely by client)
CLIENT
request
response
Inputs collection resource
GET
POS
PUT
DEL
Internal state
(managed uniquely by resource)
Resource state
(exchanged between clients and resources)
may update
is made public as

Figure 7. Relationships among internal, resource and application states in RESTful interactions

Also a software component can expose specific methods to retrieve and update the values of state variables (e.g., Donatelli et al. (2008) and Janssen et al. (2009)). The underlying concept is essentially the same: components (or services) and resources are stateless because both hide their internal state, and minimize dependencies with other components and deployment environments. The key difference here is that the access interface to the resource state is uniform in REST, while each component- and service-based framework implements a different method, i.e., it offers an uneven interface.

Moreover, due to stateless communication, RESTful client applications need a way to persist state (e.g., previous results) across several interactions with resources since the next interaction may depend on previous ones. This persistent state is called *application state*. In service-based approaches, statelessness is addressed by using dedicate middleware services that store and manage the application state shared by involved services (e.g. Kassahunm et al., 2010). Again, the key difference is not the role of the application state but who is in charge of managing and storing it. In REST, instead of using middleware, application state should be maintained exclusively by client applications (left part of Figure 7). This is required because the application state changes as a client interacts with resources in the server side. The emphasis here is that the management of application states in a RESTful manner is not a barrier but an opportunity to build decoupled resources and clients.

It is worth noting that RESTful clients are more similar to desktop clients than current clients on the Web (e.g. web mapping and mash-up clients) in terms of the management of application state. This means that RESTful client applications should be intelligent enough to perform certain actions and decide or suggest the next step to follow depending on the current application state. In the following we focus uniquely on the design aspects of the resource state since it is directly related with other design aspects such as resource representations (Section 5.4) and hypermedia (Section 5.6).

Design practices: A RESTful interface designer must respond to a relevant design issue: what part of the internal state should be projected in the resource representation as resource state? The answer depends on the specific resource to design and the application needs as a whole. Despite this, some best practices are available in mainstream ICT and therefore may be applicable to IEM too.

First of all, informative data such as attributes, parameter values, and textual descriptions are the first candidates to be part of the resource state because clients are normally interested in such kind of data. Nevertheless, the informational view of a resource is only one aspect of its internal state. Linkages to related resources are another aspect. They may have varied meanings or intentions. As commented on Figure 6, the representation of the *Inputs* collection resource contains all URI identifiers of the *input* resource. Such linkages, which are meant for structural purpose, may be part of the resource state. Other linkages represent transitions, i.e., URIs to related resources from the current resource in terms of the “possible” next steps in the application-specific information flow. (The notion of linkage and hypermedia will be examined in the next section.)

Additionally, the HTTP protocol itself already provides useful information about the resource state, which does not have to be embedded in resource representations. HTTP headers, which are used in the request-response communication, carry information to better understand the exchanged resource representation. These pieces of metadata may be considered part of the resource state since they help client applications to manage the resource representation. Delimiting the boundaries of the resource state is a vague task to a certain extent. As opposed to a centralized, shared session space (session state), which is common in service- and component-based solutions, RESTful applications rely merely on the elements involved in the request (e.g. HTTP headers, representations, hypermedia) to understand the state of a given resource. This leads to loosely-coupled, scalable systems since clients are not tied to specific implementations to model state (Foster et al., 2008).

Aside from previous practices, the resource state design in IEM may face specific requirements, such as time based execution of models and the ability to step back to a previous state of a given resource. For example OpenMI supports time-dependent models, which allows scientists to return to a previous time-step. One may believe that such specific requirements may have some implications on how state is maintained and managed in resource-oriented applications. In Section 6.5 we give some pointers to prototype applications that support the management of resource states by exclusively using HTTP capabilities. Innovative ways of exploiting existing mechanisms can deal with specific requirements which seemed to be a priori unaffordable.

5.6 Hypermedia Protocol

Just like uniform interface and identification of resources, hypermedia-driven application protocol (aka hypermedia) is yet another REST principle (Fielding, 2000). The term hypermedia refers to the interaction approach between clients and resources that is built upon the combination of valid state transitions. A state transition indicates a valid path between two resources. Returning to our use case (Figure 2), the data publication process (B) should be accessed only after getting the results from the impact forest fires model (A). This means that there is a valid transition from A to B, but not from B to A. Furthermore state transitions define dependences between resources that let client applications interact and navigate across resources keeping the application state consistent.

Design practices: The role of hypermedia is to certain extent similar to the use of workflow patterns to capture control-flow dependences between distinct tasks (Russell et al., 2006). These patterns yield the theoretical background in most workflow engines and component-based frameworks. While workflow patterns (sequences, conditionals, etc.) are the glue between components (or services), hypermedia manages the application flow between resources. The former is made explicit at design time and often does not require human intervention at run time. The latter is discovered at run time and is more flexible to changes, but may depend on human intervention. 'Intelligent' clients (see also Section 5.5) should be able to recommend users the next link (state) to follow according to user preferences and the application state.

Apart from the differences in the control-flow the main difference is the specification of control-flow primitives. As mentioned earlier (in Section 5.5), resource representations contain data (e.g. attributes values) and pointers/links to related resources. Whereas control flow primitives are independent in component- and service-based solutions because they are often defined in separated description languages, hypermedia (linkages) is an integral part of resource descriptions. Each single resource defines its network of related resources, such that there is no need for a third-party language because each resource indicates the set of potential state transitions. Coming back to our example (see also Figure 6), the representation of the *inputs* collection resource advertises a list of next transitions in terms of related resources (e.g. *input* resources). The linkages that are embedded in the representation of the *inputs* resource describe the *inputs'* valid state transitions. Furthermore, client applications interact with the deployed resources by using a hypermedia protocol as follows. As the set of state transitions in a given resource may vary over time, clients first retrieve the resource representation so as to get an up-to-date list of available state transitions. Next, clients make decisions on the most suitable target transitions depending on the meaning of these transitions and the current application state. Finally, the clients proceed to the selected state transition by issuing HTTP requests to the corresponding resource (see Section 5.3).

Nevertheless, RESTful client applications must be able to understand and correctly interpret such state transitions, as the workflow engines do with executable description languages (see

Figure 3). This implies each state transition must have a clear purpose and meaning, which is known to all client applications. A key point in designing resource-oriented interfaces is the identification and description of the state transitions to support linkages between resources. Clients need accurate information about the possible state transitions to ease integrated modelling. One simple mechanism to support proper state transitions is the use of typed links, which explicitly describe the meaning of each available transition. Typed links can be semantically annotated in order to provide client applications with the intended interpretation of links. We propose some examples of typed links for IEM in Section 6.6.

6. Implementation Recommendations

This section demonstrates the applicability of the design practices described in Section 5 for an integrated modelling use case. Practical examples throughout this section are based on part of the modelling use case introduced in Section 2. In particular, we stick our discussion to the example presented in Figure 2, which integrates an Impact Forest Fire model and a Data Publication Process to monitor the impact of forest fires in protected areas of interest.

6.1 Specification of Resources

Rather than specific technologies, the identification of the involved resources and their possible partonomic relations requires a clear understanding of the study problem. Resource modellers should rely on best practices and successful use cases when identifying the set of resources for particular applications (Allamaraju, 2010). Figure 8 represents the mapping of the Impact Forest Fire model in Figure 2 into a set of resources (Section 5.1).

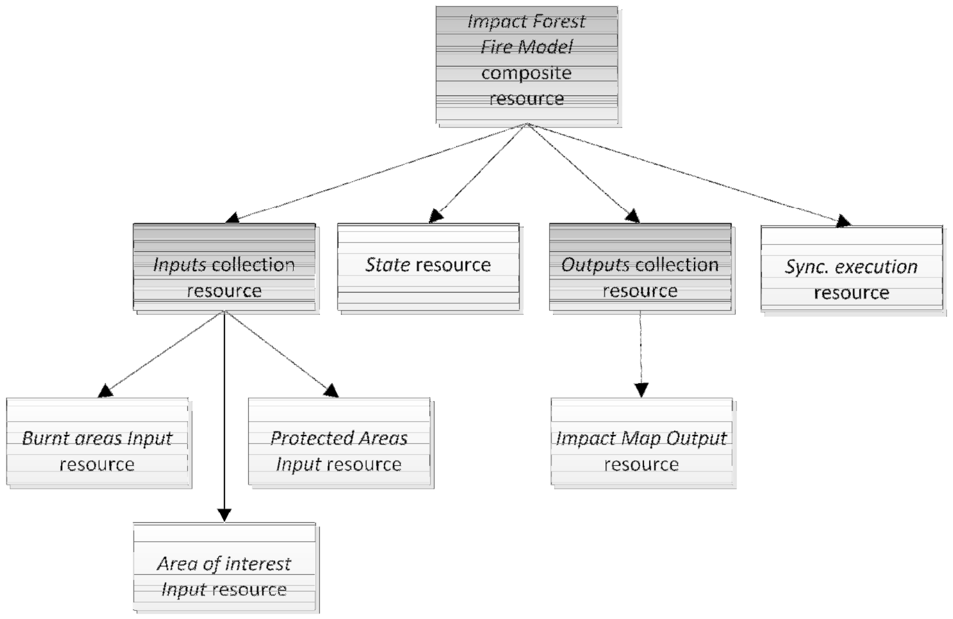

Impact Forest Fire Model composite resource
Inputs collection resource
State resource
Outputs collection resource
Sync. execution resource
Burnt areas Input resource
Protected Areas Input resource
Impact Map Output resource
Area of interest Input resource

Figure 8. Use of composite and collection resources (in dark) in the case of the Impact of Forest Fire (IFF) model (an instance of Figure 4)

To the best of our knowledge, resource-oriented approaches in IEM are still rare. However, some initial examples are emerging from some service-based modelling approaches, which attempt to specify a set of resources for a variety of OGC geospatial services with well-known, standardized data models (Mazzetti et al., 2009; Gao et al., 2010; Foerster et al., 2011; Finney and Watts, 2011; Janowicz et al., 2012).

6.2 Resource Identifiers

HTTP URIs, i.e., URI mechanism using HTTP schema, are needed to identify properly each resource on the Web. The same technology may also be utilized for resource identification in IEM. Immediate benefits from being aligned with the architecture of the Web (Jacobs and Walsh, 2004) is that all the features already provided by the Web such as URI bookmarking, syndicating, browsing, and discovery through general search engines are directly available when building resource-oriented applications.

Figure 9 shows a simple example of URI design of the space of public resources in the context of our use case[ix]. For example, *http://mynode.org/resources* represents the root or canonical URI (Section 5.2), i.e., the entry point to the public resources in an infrastructure node at *http://mynode.org/*. Client applications would discover contained resources in such a node by querying this public URI. The response to such a query may provide the collections of deployed resources grouped following a domain specific categorization, such as the spatial data services (SDS) categories defined by INSPIRE (European Parliament and Council, 2007). As each collection is actually a resource (Section 5.1), each one has a unique HTTP URI. In INSPIRE terms, the resource collection identified by the URI *http://mynode.org/resources/sds/invoke* would point to resources of type *Invoke SDS Services*, while the one pointed to by the *http://mynode.org/resources/sds/view* fragment would refer to resources of type *View Services*.

We can now make a case for membership relationships between resource collections. By following the URI *http://mynode.org/resources/sds/invoke,* the *invoke*[x] collection resource would contain the list of forest assessment models and processes. The use of a collection resource makes sense to group similar models and process in the forest assessment use case. Each individual member of the *invoke* collection, called member resource, is a distinct resource, i.e., a slope-based algorithm. The URI *http://my.sdinode.org/resources/sds/invoke/iff* refers for instance to the Impact Forest Fire model composite resource depicted in Figure 9. As commented in Section 5, each resource allows introspection by selecting certain outgoing links (Section 5.6) from its representation (Section 5.4) so as to inspect contained resources (e.g., *inputs*, *outputs*, and *state*).

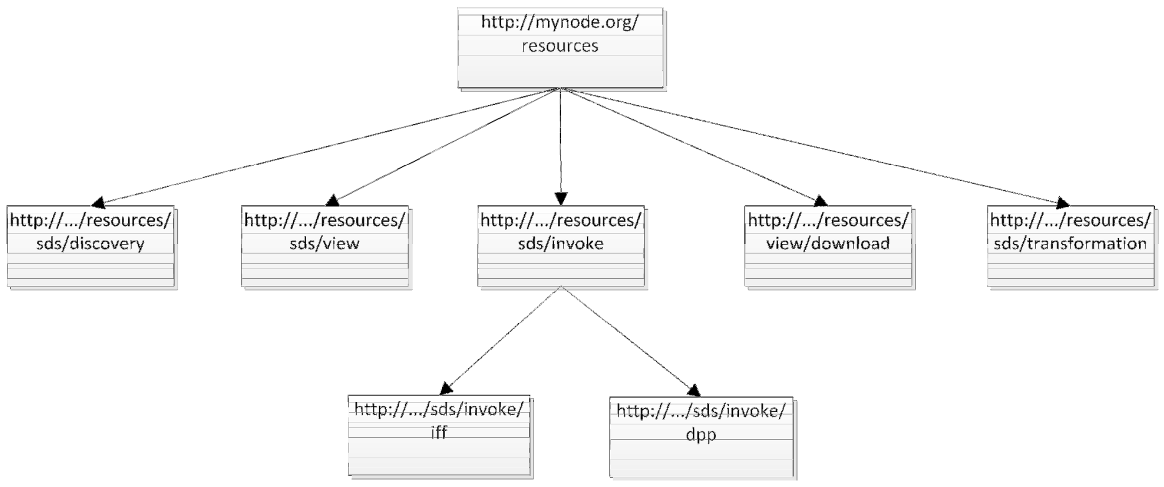

http://mynode.org/ resources
http://.../resources/ sds/discovery
http://.../resources/ sds/view
http://.../resources/ sds/invoke
http://.../resources/ view/download
http://.../resources/ sds/transformation
http://.../sds/invoke/ iff
http://.../sds/invoke/ dpp

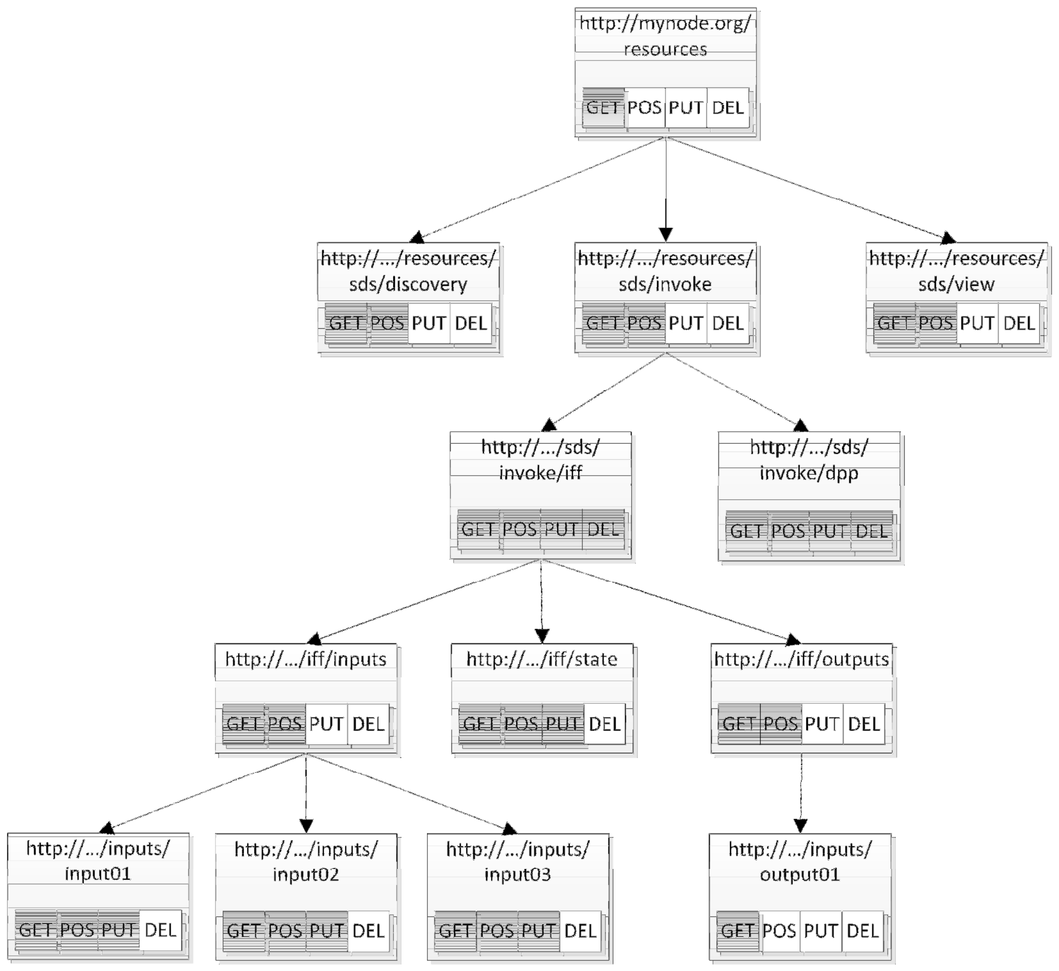
Figure 9. Specification of resources and their URI identifiers for the use case. The resource *iff* represents the *Impact Forest Fire Model* composite resource in Figure 8

In general, it is desirable to organize resources according to well-known categorizations (e.g. INSPIRE service types) or application-specific vocabularies. In this particular example, ‘invoke’ and ‘view’ are terms adhered to actual classification of SDS in INSPIRE, while ’iff’ and ’dpp’ should be defined in a domain-dependent vocabulary. Such terms can be used as keywords for browsing and searching resources. The promotion of standardized lists and shared classification schemes is encouraged to ease the specification of typed links (Section 6.6), as opposed to the standardization of URIs syntax.

6.3 Uniform Interface to Manipulate Resources

From the IEM’s point of view, the uniform interface provides an elegant and simple solution to access and manipulate any resource regardless of its nature and application domain. The HTTP protocol is the only technology required and, most importantly, many tools and libraries that abstract from the syntactical details of HTTP already exist for desktop, web and mobile environments.

Figure 10 depicts the role of the HTTP methods as the unique application-level protocol. The degree of freedom in the definition of method names in service- and component-based approaches is restricted here to a fixed set of meaningful methods. This eliminates a common source of errors, above all in client developments, which in practice often leads to a lack of interoperability. Notably, a given resource does not have to support all of the HTTP methods. As illustrated in Figure 10 the collection resource *invoke* supports GET and POST. This means that client applications can obtain the list of member resources (*iff* and *dpp*) via GET and create new member resources for this collection via POST. In contrast, a member resource, for example *iff*, supports all HTTP methods. A client application can thus perform any action against this resource. By selecting the appropriate HTTP methods, a resource designer can simulate any behaviour over individual resources.


http://mynode.org/resources
GET POS PUT DEL
http://.../resources/sds/discovery
http://.../resources/sds/invoke
http://.../resources/sds/view
http://.../sds/invoke/iff
http://.../sds/invoke/dpp
http://.../iff/inputs
http://.../iff/state
http://.../iff/outputs
http://.../inputs/input01
http://.../inputs/input02
http://.../inputs/input03
http://.../inputs/output01
Figure 10. Inter-linked resources modelled as collections and members with uniform interface for the use case. For the HTTP methods, grey boxes denote allowed actions, white boxes unsupported actions (an extension of Figures 8 and 9)

Finally, it is simple to add a new resource (input data item, process, model, etc.) into a resource-oriented application. It is just a matter of issuing a POST or PUT request to the corresponding collection resource that acts as a factory (i.e., a well-known creational design pattern (Gamma et al., 1995)) to create new member resources. In our use case, only POST requests are allowed against the *inputs* collection resource to create a new *input* member resource (Figure 10). Yet, some additional aspects must be taken into account when adding new resources to a given collection. Questions such as 'What attributes/properties should be part of the representation of the resource?', 'What representation formats to use?', and 'Which are the related resources to the current resource?' have to be addressed (Section 5.3).

6.4 Resource Representations

Media type selection is supported through the HTTP content negotiation mechanism (Fielding et al., 1999). In short, it lets clients set preferences (format, language, etc.) of the requested resource representations. For example, by using the *Accept* header as 'text/html, application/pdf', the client indicates that HTML is the preferred return format, although pdf is also acceptable. Although this mechanism is as old as the Web itself, it provides the basic protocol to select and receive a concrete representation format when a given single resource has multiple representations in different data formats (Richardson and Ruby, 2007). We revisit the capabilities of content negotiation in the next section.

Again, the HTTP protocol and the use of MIME types are the required technologies to support multiple resource representations. Most MIME types are registered under IANA (Internet Assigned Numbers Authority[xi]), so this is the first place to seek for candidate MIME types in defining RESTFul interfaces. However, in particular applications like environmental modelling, other media types may be required because they are either imposed in the particular application or simply generic media types that are not able to specify all the semantics of a given resource. In such cases the use of specialized media types may solve the problem, while these media types are made visible and shared between clients and resource providers (for instance being registered under IANA). An example of specialised MIME type is agroXML[xii], an XML-based language for soil data exchange in agriculture (Martini et al., 2009). A resource may advertise to clients the MIME type *text/xml; subtype+agroxml* if it comes encoded in the agroXML format.

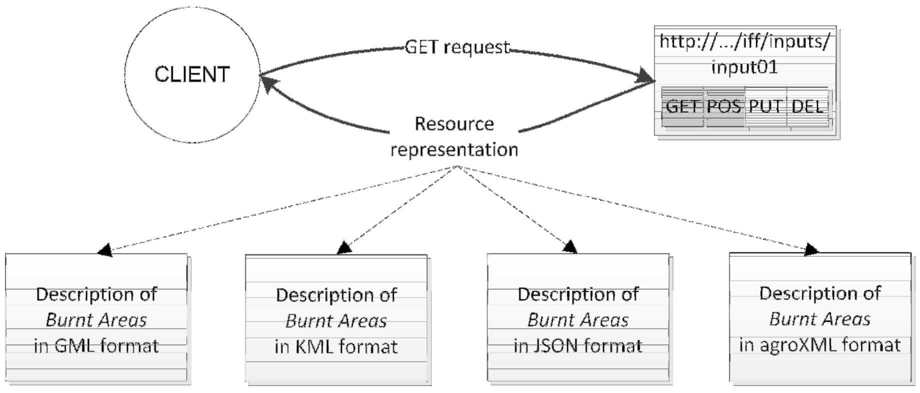

CLIENT
GET request
http://.../iff/inputs/ input01
GET
POS
PUT
DEL
Resource representation
Description of *Burnt Areas* in GML format
Description of *Burnt Areas* in KML format
Description of *Burnt Areas* in JSON format
Description of *Burnt Areas* in agroXML format

Figure 11. Different resource representations (GML, KML, JSON, agroXML) for the input resource that represent burnt areas in the impact of forest fire model.

Returning to our use case, Figure 11 illustrates four different representations for the resource *input01*, that refers to the burnt areas data fed into the Impact Forest Fire model. Depending on their particular needs, client applications can get access to burnt areas description in various formats. Selecting the most appropriate MIME type through the HTTP content negotiation mechanism opens the door to a wider range of client applications. For instance, some specialized clients may support just one response format (e.g., agroXML), while others may be interested in widely used formats such as KML and JSON. Decoupling formats and data encodings from the access interface makes RESTful services scalable and flexible for future needs. However, most service- and component-based solutions support a limited set of formats (one or two). This is because details of the access interface are often embedded in the supported format (e.g., WPS services only support XML format according to the WPS schemas).

Compared with RESTful applications, service- and component-based applications have a lack of flexibility to support and adapt to emerging data formats needs.

6.5 Resource State

HTTP headers, HTTP URIs, MIME types and format representations that support links natively are needed technologies to support and manage resource state in resource-oriented applications. Designing is the most difficult, time-consuming part which consists of the definition of relationships between resources, how these relationships may change and evolve, the set of valid transitions between resources, etc. Once all these requirements are covered, the implementation should be straightforward. Next we provide examples to highlight how RESTful developments may manage resource state.

The first example is concerned with the question we posed in Section 5.5 about what part of the internal state of a resource should be projected as a resource state in the resource representation. Michener et al. (2012) described DataONE as a network composed of inter-disciplinary, institutional nodes to support the access, sharing and long-terms preservation of ecological, earth observation and environmental data to be used by different kinds of stakeholders. Each node in the DataONE infrastructure exposes RESTful APIs[xiii] to allow users interact with available data resources. From the REST perspective, since resource state descriptions are exchanged between client-server communications, environmental scientists through 'intelligent' clients (called 'Investigator Toolkit' in DataONE terms) are able to interpret resource responses and manage properly related resources. Environmental scientists can perform several tasks such as seeking for replicas of a given data set. Essentially, appropriate representations of resource states ease the correct interpretation and further interaction with these resources without the need of a shared session (Section 5.5).

The second example is concerned with restoring previous states of a given resource. This aspect is central in IEM as mentioned in Section 3. In general, de-referencing a URI does not return a representation of prior states but a representation of the current state of that resource. Indeed, HTTP protocol lacks temporal capabilities that impede getting to an archived version of a resource on the basis of its original URI. Van de Sompel et al. (2009; 2010) addressed this issue extending the HTTP protocol with temporal capabilities. The authors described 'Memento', a protocol to support the access to time-specific versions from the current resource. The underlying idea of Memento is to extend HTTP with content negotiation in the temporal dimension. As said above, the HTTP protocol supports content negotiation in the sense that it allows clients to express preferences according to format representation and language. Van de Sompel et al. (2009) introduced the concepts of *memento* and *timegate* resources, which refer to archived versions of a resource, and provided dedicated mechanisms to support datetime-based negotiation. Clients looking for prior version of the current resources are provided with an extended Accept header that includes a specific slot for time values. Memento has been successfully demonstrated in several Web scenarios (Van de Sompel et al., 2010). Accordingly, Memento may be a suitable approach to manage time-based execution of environmental models, where previous steps may be reachable through *mementos* resources of the current resource at earlier times.

6.6 Hypermedia Protocol

The technological subtract for the typed links consider HTTP, HTTP URIs, and Web Linking. Nottingham (2010) proposed a way to include links within HTTP headers. The syntax of a link header is a set of pairs in which the *rel* attribute adds semantics to the link in terms of established relation types. A link relation type conveys the role or purpose of the link and acts as an identifier for the semantics associated with the link. This approach has already been tested for connecting geospatial services (Schade et al., 2010).

In the following, we elaborate on the kind of typed links needed to enable inter-connected resources in an infrastructure node. Given that some typed links[xiv] are already registered under IANA, as in the case of MIME types (see section 6.4), the proposed set of typed links is split into two groups. The first group (Table 1) is taken from specifications that have already been registered under IANA. We argue that all of these relations can be re-used in connecting related resources without altering their original meaning. We distinguish three roles for a given link: navigational, informational, and operational. The first two are commonplace for traditional web navigation (Leven, 2006), while relations under the operational category are specifically dedicated to controlling the resource life cycle.

Link name	**Specification**	**General semantics**	**Specific semantics (for IEM)**
service	AtomPub RFC5023 (Gregorio and de hOra, 2007)	Indicates a URI that can be used to retrieve a service document.	[Navigational] Points to the service document (entry point).
edit	AtomPub RFC5023	Refers to a resource that can be used to edit the link's context.	[Operational] Retrieves, edits and eliminates metadata of an entry resource (process metadata in Atom).
self	Atom RFC4287 (Nottingham and Sayre, 2005)	Conveys an identifier for the link's context.	[Navigational] Retrieves the collection document again.
up	Web linking RFC5988 (Nottingham, 2010)	Refers to a parent document in a hierarchy of documents.	[Navigational] Refers form entries to feeds or from process description (inputs/outputs) to entry.
latest-version	Simple Navigation RFC5829 (Brown et al., 2010)	Points to a resource containing the latest version of the context.	[Navigational] Links to latest version of a given process.
first	HTML5 (Hickson, 2011)	Refers to the furthest preceding resource in a series of resources.	[Navigational] Links to the first process in a given collection.
last	HTML5	Refers to the furthest following resource in a series of resources.	[Navigational] Links to the last process in a given collection.

prev	HTML5	Refers to the previous resource in a series of resources.	[Navigational] Links to the previous process entry in a given collection.
next	HTML5	Refers to the next resource in a series of resources.	[Navigational] Links to the next process entry in a given collection.
via	Atom RFC4287	Identifies a resource that is the source of the information in the link's context.	[Informational] Refers to the original service through the OGC WPS *getCapabilities* method.
alternate	HTML5	Refers to a substitute for this context.	[Informational] Refers to the source process description through the OGC WPS *describeProcess* method.
describedby	POWDER Perego and Archer, 2007	Refers to a resource providing information about the link's context.	[Informational] Link an Atom entry to a describe process resource in terms of inputs and outputs.
related	Atom RFC4287	Identifies a related resource.	[Informational] Not used but it is base semantics for the extended operational links in Table 2.
search	OpenSearch 1.1 (Clinton, 2011)	Refers to a resource that can be used to search through the link's context and related resources.	[Operational] Enables clients to auto-discover search interfaces.

Table 1. List of re-used typed links from other specifications

Whereas the first group (Table 1) addresses mainly navigational aspects between resources, the second group of typed relations (Table 2) is particularly centred on resource lifecycle management. The suggested relations mirror the high-level activities carried out in IEM. For instance, after executing a given resource (e.g. a forest assessment model) through the relations *execute, re-execute,* and *job*, the output results are accessible via the *result* relation. These results may not be satisfactory and scientists need a second run with slightly different values for inputs parameters (*reset*). Otherwise, the output (intermediate or final) results are good enough to be shared with the community so that they are published as new resources (*publish*), visualized via View Services (*view*), or they may be combined with other processes (*chained* and *composite* relations) because the current process is a step in a workflow. As part of the use case (McInerney et al., 2012), experiments in publishing intermediate results through WPS-interfaced services (i.e., data publication process in Figure 2) are on-going (Díaz and Schade, 2011).

Link name	**Specific semantics (for IEM)**
execute	[Operational] Triggers a synchronous execution of the associated process resource.
re-execute	[Operational] Triggers a synchronous execution after execution without changing input values to repeat the last execution again.
reset	[Operational] Combines edit (change inputs values) and execute.
job	[Operational] Triggers an asynchronous execution.
publish	[Operational] Publishes process results using publishing services.
view	[Operational] Displays process results on-the-fly using accessing services (map service).
chained	[Operational] Creates a composite process given the current results with other process member (direct composition).
composite	[Operational] Creates a composite process given the current results with other collections (indirect composition).
result	[Informational] Refers to the result of an action (search result, process execution results, etc.).

Table 2. List of extended typed links for resource lifecycle management

Figure 12 represents the hypermedia protocol, i.e., the valid interactions between clients and resources based on the widely used Atom Publication Protocol (AtomPub) (Gregorio and de Hora, 2007) and extended with the typed links in Table 1. The figure shows a directed graph where the nodes are resources and the edges represents the set of valid state transitions between such resources. For example, an instance of the generic *Invoke Category Collection* resource in Figure 12 corresponds to the *invoke* collection resource in Figure 10. From its representation, clients are able to inspect the list of forest assessment models (member resources) as well as the list of available typed relations (*service*, *self*, *search,* and *edit*). This means that only four outgoing state transitions are possible from the *invoke* collection resource: back again to the root resource (*service*), stay at the same resource (*self*), search concrete member resources (*search*) or move forward toward one entry resource (*edit*).

The meaning of a state transition is not only given by the typed link relation. Other aspects, such as the HTTP method used, should be considered because HTTP plays the role of application protocol and each method brings well-defined semantics (see Section 5.3). From the *Process Entry* node in Figure 12 that corresponds to a member resource (e.g., the impact forest fire model), the meaning of the *edit* relation varies in the context of HTTP GET, PUT or DELETE requests. First, retrieving the resource representation implies simply to send a GET request. Second, updating implies issuing a PUT request including the update representation document within the request. This allows clients to perform partial and concise updates on a given resource, such as the addition of new links to related resources (e.g., *publish* typed link pointing to the data publication process). Third, deleting a resource implies sending a DELETE request. Depending on the state transition chose (*edit* relation through GET, PUT or DELETE methods), the resulting set of available state transitions will be distinct. For instance, clients are advised of the response of a HTTP GET request tagged with the *edit* relation with a potential list of valid states (*edit, up, latest-version, first, last, prev, next, describedby, via, alternate, and execution*), which is completely different from the possible state transitions (none) after issuing a DELETE request tagged with the same typed link.

7. Summary and Discussion

Resource-oriented architectures and the designing and development of RESTful web services (Richardson and Ruby, 2007) are recently gaining much attention. In the SDI context, most of the works have been devoted to adapting data models of some OGC specifications to REST (Lucchi et al., 2008; Mazzetti et al., 2009; Foerster et al., 2011; Finney and Watts, 2011; Janowicz et al., 2012), although other attempts are gradually emerging to explore the applicability of resource-orientation in modelling geoprocessing workflows (Chen et al., 2010) and specialized geospatial services like gazetteers (Gao et al., 2010).

This paper differs from the related work in that we explore resource-based architectures in a broader sense as a hybrid approach for integrated modelling. We especially highlighted central aspects to enable interlinking resources. The definition of typed links and representation formats (documents serialized according to media types) are the basis for constructing resource-oriented applications. RESTful client applications must be intelligent enough to record the application state, as users interact with the set of resources, and help users in the decision-making process. In this sense, client applications need prior knowledge to properly interact and interpret resource representations. The meaning of each typed link, media types and representation formats used should be shared and understood in advance between clients and resource providers. In contrast to standardize service and component interfaces, RESTful approaches (that already take for granted a uniform interface) attempt to standardize representations and typed relations, which seems more affordable.

We have started some proof-of-concept experiment in adopting RESTful interfaces for geoprocessing services (Granell et al., 2012). The experiment is still very limited in scope but initial results suggest that RESTful interfaces are flexible enough to ease reuse and adaptation of geoprocessing service in varied compositions. We are not claiming with this experiment that RESTful services are better or worse than current component- and service-based solutions. We believe that resource-oriented approach may be a suitable alternative, with further experimentation, for supporting IEM and inter-linked web models.

Nevertheless, several issues for designing resource-oriented interfaces in IEM such as the handling of large amounts of data transfers and the data-intensive computations remain yet challenging (Reichman et al., 2011). Cloud computing is becoming a potential solution to address these issues, particularly to account for the intensiveness of data, computing, concurrent access, and spatiotemporal and environmental data management (Yang et al., 2011). Several research pilots (e.g. global climate change) are being conducted to operate geospatial applications in cloud computing environments and determine how to best utilize available distributed computing resources (Yang et al., 2011b). The authors presented some preliminary results which anticipate the benefits of enabling cloud computing infrastructures in data and computing intensive scenarios.

8. Future Roadmap

Starting from existing integrated modelling technologies, and recent developments considering RESTful implementations of standard geospatial processing and sensor observation services (Foerster et al., 2011; Janowicz et al, 2012), prototype systems have to be established for small and medium scales. Once such proof-of-concept experiments have been accomplished, scalability to the global dimension and robustness of the provided services has to be illustrated in the environmental sector. Such large-scale pilots should take existing infrastructures (e.g., DataONE, SDIs) and standardisation bodies, such as OGC, into account, but also need to address wider ICT and Web standards. In the end, novel solutions to integrated modelling will only become useful if the targeted audience is directly and from an early stage involved in the experiments. We believe that any innovation in this field heavily relies on the endorsement of environmental modelling (and non-ICT) experts and in the engagement with other cross-cutting domains, such as socio-economics or behavioural sciences. The technology solution proposed in this paper will have to be balanced with realistic application requirements and use cases.

Considering the technical developments we see a major need for a spiral engineering process, in which initial simple prototypes grow organically to form operational large scale implementations. Developments should address the following six central issues, which should involve intense user tests, i.e., feedback experts of all involved disciplines:

- Provision of resource-oriented use cases and experiences in environmental modelling, since these are still missing.
- Use of HTTP URIs for resource identification in environmental modelling, including canonical URIs and IM resources categories.
- Use of the uniform HTTP interface provides access and manipulation of any resource.
- Establishment of a list of widely used MIME types and formats for environmental modelling.
- Demonstration of the Memento-based mechanism for handling states in environmental models executions.
- Cloud-based solutions for handling the large data volumes and data-intensive computations which appear within environmental modelling.
- Use of Linked Data practices in resource representations to enhance inter-connected resources.

The endeavours mentioned above find support in recent European policies such as the Europe 2020 strategy (EC 2011). With this strategy, and especially under the umbrella of the Digital Agenda (EC 2011b) and Innovation Union (EC 2011c) flagship initiatives, the European Commission provided the required policy context for addressing the earlier mentioned Grand Challenges (Section 1) and for implementing a next generation of environmental information systems (Craglia et al., 2012). As part of the Digital Agenda for Europe the concept of the Future Internet (FIA 2011) summarizes efforts to deliver economic benefits from fast to ultrafast Internet and interoperable applications. The solution proposed in this paper provides valuable input to Future Internet developments, especially since it provides a lightweight solution for opening data silos, which dominate the field of environmental information sharing today. This new paradigm should start within a single existing infrastructure node with the purpose of increased connectivity between the internal resources and increasingly expand to other infrastructure nodes.

First efforts in promoting the presented solution within the context of Future Internet have already been made (Havlik et al., 2011; Roman et al., 2011) and we expect to contribute more in due time. Particularly, research for including semantic technologies, such as Linked Data (Bizer et al. 2009; Auer et al., 2009) and ontologies (Villa et al., 2009) could help to overcome heterogeneities at higher levels. We plan to follow this second line of research, mainly in order to connect current geospatial resources with user-contributed content (Granell at el., 2011).

Acknowledgements

Díaz and Huerta's work has been partially supported by the 'España Virtual' project (ref. CENIT 2008-1030) through the *Instituto Geográfico Nacional* (IGN). The authors also wish to thank Edward Pultar for proof-reading the manuscript and the anonymous reviewers for their valuable comments.

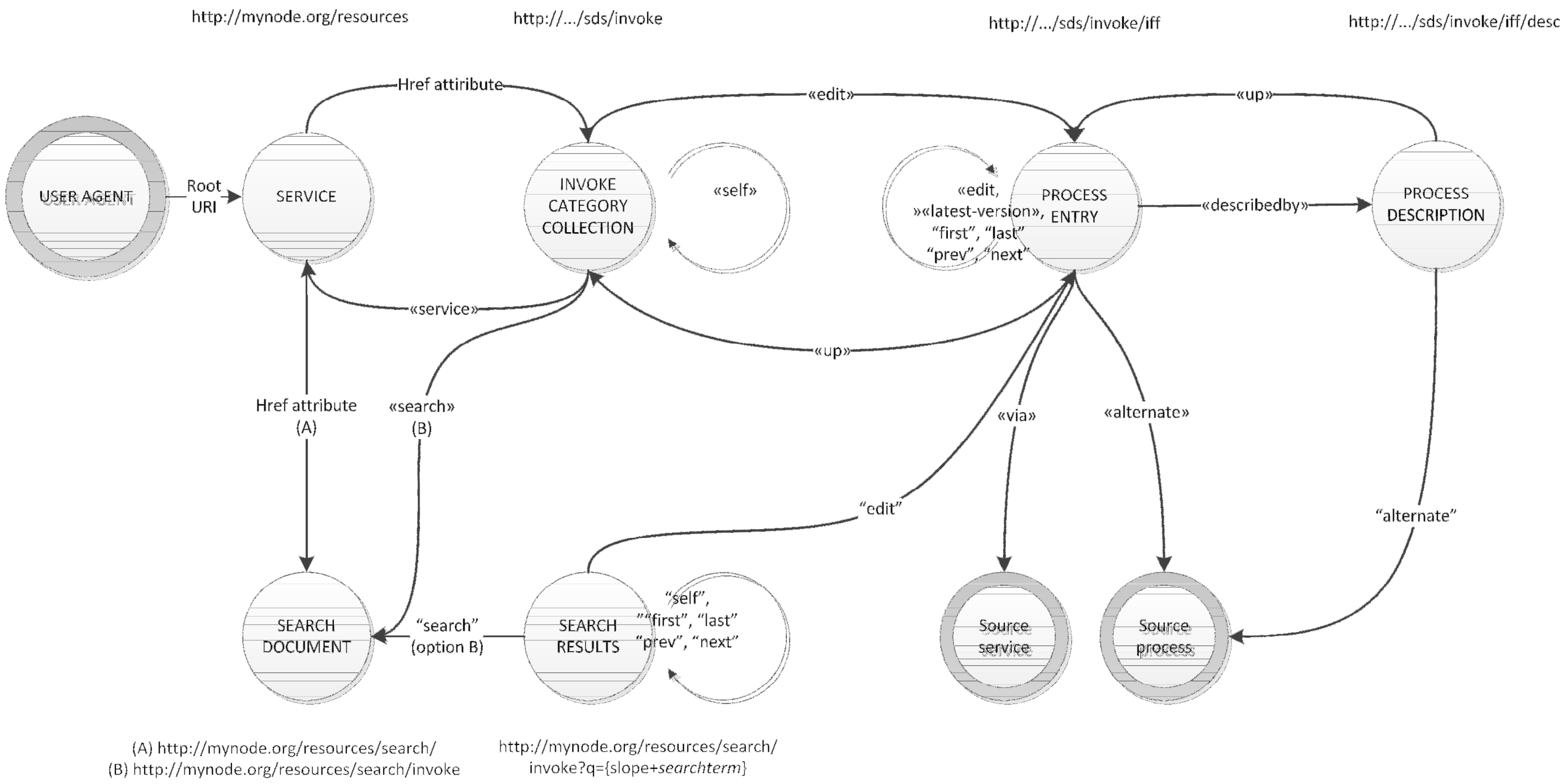

http://mynode.org/resources
http://.../sds/invoke
http://.../sds/invoke/iff
http://.../sds/invoke/iff/desc
Href attiribute
«edit»
«up»
USER AGENT
Root URI
SERVICE
INVOKE CATEGORY COLLECTION
«self»
«edit, »«latest-version», "first", "last" "prev", "next"
PROCESS ENTRY
«describedby»
PROCESS DESCRIPTION
«service»
«up»
Href attribute (A)
«search» (B)
«via»
«alternate»
"edit"
"alternate"
SEARCH DOCUMENT
"search" (option B)
SEARCH RESULTS
"self", ""first", "last" "prev", "next"
Source service
Source process
(A) http://mynode.org/resources/search/
(B) http://mynode.org/resources/search/invoke
http://mynode.org/resources/search/ invoke?q={slope+searchterm}

Figure 12. The directed graph represents the valid state transitions between resources considering the typed links in Table 1. Nodes represent resources, while edges are transitions between resources.

References

Alexandrov, G.A., Ames, D., Bellocchi, G., Bruen, M., Crout, N., Erechtchoukova, M., Hildebrandt, A., Hoffman, F., Jackisch, C., Khaiter, P., Mannina, G., Matsunaga, T., Purucker, S.T., Rivington, M., Samaniego, L., 2011. Technical assessment and evaluation of environmental models and software: Letter to the Editor. Environmental Modelling and Software 26 (3), 328-336

Allamaraju, S., 2010. RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity. O'Reilly Media / Yahoo Press

Alonso, G., Casati, F., Harumi, K., Machiraju, V., 2004. Web services: concepts, architectures and applications. Heidelberg: Springer

Argent, R.M., 2004. An overview of model integration for environmental applications-components, frameworks and semantics. Environmental Modelling and Software 19 (3), 219-234

Auer, S., Lehmann, J., Hellmann, S., 2009. LinkedGeoData: adding a spatial dimension to the Web of data. In Proceedings of the ISWC 2009, LNCS 5823, 731-746

Auer, T., MacEachren, A.M., McCabe, C., Pezanowski, S., Stryker, M., 2011. HerbariaViz: A web-based client–server interface for mapping and exploring flora observation data. Ecological Informatics 6 (2), 93-110

Barseghian, D., Altintas, I., Jones, M.B., Crawl, D., Potter, N., Gallagher, J., Cornillon, P., Schildhauer, M., Borer, E.T., Seabloom, E.W., Hosseini, P.R., 2010. Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis. Ecological Informatics 5 (1), 42-52

Bernard, L., Kanellopoulos, I., Annoni, A., Smits, P., 2005. The European geoportal - one step towards the establishment of a European Spatial Data Infrastructure. Computers, Environment and Urban Systems 29 (1), 15-31

Bizer, C., Heath, T. and Berners-Lee, T., 2009. Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems 5(3), 1-22

Brown, A., Clemm, G., Reschke, J., 2010. Link Relation Types for Simple Version Navigation between Web Resources. Internet Engineering Task Force (IETF), RFC 5829. April 2010. http://tools.ietf.org/html/rfc5829

Brunner, D., Lemoine, G., Thoorens, F.-X., Bruzzone, L., 2009. Distributed Geospatial Data Processing Functionality to Support Collaborative and Rapid Emergency Response. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 2 (1), 33-46

Castronova, A. M., Goodall, J.L., 2010. A generic approach for developing process-level hydrologic modeling components. Environmental Modelling and Software 25 (7), 819-825

Castronova, A. M., Goodall, J.L., 2012. Simulating watersheds using loosely integrated model components: Evaluation of computational scaling using OpenMI. Environmental Modelling and Software. doi: 10.1016/j.envsoft.2012.01.020

Chen, N., Zhen, Z., Di, L., Yu, G., Zhao, P., 2009. Resource Oriented Architecture for heterogeneous Geo-Processing Workflow Integration. In Proceedings of 17th International Conference on Geoinformatics, Fairfax, VA

Clinton, D., 2011. OpenSearch 1.1 specification, Draft 4. OpenSearch.org, http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft_4

Craglia, M., Goodchild, M.F., Annoni, A., Câmara, G., Gould, M., Kuhn, W., Mark, D., Masser, I., Maguire, D., Liang, S., Parsons, E., 2008. Next-Generation Digital Earth: A position paper from the Vespucci Initiative for the Advancement of Geographic Information Science. International Journal of Spatial Data Infrastructures Research 3, 146-167.

Craglia, M., de Bie, K., Jackson, D., Pesaresi, M., Remetey-Fülöpp, G., Wang, C., Annoni, A., Bian, L., Campbell, F., Ehlers, M., van Genderen, J., Goodchild, M., Guo, H., Lewis, A., Simpson, R., Skidmore, A., Woodgate, P., 2012. Digital Earth 2020: towards the vision for the next decade. International Journal of Digital Earth 5(1), 4-21

Díaz, L, Granell, C., Gould, M., Huerta, J, 2011. Managing user generated information in geospatial cyberinfrastructures. Future Generation Computer Systems 27 (3), 304-314

Díaz, L., Schade, S., 2011. GEOSS Service Factory: Assisted Publication of Geospatial Content. In S. Geertman, W. Reinhardt, F. Toppen (Eds): Advancing Geoinformation Science for a Changing World (Proceedings of the 14th AGILE Conference), Springer, LNGC, pp. 423-442

Dolk, D.R., 1993. An introduction to model integration and integrated modelling environments. Decision Support Systems 10 (3), 249-254

Donatelli, M., Rizolli, A.E., 2008. A design for framework-independent model components of biophysical systems. In International Congress on Environmental Modelling and Software (iEMS 2008), Barcelona, Spain

Donchyts, D., Hummenl, S., Vaneçek, S., Gross, J., Harper, A., Knapen, R., Gregersen, J., Schade, P., Antonello, A., Gijsbers, P., 2010. OpenMI 2.0 – What's new? In International Congress on Environmental Modelling and Software (iEMS 2010), Ottawa, Canada

EC, European Commission (2011) Official Homepage of the European Commission Europe 2020 Strategy, http://ec.europa. eu/europe2020/index_en.htm

EC, European Commission (2011b) Official Homepage of the European Commission Digital Agenda for Europe (DG Information Society and Media), http://ec.europa.eu/information_society/digital-agenda/index_en.htm

EC, European Commission (2011c). Official Homepage of the European Commission Innovation Union

European Parliament and Council, 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). Official Journal on the European Parliament and of the Council

Future Internet Assembly (FIA), 2011. European Future Internet Portal, http://www.future-internet.eu/

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., 1999. Hypertext Transfer Protocol -- HTTP/1.1. Internet Engineering Task Force, available at http://tools.ietf.org/html/rfc2616

Fielding, R.T., 2000. Architectural Styles and the Design of Network-based Software Architectures. PhD dissertation, University of California (Irvine), http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Finney K.T., Watts, D., 2011. REST-based semantic feature catalogue services. International Journal of Geographical Information Science 25 (9), 1507-1524

Foerster, T., Lehto, L., Sarjakoski, T., Sarjakoski, L.T., Stoter, J., 2010. Map generalization and schema transformation of geospatial data combined in a Web Service context. Computers, Environment and Urban Systems, 34 (1), 79-88

Foerster, T., Brühl, A., Schäffer, B., 2011. RESTful Web Processing Service. In Proceedings of the AGILE 2011 conference, Utrecht, The Netherlands

Foster, I., Parastatidis, S., Watson, P., Mckeown. M., 2008. How Do I Model State?: Let Me Count the Ways. Communications of the ACM 51 (9), 34-41

Fook, K.D., Monteiro, A.M.V., Câmara, G., Casanova, M.A., Amaral, S., 2009. Geoweb Services for Sharing Modelling Results in Biodiversity Networks. Transactions in GIS 13 (4), 379-399

Friis-Christensen, A., Lucchi, R., Lutz, M., Ostländer, N., 2009. Service chaining architectures for applications implementing distributed geographic information processing. International Journal of Geographical Information Science 23 (5), 561-580

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Software Architecture. Reading: Addison-Wesley.

Gao, S., Yu, H., Gao, Y., Sun, Y., 2010. A design of RESTful style digital gazetteer service in cloud computing environment. In Proceedings of 18th International Conference on Geoinformations, Beijing, China.

Goodall, J. L., Robinson, B.F., Castronova, A.M., 2011. Modelling water resource systems using a service-oriented computing paradigm. Environmental Modelling and Software 26, 573-582

Goodchild, M.F., Fu, P., Rich, P., 2007. Sharing geographic information: an assessment of the geospatial one-stop. Annals of the Association of American Geographers. Association of American Geographers 97 (2), 250–266

Granell, C., Díaz, L., Gould, M., 2010. Service-oriented applications for environmental models: reusable geospatial services. Environmental Modelling and Software 25 (2), 182-198

Granell, C., Schade, S. Hobona, G., 2011. Linked Data: Connecting Spatial Data Infrastructures and Volunteered Geographic Information. In Zhao, P., Di, L. (Eds): Geospatial Web Services: Advances in Information Interoperability, IGI Global, 189-226

Granell. C., Díaz, L., Tamayo, A., Huerta, J., 2012. Assessment of OGC Web Processing Service for REST principles. International Journal of Data Mining, Modelling and Management (Special Issue on Spatial Information Modelling, Management and Mining)

Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P., 2007. OpenMI: open modelling interface. Journal of Hydroinformatics 9 (3), 175

Gregorio, J., de hOra, B., (ed) 2007. The Atom Publishing Protocol. Internet Engineering Task Force, RFC 5023, http://tools.ietf.org/html/rfc5023

Harris, G., 2002. Integrated Assessment and Modelling – Science for Sustainability. In Costanza, R., Jørgensen, S.E. (Eds): Understanding and Solving Environmental Problems in the 21st Century, Elsevier, 5-17

Havlik, D., Schade, S., Sabeur, Z., Mazzetti, P., Watson, K., Berre, A., Mon, J., 2011. From Sensor to Observation Web with Environmental Enablers in the Future Internet. Sensors 11, 3874-3907

Hickson, I. (Ed.), 2011. HTML5 – A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft 05 April 2011

ICSU, International Council for Science, 2011. Scientific Grand Challenges identified to address global sustainability – International Council for Science. Pre-publication, http://www.icsu-visioning.org/wp-content/uploads/Grand Challenges_Pre-publication.pdf

Jacobs, I., Walsh, N., 2004. Architecture of the World Wide Web, Volume One. W3C Recommendation, 15Dec 2004, http://www.w3.org/TR/webarch/

Jagers, H.R.A., 2010. Linking Data, Models and Tools: An Overview. In International Environmental Congress on Environmental Modelling and Software Society (iEMS 2010), Ottawa, Canada

Jakerman, A.J., Letcher, R.A., Norton, J.P., 2006. Ten iterative steps in development and evaluation of environmental models. Environmental Modelling and Software 21 (5), 602-614

Janowicz, K., Broering, A., Stasch, C., Schade, S., Everding, T., Llaves, A., 2012. A RESTful Proxy and Data Model for Linked Sensor Data. International Journal of Digital Earth, doi: 10.1080/17538947.2011.614698

Janssen, S., M.K. Van Ittersum, K. Louhichi., P. Zander, N. Borkowski, A. Kanellopoulos, H. Hengsdijk et al. (2009) Integration of all FSSIM components within SEAMLESS-IF and a stand alone Graphical User Interface for FSSIM, SEAMLESS Report No.38, SEAMLESS integrated project, EU 6th Framework Programme, contract no. 010036-2, www.SEAMLESS-IP.org, 59 pp. ISBN no. 978-90-8585-126-4.

Jordan, D., Evdemon, J., (chairs) 2007. Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf, April, 2007

Kassahunm A., Athanasiadis, I.N., Rizzoli, A.E., Krause, A., Scholtena, H., Makowskid, M., Beulensa, A., 2010. Towards a service-oriented e-infrastructure for multidisciplinary environmental research. In International Congress on Environmental Modelling and Software (iEMS 2010), Ottawa, Canada

Knapen, M.J.R., Verweij, P., Wien, J.E, Hummel, S., 2009. OpenMI – The universal glue for integrated modelling? In Proceedings of the 18th World IMACS / MODSIM Congress, Cairns, Australia July 2009

Lee, C., Percivall, G., 2008. Standards-based computing capabilities for distributed geospatial applications. Computer 41 (11), 50–57

Leven, M., 2006. An Introduction to Search Engines and Web Navigation. Addison Wesley: Essex (England)

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

Lucchi, R., Millot, M., Elfers, C., 2008. Resource oriented architecture and REST: Assessment of impact and advantages on INSPIRE. JCR Scientific and Technical Report EUR 23397

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y., 2006. Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 18 (10), 1039-1065

Martini, D., Schmitz, M., Frisch, J., Kunisch, M., 2009. A Service Architecture for Facilitated Metadata Annotation and Resource Linkage Using agroXML and ReSTful Web Services. In F. Sartori, M.A. Sicilia, N. Manouselis (Eds): Metadata and Semantic Research, Springer, pp. 257-262

Masser, I., 2005. GIS Worlds: Creating Spatial Data Infrastructures. ESRI Press, Redlands

Mazzetti, P., Nativi, S., Caron, J., 2009. RESTful implementation of geospatial services for Earth and Space Science applications. International Journal of Digital Earth 2 (1), 40-61

McInerney, D., Bastin, L., Díaz, L., Barredo, J.I., Ayanz, J-S-M, 2012. Developing a Forest Data Portal to Support Multi-scale and Multi-disciplinary Decision Making. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (submitted)

Michener, W. K., Allard, S., Budden, A., Cook, R. B., Douglass, K., Frame, M., Kelling, S., Koskela, R., Tenopir, C., Vieglais, D.A., 2012. Participatory design of DataONE—Enabling cyberinfrastructure for the biological and environmental sciences. Ecological Informatics. doi:10.1016/j.ecoinf.2011.08.007

Müller, M., Bernard, L., Brauner, J., 2010. Moving Code in Spatial Data Infrastructures - Web Service Based Deployment of Geoprocessing Algorithms. Transactions in GIS 14 (s1), 101-118

Nebert, D. (ed), 2004. Developing Spatial Data Infrastructures: The SDI Cookbook v.2.0. Global Spatial Data Infrastructure (GSDI), http://www.gsdi.org

Nottingham, M., Sayre, R., (ed) 2005. The Atom Syndication Format. Internet Engineering Task Force, RFC 4287, http://tools.ietf.org/html/rfc4287

Nottingham, M., 2010. Web Linking. Internet Engineering Task Force (IETF), RFC 5988. October 2010. http://tools.ietf.org/html/rfc5988

Oinn, T., Greenwood, M., Addis, M., Alpdemir, M.N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M.R., Senger, M., Stevens, R., Wipat, A., Wroe, C., 2006. Taverna: lessons in creating a workflow environment for the life sciences. Concurrency and Computation – Practice and Experience 18 (10): 1067-1100.

Papazoglou, M.P., van den Heuvel, W.-J., 2007. Service oriented architectures: approaches, technologies and research issues. VLDB Journal 16 (3): 389-415

Parker, P., Letche, R., Jakeman, A. , Beck, M.B., Harris, G., Argent, R.M., Hare, M., Pahl-Wostl, C., Voinov, A., Janssen, M., Sullivan, P., Scoccimarro, M., Friend, A., Sonnenshein, M., Barker, D., Matejicek, L., Odulaja, D., Deadman, P., Lim, K., Larocque, G., Tarikhi, P., Fletcher, C., Put, A., Maxwell, T., Charles, A., Breeze, H., Nakatani, N., Mudgal, S., Naito, W., Osidele, O., Eriksson, I., Kautsky, U., Kautsky, E., Naeslund, B., Kumblad, L., Park, R., Maltagliati, S., Girardin, P., Rizzoli, A., Mauriello, D., Hoch, R., Pelletier, D., Reilly, J., Olafsdotti, R., Bin, S., 2002. Progress in integrated assessment and modelling. Environmental Modelling and Software 17 (3), 209-217

Pearlman, J., Craglia, M., Bertrand, F., Dubois, G., Fritz, S., Gaigalas, G., Nativi, S., Niemeyer, S., 2011. Interdisciplinary Approaches for Decision-making Information Across Forestry, Biodiversity and Drought. EGU General Assembly 2011, Vienna, Austria

Perego, A., Archer, P., 2007. Protocol for Web Description Resources (POWDER): Web Description Resources (WDR) Vocabulary. W3C Working Draft 25 September 2007, http://www.w3.org/TR/powder-voc/

Pratt, A., Peters, C., Guru, S., Lee, B, Terhorst, A., 2010. Exposing the Kepler Scientific Workflow System as an OGC Web Processing Service. In International Environmental Congress on Environmental Modelling and Software Society (iEMS 2010), Otawa, Canada

Rajabifard, A., Binns, A., Masser, I., Williamson, I., 2006. The role of sub-national government and the private sector in future spatial data infrastructures. International Journal of Geographical Information Science 20 (7), 727–741

Reichman, O.J., Jones, M.B., Schildhauer, M.P., 2011. Challenges and Opportunities of Open Data in Ecology. Science 331, 703-705.

Richardson, L., Ruby, S., 2007. RESTful Web Services: Web Service for Real World. O'Reilly

Rizzoli, A.E., Leavesley, G., Ascough, J.C., Argent, R.M., Athanasiadis, I.N., Brilhante, V., Claeys, F.H.A., David, O., Donatelli, M., Gijsbers, P., Havlik, D., Kassahun, A., Krause, P., Quinn, N.W.T., Scholten, H., Sojda, R.S., Villa, F., 2008. Integrated Modelling Frameworks for Environmental Assessment and Decision Support. In: A.J. Jakeman et al. (ed): Developments in Integrated Environmental Assessment, pp 101-118

Roman, D., Schade, S. and Berre, A., 2011. An Open Environmental Platform: Top-Level Components and Relevant Standards. International Symposium on Environmental Software Systems (ISESS) 2011, Brno, Czech Republic, 27-29th June 2011

Russell, N., ter Hofstede, A.H.M., van der Aalst, W.M.P., Mulyar, N., 2006. Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, http://bpmcenter.org/wp-content/uploads/reports/2006/BPM-06-22.pdf

Schade, S., Granell, C., Díaz, L., 2010. Augmenting SDI with Linked Data. Proceedings of the Workshop on Linked Spatiotemporal Data 2010 (LSTD 2010). Zurich, Switzerland. CEUR Workshop Proceedings, Vol 691, http://ceur-ws.org/Vol-691/paper3.pdf

Van de Sompel, H., Nelson, M.L., Sanderson, R., Balakireva, L., Ainsworth, S., Shankar, H. (2009) Memento: Time Travel for the Web. Arxiv:0911.1112; http://arxiv.org/abs/0911.1112

Van de Sompel, H., Sanderson, R., Nelson, M.L., Balakireva, L.L., Shankar, H., Ainsworth, S., 2010. An HTTP-Based Versioning Mechanism for Linked Data. Proceedings of Linked Data on the Web (LDOW2010). Arxiv:1003.3661; http://arxiv.org/abs/1003.3661

Verweij, P.J.F.M., Knapen, M.J.R., de Winter, W.P., Wien, J.J.F., te Roller, J.A., Sieber, S., Jansen, J.M.L., 2010. An IT perspective on integrated environmental modelling: The SIAT case. Ecological Modelling 221 (18), 2167-2176

Villa, F. Athanasiadis, I.N., Rizzoli, A.E., 2009. Modelling with knowledge: A review of emerging semantic approaches to environmental modelling. Environmental Modelling and Software 24 (5), 577-587

Tamayo, A., Granell, C., Huerta, J, 2012. Measuring Complexity in OGC Web Services XML Schemas: Pragmatic Use and Solutions. International Journal of Geographical Information Science. doi: 10.1080/13658816.2011.626602

W3C/IETF, 2001. URIs, URLs, and URNs: Clarifications and Recommendations 1.0 – Report from the joint W3C/IETF URI Planning Interest Group. W3C Note 21 September 2001

Weber, J., Parastatidis, S., Robinson, I., 2010. REST in Practice: Hypermedia and Systems Architecture. O'Reilly Media

Wiederhold, G., 1992. Mediators in the Architecture of Future Information Systems. IEEE Computers 25 (3), 38-49

Yang, C., Goodchild, M., Huang, Q, Nebert, N., Raskin, R., Xu, Y., Bambacus, M., Fay, D., 2011. Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? International Journal of Digital Earth 4 (4), 305-329

Yang, C., Huayi Wu, H., Huang, Q., Li, Z., Li, J., 2011b. Using spatial principles to optimize distributed computing for enabling physical science discoveries. Proceedings of National Academy of Sciences, 106 (14), 5498-5503

List of Figures

Lists of Tables

[i] In this paper the term IEM (Integrated Environmental Modelling) denotes here the science and technology behind the notion of integrating various models and components in environmental-related fields to support assessment and decision-making activities.

[ii] In this paper the term resource is considered in terms of informational resources and not as physical (environmental) resources, such as oil and wood.

[iii] http://effis.jrc.ec.europa.eu/

[iv] http://efdac.jrc.ec.europa.eu/

[v] Advancing Open Standards for the Information Society, http://www.oasis-open.org/

[vi] http://activevos.com/products/activevos/overview

[vii] Apache ODE (Orchestration Director Engine), http://ode.apache.org/

[viii] http://www.opengeospatial.org/pressroom/pressreleases/1450

[ix] It should be noted that the presented approach provides certainly only one possibility; other naming conventions might also be plausible. For example, Janowicz et al. (2012) proposed to assign URIs to resources by appending the resource type associated with the data model of sensor services such as *sensors/*, *sensors/thermometer1* and so on.

[x] For readability, we used here the URI fragment instead of the full URI identifier http://mynode.org/resources/sds/invoke/

[xi] http://www.iana.org/assignments/media-types/index.html

[xii] http://www.agroxml.de/

[xiii] http://mule1.dataone.org/ArchitectureDocs-current/apis/index.html

[xiv] http://www.iana.org/assignments/link-relations/link-relations.xhtml