Jun 16th, 10:40 AM - 12:00 PM

A Proposed Approach to the Development of Federated Model Sets

Kenneth M. Bryden
US Dept of Energy, kmbryden@iastate.edu

International Environmental Modelling and Software Society (iEMSs)
7th Intl. Congress on Env. Modelling and Software, San Diego, CA, USA
Daniel P. Ames, Nigel W.T. Quinn and Andrea E. Rizzoli (Eds.)
http://www.iemss.org/society/index.php/iemss-2014-proceedings

A Proposed Approach to the Development of Federated Model Sets

Kenneth M. Bryden PhD

Program Director; Simulation, Modeling and Decision Science
Ames Laboratory, US Dept of Energy, Ames, IA USA
kmbryden@iastate.edu

Abstract: Integrated modelling is a critical environmental modelling tool that brings together sets of models and data to describe the impacts of various management practices and choices within the environment. Although each model may (or may not) run on separate hardware, typically these model sets are organized together as a large integrated, centralized model. These integrated models share a single conceptual schema and semantic framework. Additionally, information flow and convergence of boundary conditions between the models are generally built on a fixed framework. Although this is a logical approach that has proven to be effective, it has several limitations. The models need to be managed and coordinated by a centralized administrator, and adding new models to the model set or revising the existing models can be time consuming and challenging. Moreover, as larger, more complex systems are modelled across various scales, the management and coordination challenges increase geometrically. This paper describes a proposed decentralized approach to the coordinated sharing and exchange of information between autonomous models. The proposed approach consists of independent models and model systems loosely federated together to represent various environmental choices. As envisioned here, a federated model consists of self-describing information entities (e.g., models, data, and user input); a library that contains the set of available models and manages the model development community; a federation schema that defines the purpose of the federation, describes the models used in the federation, and defines the topology of the federation; and a federation management system that coordinates information transport, convergence of boundary conditions, and error.

Keywords*:* Peer-to-peer ontology; federated model sets; integrated environmental modelling; model integration.

1 INTRODUCTION

Many natural and built systems are composed of interdependent parts, each of which can be represented by a separate model. Integrated modelling seeks to bring these independent models together to form a holistic basis for understanding the behaviour of the system and its component parts in response to various design and management choices (Arnold, 2013). The challenge of integrated modelling is that the integrated modelling process is significantly different from the traditional applications development paradigm. In a traditional application an individual or group of individuals work together to develop a single coherent monolithic information artefact. This artefact may or may not be a single piece of code but shares a common taxonomy, ontology, set of controls, and centralized information processing. If composed of separately conceived and developed models, these models are captured and brought into the larger systems model. Alternately, information is shared between codes using an application programming interface (API) that in many cases provides a library of specifications for how to interact with the code. A logical (and reasonable) extension of this paradigm to integrated modelling defines integrated modelling as an interdisciplinary group of domain experts working together to create a centralized model encompassing a set of other models (Arnold, 2013). Generally, these models are brought together in an integration framework. In contrast to this, the creation of the constituent models is often a decentralized process. A model is developed to

describe a particular phenomenon of interest or to answer a particular question; it can then be repurposed to a new use as needed. Integrated modelling often responds to this decentralization and independence by seeking to reintegrate the model into a system of models with a centralized ontology describing the modelling space. This centralized approach to integrated modelling has both positive and negative aspects. By ensuring consistency of the models, goals, complexity, and framework; a centralized top-down approach to integration provides a traceable consistent product. However, this requires a significant coordination effort, and many developers of models may be uncomfortable or unwilling to modify their models to meet the global ontology and semantics needed. In addition, revisions and extensions of the model may require revisions to the ontology and further discussions.

These challenges are similar to the difficulties of incorporating distributed, decentralized databases into a centralized integrated database system. Noting the challenges of decentralized databases, Heimbigner and McLeod (1985) proposed an approach to coordinated information sharing between independent autonomous databases. The key aspects of this federated architecture include the autonomous databases that determine the degree of information share and participation in the federation and the federation library or document. Each database within the federation utilizes an import schema to set the types of requests that can be made and an export schema to control its information share with the federated database. The federated architecture negotiates the activities of the federation and provides the mechanisms for information share between the components. The federated database architecture is focused explicitly on intercomponent interfaces. As noted by Heimbigner and McLeod (1985), an alternative response to the challenge of decentralized databases is to reintegrate the data into a composite or heterogeneous database using a global schema. This global schema “describes the information in the databases being composed.” The “database access and manipulation operations are then mediated through this new conceptual schema.” This approach is similar to many approaches to integrated modelling that emphasize ontological consistency and a common semantic interface (Laniak et al., 2013). As noted by Laniak et al. (2013), these needs are complicated by legacy software and need to include models from multiple disciplines, each with different work practices and goals.

By considering the degree of model autonomy and the degree of ontological and semantic independence, integrated modelling can be catalogued into four groups (Fig. 1). Model autonomy refers to the degree of independence of the models with respect to the compute platform and independent operation. Models that are compiled together and run within the same compute environment have low autonomy. In contrast models that run (or can run) on separate platforms and are linked by communication protocols are autonomous even if the same semantic and ontological schema are used. In the same way integrated models with low ontological and semantic independence use the same formal definition of the data and other information contained within the models and the same set of rules for interpreting this data (i.e., a global semantic schema). As shown

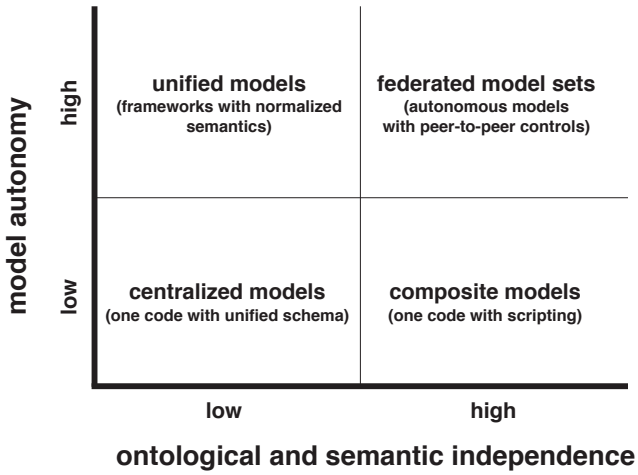

Figure 1. Model portability as a function of model autonomy and ontological and semantic independence.

in Fig. 1, model sets with low autonomy and low ontological and semantic independence can be thought of as centralized models that are basically one code, even if they are composed of multiple codes. Using various framework packages increases the autonomy of models but not the ontological and semantic independence. In many cases these codes are knit together with translators and because of this they may appear to have ontological and semantic independence. However, the translators provide the needed adherence to the global ontology and semantics. And because both centralized model sets and the unified model sets utilize a centralized ontology and semantics, they impose (at least at a high level) a pre-determined meta model that limits the ready expansion of the model set and reuse of the information entities within the model set. To provide greater semantic independence, models can be linked together with scripting to form a virtual single model. In the case of these composite models the scripting can be done on a custom basis, and hence, the ontological and semantic independence of the models can be preserved, albeit at a significant cost. As a result scripting as a linkage tool does not scale well. In contrast, if both the autonomy and ontological/semantic independence of the component models are preserved, the models can join (or leave) the federation composing the integrated model (and join multiple federations) without impacting the function of the model. There are several approaches to the creation of integrated model sets (or infrastructure interoperability) with varying degrees of ontological and semantic independence. One approach is the brokering approach in which brokers act as a middleware layer and provide the necessary interoperability, mediation, and semantic consistency (Nativi et al. 2013). Although this minimizes the impact on the model development compared to a global ontology and semantic framework, it introduces additional computational and curatorial overhead. The brokers are built to handle specific sets or classes of information. Ad hoc models, unexpected models, or simply new modelling requirements may require expansion of the broker's capability. An alternate approach that is proposed here is to create a set of peer-to-peer controls that ensure local ontological and semantic consistency between adjacent information entities and eliminate the need for translators and brokers.

This paper outlines for discussion an integrated modelling approach based on developing a novel federated architecture for model management. This architecture provides for decentralized model development and maintenance without a global ontology or schema. Rather, a peer-to-peer ontology and brokered information exchange between models is used to create constituency within a federation of models. The key aspects of this approach are

- Peer-to-peer ontology
- Development of collaborative communities based on shared interfaces
- Decentralized development of models and model federations

This architecture can be implemented within a small tightly-connected group of individuals across an organization or company with limited direct connection between developers of models and consumers of models, or at a global scale with ad hoc providers and consumers of models.

2 FEDERATED MODEL SETS

The goal of integrated modelling is to integrate a set of models from disparate disciplines together to create a holistic model capable of answering larger system level questions. In this case the term "system" extends beyond the traditional understanding of system to include traditional multiscale systems, biological and natural systems, detailed models of fluid-solid interfaces, and any other system in which two or more models are brought together to provide knowledge not available from an individual model. The term "model" has a number of meanings and uses depending on context and discipline. Here "model" is simply defined as an information entity that can be run independently of the federation and can be linked to the federation by information transfer (i.e., there is a set of inputs and outputs). This can include traditional models, databases, sensors, or any other similar information source. From this, the goals of developing an architecture for federated model sets are to provide the largest possible degree of independence for the development and deployment of the component models; support a common, light-weight mechanism for model linkage; and maintain a basis for deploying (i.e., accepting, solving, and interacting with) the integrated model. Meeting these goals requires

- *Constituency*—the capability of models to come together in groups that have coherence and substitutability
- *Articulation*—a simple and precise mechanism for describing how the models are chosen, linked, executed, and results reported

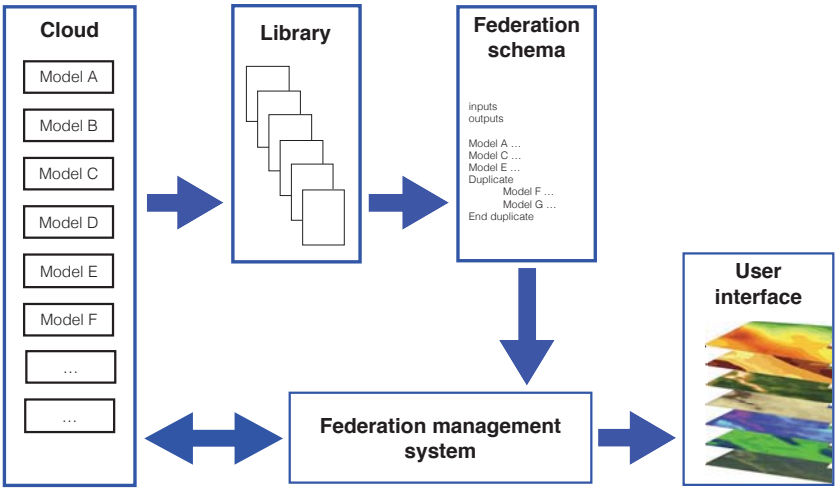

Figure 2. Components and information flow in a federated model architecture.

- *Convergence*—a knowledge of the topological mapping of the federation and the capability to route and converge information through the federation to complete the assigned tasks

As shown in Fig. 2, the proposed components of the architecture for federated models are

- *Potential federation members (models/information entities)*
- *Library that can support models*
- *Federation schema*
- *Federation management system*

Each of these components is discussed in greater detail in Sections 2.1 through 2.4. As envisioned, the overall development of the federated model system is as follows. Development of the integrated model starts with area experts who independently develop models that address specific issues. These models then incorporate standard wrappers that utilize globally unique identifiers (GUID) to provide a locally defined set of tests that confirm the ability of two or more models to negotiate the transfer of consistent data between them. Once developed, these models can apply for admission to the library. The library admits models that meet the minimal set of requirements for inclusion in a federation. These requirements include access to run the model and a wrapper that provides one or more input/output ports. Developers can update models and release new versions of the models as long as the updates meet the requirements for inclusion in the library. In addition, there may be multiple models that address the same information. As the library fills with models, the systems level developers of an integrated model can then specify via the federation schema which of the models in the library to use in a specific federated model set. The choice of models can be based on availability, accuracy, expectation of future updates, user comments and rating, and any other user defined preference. In most cases not all models in the library will be used in a specific federation. Rather, each library will support an ecosystem of federated models addressing similar questions. Once a particular federation schema is available, the federation management system will translate the federation schema into an actionable set of commands and actions that route and converge information through the federation to complete the assigned tasks and manage the user interface.

2.1 Federation members (models/information entities)

As noted earlier, models are independently developed and maintained. The developer of the model may choose have their model join (or not join) a particular federation library. This independence is one of the primary difficulties in the federated model concept. Specifically, how will individual models know that they can talk to each other and how will the federation management system know what kinds of data translations are necessary to make a given connection happen? As shown in Fig. 3, a model in this system consists of the underlying application, a container that makes the underlying

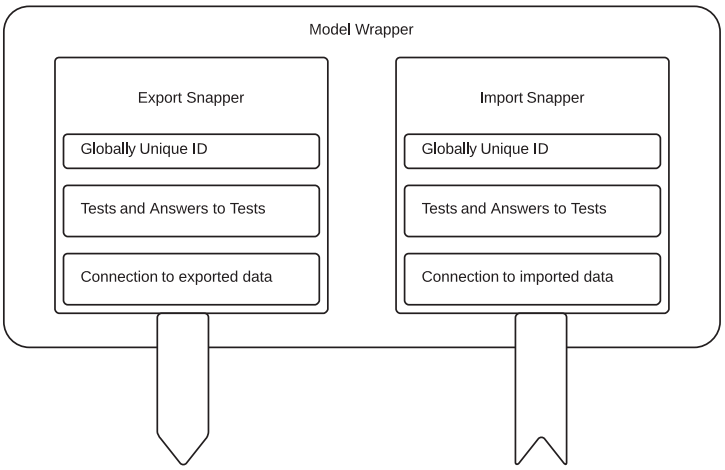

Figure 3. Structure of a component model.

application runnable in an unknown cloud computing environment, and an interface (or wrapper) containing one or more export schemas and one or more import schemas. These import and export schemas need to be able to answer the questions “does it make sense to connect my export/import schema to this other import/export schema?” and “what data translations are necessary?” That is, how do we know that we are talking about the same variable in the same way? How these questions get answered and how a model knows that they have been answered satisfactorily is a critical component of the federated model concept. The following structure is proposed for this negotiation.

Each model wrapper has a “snapper” for the data it exports and another “snapper” for the data it imports (Fig. 3). Also each model may potentially have a large set of input and export snappers, any of which can fulfill its import/export needs fully or partially. Each snapper has a GUID (and possibly version information) that allows the management system to figure out whether it is possible to perform the tests that determine whether an import and export snapper can fit together. As shown in Fig. 4, if the export and import snappers share the same globally unique identifier, the shared tests and answers can be used to determine if the two models can be connected. The list of GUIDs can be stored in the library, and a new GUID can be proposed and approved by model developers as needed. Each of the GUIDs establish a local ontology, and the model wrappers are written in a peer-to-peer fashion. The users are free to build snappers and wrappers as they choose. No overarching ontology or set of standards are necessary. Each developer of a model wrapper is responsible for deciding the sets of questions and answers (keyed by GUIDs) their wrapper will support. This will impact what other models they are able to connect to. This approach results in sets of models that can work together to form various federated model systems. This peer-to-peer ontology only requires

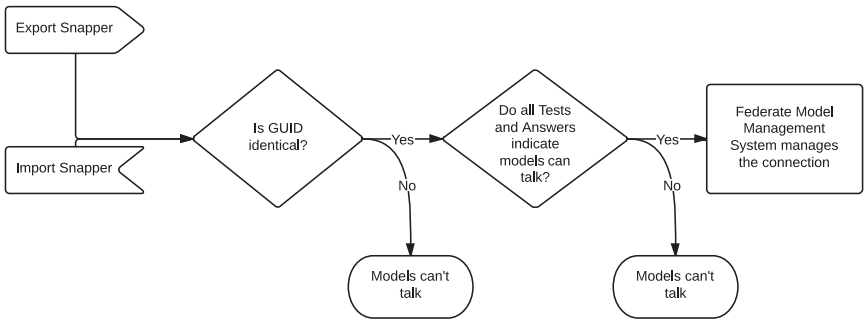

Figure 4. Connection process.

that developers of models that will exchange information agree on consistent meaning and representation of the information to be exchanged (i.e., a local ontology). This peer-to-peer ontology need only exist for the data to be exchanged. Consider the case where Model A exchanges information with Model B (perhaps area in m^2), and Model B then uses this information and sends information (perhaps cost data in $) to Model C. Model C then uses this information in yet another calculation. No global meaning of length, area, or cost is needed, and other models in the federation may use these terms in entirely different ways. In addition, Models A and C do not need to use the same ontology or understand each other. They only need to communicate with Model B.

2.2 Library

To support this process, a community hub needs to exist that hosts models, translators, and model wrappers; tracks GUIDs; and streamlines the process of acquiring a model, encapsulating it in a wrapper, running it on the cloud or on an internal server/cloud, and connecting it with other models as desired. This library should be capable of analyzing the network of GUIDs and running the tests of each snapper with other corresponding snappers that share the same GUID to determine which models can be part of the same federated model system and what connections are valid. Part of the success of the federated model concept will be determined by how easy it is to go from knowing what system needs to be created to finding and connecting the right models and to deploying the federated model system on the cloud. The library is the central element where developers, integrated modelers, and others meet.

Of particular concern is encouraging the participation of the domain experts who create the models. Much of the challenge of connecting independent models is a people problem not a software problem. Although we can develop a wide range of ontologies and architectures to support integration modelling, if the domain experts developing and maintaining the models cannot easily cooperate, it will be hard to gain their willing participation. And without the participation of a large number of domain experts, it will not be possible to develop a rich and broad base of models needed to support this vision of integrated modelling. These domain experts have a number of reasons not to participate. They may wish to control how their models are used, preparing the models for wide application is additional work, and they may not find value in integrating their models with others' models. Because of this, creating a user friendly, open, and easily understood library system that provides the lowest barrier to entry is essential to pursuading the domain experts that write models to work together and to encourage them to do so from a position of self interest. The library is a community hub that stores all open source and closed source model wrappers (and can be deployed internally at a large corporation) than can encourage this behavior. Ideally, the barrier to connecting a model with another model (or even a large number of other models) will become so low that model creators will eventually consider a federated model system to be the default execution environment for their models.

2.3 Federation schema

The federation schema provides the language and process for creating, documenting, and updating the federation of models. To do this, a grammar is needed that provides for an expected set of operations within the federation of models. It is anticipated that this set of commands will not specify the order in which the models are run but rather will specify which models in the library are to be used and to which models each will connect with (i.e., the neighbours for each model). At first glance it seems that specifying the neighbour models is the same as specifying the order of model execution. However, specifying the neighbours specifies the directed graph that represents the information flow within the integrated model. Once the directed graph representing information flow is developed, various tools can be used to identify strongly connected components within the federation (McNunn and Bryden 2014) and to develop the needed blocking and scheduling to run the integrated model.

2.4 Federation management system

The federation management system utilizes the federation schema and the library to develop the needed plan and to execute the model. This includes brokering the needed communications between

models, securing the needed resources, scheduling the run, and converging the shared variables within clusters of strongly connected components. The specific needs of the federation model management system include fault tolerance, ability to control access to the model, and assessment of error.

A critical property of both the model wrappers and the federation management system is that the models and their wrappers need the ability to either run on cloud computing resources or be generally available as an internet connected API (in the case where a company or other entity wants to make the model available as a service). This means that models are likely to be provisioned in a wide variety of different ways including provisioning API usage (for models that are sold as a service), deploying virtual machines, and deploying lightweight virtualized environments such as docker (www.docker.io). The federation management system needs the ability to manage these and other methods to provision the resources to execute the models and to broker the needed communication between them. This provisioning needs to occur in such a way that errors are found and corrected and such that financial costs of computing and API usage are estimated, minimized, and limited.

3 OBSERVATIONS

The federated model architecture discussed here enables the system level developers of an integrated model to focus on the interaction between the component models rather than the development of the component models. This is probably one of the most critical aspects of integrated modelling. The outputs of many models are very sensitive to the inputs (i.e., the boundary conditions). And as the outputs of one model become the inputs of another model, and so forth through a set of models, building an infrastructure that explicitly focuses on the “in between” enables users to examine the impact of various linkage assumptions, error propagation, and other system level questions. In addition, groups of strongly connected components can be readily identified, and methods for converging the inputs and outputs specific for the set of models can be developed.

Because models and sets of models can readily be added or deleted from the integrated model, alternate representations of the information (e.g., reduced order models, data tables, acceptance of less accuracy or sensitivity) can be developed and explored and may simplify the information flow or otherwise improve the integrated model. In addition, the performance of components represented by different models can be examined within the same integrated model. This can provide a mechanism for examining different components or management practices, or of simply comparing two models. That is, it can provide a basis for the exploration and “what if” process of design.

Within web development the term “mashup” describes a web application that is composed of information from multiple sources that in many cases were not developed for the current use but have been integrated together to create a new application (Benslimane et al., 2008). The easy integration of these sources often uses application programming interfaces (APIs). API is a generic descriptor of a set of instructions about how various software components can interact with other software components. In many ways the integration tools (e.g., GUIDs and tests) discussed here are a form of an API that can automatically negotiate linkage and integration for complex models. APIs are a well developed and commonly used software tool with multiple specifications and libraries. This suggests that much of the work and standards used in APIs can be examined and used in the development of the integration standard.

As noted earlier, one of the key features of this proposed architecture is the use of a peer-to-peer ontology to provide a high degree of independence for model development. This requires that the teams developing a particular model work closely with the development teams of neighbouring models to develop the shared ontology and tests. While there are several models for how this developer community can be established, one appealing model is the one used by GitHub™. GitHub™ provides a deeply integrated social aspect for collaboration on both open and closed source projects. In many development communities someone seeking to fork an open source software project is most often viewed as a threatening event by the current development team. This means that to contribute to an open source project, new developers need the permission of the project maintainers or must commit to running and maintaining their own fork. The model used for collaboration on most projects in GitHub™ is “Fork and Pull” (https://help.github.com). Under this

model it is easy for anyone to make a personal fork of a software project. A “pull request” then asks that the fork be committed to the original repository. This allows the original author of the software to consider the pull request, discuss them with team members or the requester, make changes to the pull request, and finally commit or reject all or part of the request. This approach has lowered the barrier to entry for fixing a problem with an open source software package. A similar mechanism could be developed for the wrappers within the federated model environment. A specific set of protocols based on an integrated social network could help an unrelated model developer team ask another model development team to consider a needed snapper for the two packages to share data. Both teams can review the request, negotiate the representation to be used, and adopt or reject the final snapper.

Beyond the implementation of an integration framework for a simple set of models, there are a large number of open questions. None of these are roadblocks that cannot be overcome, and in many cases, they are the same challenges that other integrated modelling efforts face. Although many of these are questions of database management and informatics, there are a number of computational science questions. These include multiscale modelling tools, multidimensional interfaces and grids, transient models, convergence and information share between adjacent domains, and other computational challenges. In all cases these challenges can be handled on an as needed basis as the framework develops.

4 CONCLUSIONS

This paper describes a proposed decentralized approach to the coordinated sharing and exchange of information between autonomous models. The proposed approach consists of independent models and model systems integrated together in a federation of models to represent various environmental choices. As envisioned here, a federated model consists of self-describing information entities (e.g., models, data, and user input); a federal dictionary that describes the model set including the topology and schema, and that admits members to the federated model set; and a conductor that coordinates information transport, convergence of boundary conditions, and errors. Key to this proposed framework is the development of a peer-to-peer approach to establish a local ontology. The goal is substantially reduce the cost to model developer. This framework is currently under development and has not yet been implemented. This paper provides an introduction to this concept and an invitation to collaborate in the development of this architecture.

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy (US DOE), Office of Fossil Energy. The research was performed at the Ames Laboratory, which is operated for the U.S. DOE by Iowa State University under contract # DE-AC02-07CH11358.

REFERENCES

Arnold, T.R., 2013. Procedural knowledge for integrated modelling: towards the modelling playground. Environ. Modell. Softw. 39, 135–48.

Benslimane, D., Dustdar, S., Sheth, A., 2008. Services mashups: the new generation of web applications. IEEE Internet Comput. 12(5), 13–5.

Heimbigner, D., McLeod, D., 1985. A federated architecture for information management. ACM T. Off. Inf. Syst. 3(3), 253–78.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., Peckham, S., Reaney, S., Gaber, N., Kennedy, R., Hughes, A., 2013. Integrated environmental modelling: A vision and roadmap for the future. Environ. Modell. Softw. 39, 3–23.

McNunn, G.S., Bryden, K.M., 2013. A proposed implementation of Tarjan’s algorithm for scheduling the solution sequence of systems of federated models. Procedia Comput. Sci. 20 (2013) 223–8.

Nativi, S., Craglia, M., Pearlman, J., 2013. Earth sciences infrastructures interoperability: The brokering approach. IEEE J. Sel. Topics in Appl. Earth Observ. 6(3), 1118–29.