INTEGRATING MODELS ON THE WEB: APPLICATIONS FOR SOCIO-ENVIRONMENTAL STUDIES

Getachew Feleke Belete

Graduation committee:

Chairman/secretary

Prof.dr.ir. A. Veldkamp	University of Twente

Promoter

Prof.dr. A.A. Voinov	University of Twente

Co-promoter

Dr. J.M. Morales Guarin	University of Twente

Members

Prof.dr.ir. A. Veldkamp	University of Twente
Prof.dr. M.R. van Steen	University of Twente
Prof.dr. K. Hasselmann	Max Plank Institut for Meteorology, Germany
Prof.dr. M. Donatelli	Research Center for Industrial Crops, Italy

ITC dissertation number 299
ITC, P.O. Box 217, 7500 AE Enschede, The Netherlands

ISBN 978-90-365-4306-4
DOI 10.3990/1.9789036543064

Cover designed by Job Duim
Printed by ITC Printing Department

UNIVERSITY OF TWENTE.

FACULTY OF GEO-INFORMATION SCIENCE AND EARTH OBSERVATION

INTEGRATING MODELS ON THE WEB: APPLICATIONS FOR SOCIO-ENVIRONMENTAL STUDIES

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr. T.T.M. Palstra,
on account of the decision of the graduation committee,
to be publicly defended
on Thursday, February 23, 2017 at 12:45 hrs

by

Getachew Feleke Belete

born on July 4, 1976

in Bahir Dar, Ethiopia

This thesis has been approved by

Prof.dr. A.A. Voinov, promoter
Dr. J.M. Morales Guarin, co-promoter

Dedicated to my extended family.

Acknowledgements

When I sit down to write this section I am emotional with the immense feeling of gratitude. First and foremost, I would like to express my deepest gratitude to the Almighty God who guides and protects me through the journey of my life. My PhD research would never come to its end without the contribution and decisive assistance from numerous people.

My deepest acknowledgment goes to my promotor and supervisor Prof. Alexey Voinov, who gave me the opportunity to carry out this research at University of Twente, faculty of ITC in the Department of Geo-Information Processing (GIP). Alexey, I am indebted to you for sharpening my insights. Your critical comments and suggestions have been valuable to my research work. I have gained invaluable scientific paper writing skills and learned how to deal with reviewers' comments. Thanks for your patience! I would also like to thank my second supervisor Dr. Javier Morales for your valuable feedback in shaping the thesis and for the warm and friendly treatment.

This research is realized with the financial support of European Union FP7 COMPLEX project. I am grateful to the project leader Prof. Nick Winder of Newcastle University and all project members from 17 different institutes of Europe for the valuable inputs at different levels of this research. I would also like to express my special gratitude to COMPLEX project members: Dr. Tatiana Filatova, Dr. Tatyana Bulavskaya, Laila Niamir, Dr. Kishor Dahlave, Dr. Inaki Arto, Dr. Saeed Moghayer, Dr. Cheryl de Boer, Dr. Dmitry Kovalevsky and Dr. Richard Hewitt for the friendly team work in conducting the case study.

I acknowledge Dr. Niels Holst from Aarhus University, who paved my way to study in ITC. Niels, your advice and support have also continued during my PhD research by co-authoring a paper. Your support and encouragement is not limited to academics, but also to personal and family matters. You have been my real motivator. I am fortunate to have you as close friend.

I am very grateful to all members of GIP department for supporting through the different phases of the research by providing valuable feedback from the PhD proposal to the final thesis. The GIPrePUBlic PhD group of our department, we really have an unforgettable time together. I would also like to thank my dear colleagues who are studying their PhD in ITC in general for the friendship and for sharing experiences through emotionally challenging times. Dr. Rahul Raj and William Lee, it was joyful to share the same office with you both. I would also like to acknowledge the assistance from ITC support, helpdesk, finance, and travel unit staff for their charming services.

ITC-Habesha community friends, I would like to thank you all for the memorable and enjoyable times together. Dr. Mesfin and his wife Meron, you have really helped us a lot in adapting life in Enschede. Dr. Abebe M., Dr. Tagel G., Dr. Dawit W., Atkilt G., Dr. Birhanu k., Berihu A., Adugna M., I really enjoyed our discussions during coffee break time. Fasil, you are always there to help anyone from Habesha community, keep the good job. Janet, Amina, and Feven you have done a wonderful job by being close friends with my family here.

My close friends back home Leul, Daniel, Belaya, Dereje, Kuji, Tewodros, Solomon M., Solomon B., Abush, Fiseha, and Eshetu thank you for being there for me all the time. Eyob G., Adamu B., Mulumebet A., Titi A., Mimi A., Tadewos A., Helen, Yered A., Tesfaye A., and Alemu sheferaw, I am lucky that I have you all. I am also grateful to my former colleagues in United Bank and Addis Ababa University for continuous encouragement. Daniel B., Michael M. and Ashenafi K. thank you for the warm reception during my conference trip to the U.S. Helen Mutaher you are really special person, my family will not forget the favor you did to us, thank you!

My parents, my father Feleke Belete and my mother Simegn Workneh, I have reached this stage because of your continuous encouragement and your never-ending prayers. You have taught me to keep faith in God and myself, and to keep trying. My very special sisters, Mulumebet and Birtukan and my dearest brothers Solomon, Habtemariam, Abraham, and Amanuel I have no words to express the courage and motivation you have been offering me during my study.

Last but not least, my beloved wife Bethlehem Assefa, you have been illuminating my life by being the closest friend, caretaker, and personal advisor. Although it is not your field of study, you are always happy to hear my explanation of 'unnecessary' technical details and to discuss the various challenges I faced during my study. Your comments, encouragement, and motivation have been invaluable. My daughters Blen and Arsema, you are blessings from God. You have been the source of my inspiration to be hopeful and to focus on the good side of life.

Finally, I would like to state that any omission in this brief acknowledgment does not mean I am not grateful to you.

Table of Contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

Most socio-environmental problems are wicked problems that are difficult or impossible to solve because of incomplete, contradictory, and changing requirements and information about the systems. The number of people and opinions involved and the interconnected nature of a problem with other problems complicate the process of providing a solution. Depending on the specific context we may need to involve people, who are causing the problem, and are affected by the problem. These will have to be brought to some kind of compromise. They may be representing interest groups, experts in the specific subject, and political decision makers. In any case, engaging stakeholders in the problem solving process is essential when dealing with wicked problems. The challenge is that it requires skills to coordinate the collaborative process and a platform to combine and analyze the different inputs of stakeholders.

Modeling has been recognized as an important tool for analyzing complex systems. A model can help to explore and experiment on the system without destroying it at the same time (Voinov, 2008). Models are developed with the objective of understanding a system, forecasting its behavior, or communicating how the system works. Thousands of models are developed for one or more of those purposes. However, there still is and will always be a need for better models, because socio-environmental problems are getting only more complex and urgent, while more and more systems become dominated and controlled by humans and require careful management and timely decision-making.

Modeling is an iterative and incremental process. When analyzing complex systems, the model tends to incorporate broad and detailed representations of the various processes that take place and the numerous elements and factors that play a role in the system. This can be done, either by incorporating the required features of the system into one rather complex, integral model, or by combining various already existing models into an integrated model (Voinov and Shugart, 2013). Integral models are commonly built from scratch based on one particular modeling paradigm, for certain decided temporal and spatial scales, by the development team consisting of a few people with specific disciplinary background, which may have its effect on the representation of complex systems that span across disciplines.

In the integrated approach, instead of developing a model of the whole system, we assume that the system is made out of components, which can be modeled separately. These component models may be developed by different teams, using the best knowledge and tools available in various disciplines. This also means that we may need to deal with different temporal

and spatial scales in the components. The integrated approach is based on reuse of existing models, and as a result we will not need all the expertise required to describe the different parts of the system. This can save us time, resources and minimize the need for reinventing models. We can also link models developed using different modeling paradigms, for example, an Agent Based model may be coupled with a Computable General Equilibrium model.

Before going further we define an integrated model as a model that links together two or more independent component models to complement each other and form representation of a certain system (Hohpe and Woolf, 2004). Similarly, integration of models is defined as the process of linking two or more models so as to produce the aspired combined representation of a system. The assumption is that it makes more sense to use existing well developed and tested models as building blocks rather than build the whole system each time from scratch. We see model integration as an option to seamlessly explore the different aspects of the represented system and to improve our understanding by switching various modules on and off and testing the overall system sensitivity, both parametrically and structurally. This requires transparency and flexibility of the model integration process, allowing stakeholders to play a more significant role in deciding what modules are to be linked, how they should be treated, and what scenarios should be analyzed.

In fact, integration of models has challenges that arise from several dimensions. We assume that the models we are considering for integration are simplified representation of systems that are coded into computer language for execution on a computer (Whelan et al., 2014). Commonly, significant amounts of critical information about the underlying principles and assumptions of a model are not provided with the model source code, they remain in the mind of the modelers. Probably some of this information can be provided using detailed model documentation. However, in much too many cases model documentation is not sufficient; it does not incorporate ample information about the model. In such situation we may need to fill the information gap by engaging in direct discussions with the modelers and then using this information learnt when linking the models. Integration requires exploring incompatible and complementary knowledge embedded in the participating models (Hamilton et al., 2015). It also requires reconciling expert suggestions of modelers (Vitolo et al. 2015), which may not be provided with documentation of models. As a result, integration of models is difficult, especially if the models are poorly documented.

The other challenge is that integration of models needs linking independently built models and translating and transferring knowledge between multiple science domains (Laniak et al., 2013). Integration requires aligning different

assumptions and logic embedded in the participating models, which becomes much harder to do when models come from very different disciplinary backgrounds. Knowledge integration assumes agreeing upon a representation of the system and conceptualizing it, reconciling one kind of representation of a concept with another, and analyzing and presenting the output. After these steps the integrated system can provide combined representation of the required system. This will help users to manipulate the integrated system regardless of the number of internal domains, subsystems, and system boundaries.

Besides, integration requires establishing data exchange between stand-alone models. The models we are considering for integration can be developed using different tools, which may not naturally interoperate. The models may also be located on remote computers with various operating systems. Establishing interoperability across different modeling tools that can span over different hardware and software platforms needs applying advanced software engineering techniques. In addition, meaningful data exchange can be achieved only after matching the different contexts used by the participating models. This needs semantic mediation between concepts embedded in the models. Due to these reasons integration of models requires that a modeler and a software engineer play complementary roles and closely interact. However, in most of the cases the development of integrated models seems to focus on modeling, the use of software engineering methodologies is limited (Verweij et al., 2010). On the other hand, software engineering had been changing rapidly and providing new techniques for interoperability and for high-performance, distributed, and cloud computing (Syvitski et al., 2011). We need to improve the model integration process by making the best use of recent advances in software engineering (e.g. design, accessibility, interoperability) and semantic processing (e.g. matching of concepts, searching).

We also see that modeling from purely scientific applications has moved to decision support and policy impact analysis (Verweij et al., 2010). This necessarily implies that stakeholder participation becomes increasingly important and should be recognized when developing tools for integrated modeling. Ideally, the integrated system should be able to serve as a platform in which stakeholders will merge their differing worldviews and priorities, and have quick and easy access to transdisciplinary information when dealing with complex problems. It should serve for "What-If" analysis for multiple choices of parameters and scenarios, for various time periods, or events and also should promote co-learning and co-management with large numbers of stakeholders involved. The integrated system should be evolving and adaptive to meet the dynamic requirements of stakeholders.

To meet the current needs of integrated modeling applications we should consider the context in which such systems are operating. On the one hand, technically, integrated modeling applications are a set of software libraries, classes, and components assembled together to deliver a range of services (Rizzole at al., 2008). On the other hand, the models we are considering for integration are rarely designed to communicate with other models and they use different ways of representing information or ontologies. Integration of independently developed component models can result in some unexpected and unintended results (Voinov and Shugart, 2013). For example, error and uncertainty in one of the components can easily propagate through the whole system generating output that will be difficult to interpret and debug. Due to this, integration of models is constrained by both technical and conceptual challenges (Argent, 2004). To address these challenges integrated modeling systems should include mechanisms for interaction of model components, semantic mediation, time stepping, up or downscaling of data, cross-platform and cross-language interoperability, interfacing with a variety of packages, multiprocessor support, and exception handling (Hohpe and Woolf, 2004; Syvitski et al., 2011). At the same time they have to make potentially quite complex and sophisticated models accessible for stakeholders.

One way to remove technical and accessibility constraints and to facilitate stakeholder participation is by presenting the models as web applications, making them available through standard web browsers. We recognize that there is also an international effort to share model components to a wider audience on the web. For example, Model Web is a generic concept lead by NASA, IEEE, the European Commission, and the National Research Council of Italy (CNR) with the aim to improve model accessibility and interoperability (Nativi et al., 2013). However, there is a lack of semantically enabled searching and linking capabilities of data and models, and as a result the reuse of these resources is still limited.

This research work can be considered as part of the international effort to make models available over the web to a wider audience. We performed this research with the objective to develop a methodology and software design that enables to transform independently developed models into web-based interoperable components and further link them into integrated models. We also investigated automatic semantic mediation of data and metadata of models. Our research can open up the modeling process for many stakeholders who are not prepared to deal with all the technical difficulties associated with installing, configuring, and running various models.

With the advent of new gadgets there is also a need to provide access to modeling applications from mobile devices. "Apps for the environment"[1] developed by the U.S. Environmental Protection Agency is a good example. In fact, this can be realized by moving computationally intensive tasks to servers and by providing flexible web-enabled user interfaces that can be accessed anywhere from the Internet by using various devices. The current momentum to leverage the World Wide Web for integrated modeling applications by using cloud computing and web services can be further extended to make models accessible from mobile devices. We believe that the results of our research can be used in the future to make models available through various mobile devices.

1.2 Motivation

Every year millions of euros are allocated to projects that investigate complex socio-environmental problems. Such projects commonly bring together various stakeholders that have differing views and priorities. Modeling tools can play a very big role in finding consensus and reconciling stakeholder views and priorities by serving as a platform to understand how systems work, to experiment with various scenarios of system development, to analyze possible actions, and develop shared planning strategies. However, despite the large number of available models, stakeholders may find it difficult to use them. On the one hand, locating, installing, configuring, and operating the models and their underlying software platforms may be hard for people without extensive computer training. On the other hand, understanding and using models in coherent way is hard, especially, as seen above, when they are poorly documented. Such platforms as eHabitat by European Union Digital Observatory for Protected Areas Models, that do not require installation of special software, and that do not need special training, are more accessible but rare. We need a mechanism to make complex models accessible for stakeholders.

Scenarios proposed by stakeholders commonly require analyzing the problem from different perspectives and this can be done using a modeling system that is flexible enough to easily accommodate changes in the model(s) used. We see integration of models as a means to provide flexible and transparent representation of complex socio-environmental scenarios. However, this requires addressing two major issues on integration of models.
The first issue is that integration of models has been practiced for a decade but there is lack of a documented model integration methodology that will guide the integration process. Lack of guiding methodology has let modelers to practice the model integration process in widely different ways, and this has effect on the length of time of the project, transparency and

[1] Apps for the environment - http://www.epa.gov/mygreenapps/

reproducibility of the process, and flexibility of the integrated system. We observe these differences in recently published literature too. Developing a model integration methodology requires defining the scope, identifying the key processes involved, characterizing each process, and organizing them in a logical manner. Integration is a modeling process that requires higher level of software engineering processes than the conventional modeling process. The modeling process involves system conceptualization, formulation of the model, calibration, verification, and sensitivity analysis. However, integration requires software engineering processes such as requirements analysis, system design, implementation, testing, and maintenance. Each of these development cycles cannot accommodate the model integration process as a whole. This motivated us to develop a methodology that could facilitate the life cycle of the model integration process.

The second issue is that integration of models requires linking independently developed models so that they can complement each other and function as a single coherent system. However, most of the models are not designed to interoperate with each other. They are also not easily available for exploration and use. Due to this integration needs creating interoperability between stand-alone models. Lack of interoperability is hindering the development of integrated modeling systems. This has inspired us to investigate the different levels of interoperability that are necessary to link legacy models into integrated system. We want to leverage the latest web technologies to transform legacy models into web-enabled interoperable components. At the same time this will help to access and combine models that are distributed on different locations into an integrated system.

A quest for a generic model integration framework that can link models in the graphical interface level is the other point that inspired us to perform this research. Indeed, aiming at generic integration framework is an ambitious target since it needs availability of standardized model metadata and full-fledged ontology, which is still work in progress that requires acceptance and much collaboration within the modeling community. However, we would like to contribute towards generic integration frameworks by investigating the techniques that help to manipulate models in a generic way and can also automate some of tasks of semantic mediation.

1.3 Study objectives

We performed this research with the objective to develop a methodology and to design software that can transform independently developed models into web-based interoperable components and further facilitate linking them into integrated models that can represent complex socio-environmental systems.

In order to address this objective the following guiding research questions were formulated:

- How to develop a methodology that facilitates the model integration process from requirements identification to system testing?
- How to design a web based model integration framework that links independently developed multidisciplinary models into an integrated system?
- How to reconcile implicit semantic relations in integration of models?
- How sensitive are integration results to time steps, numeric integration methods, and functional responses assumed in the component models?
- How can we apply integrated modeling approach to actual case study of complex socio-environmental scenarios, such as climate change mitigation actions?

1.4 Case study description

A case study was chosen to improve our understanding of the model integration process through actual practice of integration and also to test our findings as a proof of concept. The research was conducted in the context of the COMPLEX project, which is a European Union FP7 project with the objective of investigating alternative policy scenarios for the transition of Europe to low carbon society by the year 2050. The project included 17 partner organizations from 11 countries, organized into 7 work packages (WP) each comprising people from different organizations. The project had a suite of models developed using various modelling approaches – integrated assessment (IAM), system dynamics (SD), computable general equilibrium (CGE) and agent based (ABM). The models operate at different scales: global, country, regional and individual (household/firm). The project aimed at organizing these models to investigate various policy goals and scenarios. Integration of models was one of the strategies used to organize models over a hierarchy of scales, where models in one scale could inform models in other scales.

The inputs from various WPs came as models, data, results of stakeholder analysis, scenario specifications, and expert definitions of possible critical changes in the operation of the socio-environmental systems. Consider the following points to illustrate various inputs:

- Investigating climate related energies. Modeling energy production from rainfall, wind, and sun and its interaction with agriculture and also the analysis of the use of climate related energies in relation with climate change, evolution of energy demand, alternative energy equipment, and environmental policies.
- Exploration of the acceptance, implementation, and realization of climate mitigation policy options. Policy research on implementation of green

technologies, such as wind, solar, biofuel production and other forms of biomass use.
- Analyzing socio-economic and land use dynamics in the Stockholm-Mälar region, including analysis of emerging and optimally selected land use patterns and economic development and the impact of policy at shorter and longer time scales.
- Investigating non-linear climate responses and regime-shifts of economic-ecological systems, modeling non-linear processes, and representation of economic sectors with a significant potential for mitigation and resource efficiency.

By coordinating the above-mentioned inputs the project was to integrate models and knowledge generated in various case studies so that it will represent the climate-energy-economic system at different levels. This required building and analyzing the hierarchy of models, which were developed and applied within the project, and, beyond. To realize this objective the following milestones were set by the project:
- Construct the socio-environmental model space that includes identification of the complementarities and intersections in the model space, which will be a basis for model integration.
- Development and application of a methodology for integration of overlapping models.
- Development and application of tools for linking component models.
- Integrate computational and conceptual mental/verbal/graphic stakeholder models.
- Model uncertainty and sensitivity.

In general, the COMPLEX project has served as an input in most of the parts of this research. The models, stakeholders, and project objectives served as resources in conducting this research. Considering the case study, it was aimed to develop a flexible integrated modeling framework that can be used to represent the climate-energy-economic system at different levels of spatial, temporal, and structural detail and resolution.

For our pilot application we started with a stakeholder meeting that aimed to outline the integrated modeling system that could be used to investigate the effect of alternative climate change scenarios on different regions and sectors of the economy. Mostly modelers were present as stakeholders. In this meeting project members described the capabilities, spatial and temporal scales, and the tools used to develop their models. The model description effort was also reinforced: a model documentation template was proposed, in which modelers could provide the specifics about their models. After a number of follow-up meetings three models were selected for integration. We conceptualized the integrated system and we formulated the interaction and

data exchange flows between the models. At the same time we were working on the architecture of the framework and the software implementation. Finally we were able to integrate the chosen models and simulate some climate change scenarios.

1.5 Dissertation outline

The dissertation consists of the following six chapters:
Chapter 1 is a brief description of the research background, motivation, objective of the research, and context of the case study.

Chapter 2 presents a literature review performed to improve our understanding of the model integration process, and to design better strategies of integrated modeling. Integration of models is an iterative and incremental process that requires higher level of software engineering and semantic processing than conventional modeling. Five different phases of integration are identified to characterize the model integration process. Then integration strategies, features, standards, and practices that are used by different groups to improve the different phases of model integration are highlighted. Besides, issues of discoverability, accessibility, and ease of use of integrated systems are discussed. The chapter concludes by providing a list of recommendations that can be applied in developing integrated modeling systems.

Chapter 3 describes the design and prototype of web-based model integration framework developed using distributed computing and service-oriented software development approaches. We described a technique that transforms independently developed models into web-enabled interoperable plug-and-play components. Besides, we also discussed methods to reconcile implicit semantic relations in integration of various models. Semantic matching algorithms for text-based input-output data and model metadata are presented. Automatic unit conversion in integration of models using an openly available ontology is discussed. Finally, runtime integration of models, i.e. an integration method in which users can access and integrate models 'on the fly' using a graphical user interface is introduced.

Chapter 4 explores temporal and functional synchronization and sensitivity in integration of models. The time stepping in different component models can be very different and the use of inappropriate temporal resolution can result either in major run-time redundancy or in loss of model accuracy. Similarly, when linked models represent one and the same process using different mathematical expression and using different numeric integration methods, it may affect the accuracy of the integration output. The chapter

describes how the overall system performance can be sensitive to various parameters of the integration process.

Chapter 5 describes the case study where we integrated heterogeneous climate, economy and energy market models that operate at various scales, from global to household level. Pre-integration assessment processes in identifying models and conceptualizing the integrated system is presented in the context of several existing sophisticated models. Mechanisms used to transform the participating models into web services are discussed. Simulation results of some selected climate change scenarios are presented. The chapter concludes by pointing out integration strategies that could be replicated in other similar projects.

Chapter 6 summarizes the results discussed in Chapters 2-5 and suggests key directions for further research.

Chapter 2

An overview of the model integration process: from pre-integration assessment to testing*

* This chapter is based on:
Belete, G.F., Voinov, A., Laniak, G.F, 2017. An overview of the model integration process: from pre-integration assessment to testing. Environmental Modelling and Software 87, 49-63.

Abstract
Integration of models requires linking models which can be developed using different tools, methodologies, and assumptions. We performed a literature review with the aim of improving our understanding of model integration process, and also presenting better strategies for building integrated modeling systems. We identified five different phases to characterize integration process: pre-integration assessment, preparation of models for integration, orchestration of models during simulation, data interoperability, and testing. Commonly, there is little reuse of existing frameworks beyond the development teams and not much sharing of science components across frameworks. We believe this must change to enable researchers and assessors to form complex workflows that leverage the current environmental science available. In this paper, we characterize the model integration process and compare integration practices of different groups. We highlight key strategies, features, standards, and practices that can be employed by developers to increase reuse and interoperability of science software components and systems.

2.1 Introduction

A model is a simplified abstraction of reality (Schmolke et al., 2010; Voinov, 2008). To answer questions related to environmental problems, scientists and policy-makers, may need to address complex issues at regional, continental, and even global scales (Harris, 2002; Verburg et al., 2008). Developing all-inclusive models is difficult, however, and we must integrate individual models that represent specific domains (Gregersen et al., 2007). Integration of models is becoming more important due to an increased desire to understand, investigate, and mitigate human impact on the environment (Laniak et al., 2013; Voinov and Cerco, 2010). The challenge is making standalone models discoverable, reusable, and interoperable (Goodall et al., 2011). Here, discoverability means the availability of meta information of a resource so it can be effectively searched for, located and understood to ensure proper use (Erl, T., 2008b). Reusability is how existing software can, in whole or in part, be used to develop new software (Frakes and Kang, 2005), as well as the degree to which an asset can be used in more than one system or in building other assets (ISO/IEC, 2011). Reuse also implies a decision process that includes evaluating software for its ability to serve a new purpose. And, finally, interoperability is the ability of a system or a product to work with other systems or products without special effort by the user (Chang and Lee, 2004).

In this paper a component may represent an elemental physical/ chemical/biological/social process (e.g., infiltration or advection) or a logical integration of elemental processes (e.g., a watershed model). A system may

be a model (an integrated set of elemental processes) or modeling system. Note that a model can refer to a component or a system. Finally, when we refer to an integrated modeling system we are including both the collection of science components and the framework software that facilitates their orchestration. These definitions apply throughout the paper.

Data exchange and data manipulation are fundamental to integrated modeling systems (Argent, 2004; Leimbach and Jaeger, 2005), and are often constrained by technical and conceptual challenges. Technically, individual models are designed to serve as stand-alone components that serve unique purposes and goals. Conceptual challenges include resolving the different ways modelers and science domains represent data and knowledge. Interoperability of models can also be provided at different levels. At the technical level, models should be able to 'talk to each other' which requires automating data exchange, making models jointly executable, and ensuring repeatability and reproducibility of model chain configuration and processing (Knapen et al., 2013). At the semantic level, models should 'understand each other' by identifying and, if possible, bridging semantic differences in an automated manner. After semantic mediation, the dataset should be syntactically interoperable, i.e., data produced by one model are converted and formatted to serve as input for a receiving model.

We performed a literature review to improve our understanding of the model integration process and also to collect and present better strategies for building integrated modeling systems. At first, our review was limited to integrated climate change mitigation analysis systems. Due to the commonality of integration techniques with other environmental domains, however, we broadened the scope to include integration of models in general. Our search used key phrases of 'integration of models', 'integrated assessment study' and 'integrated modeling frameworks,' and we considered 38 articles (see Table 2.1), of which 18 focused on specific integration modeling systems (aka frameworks).

To provide structure and context for the paper, we look to model development process which includes requirements analysis, design, implementation, and testing phases (David et al., 2013; Moore and Tindall, 2005; Whelan et al., 2014), with these steps conducted in an iterative manner (Grimm et al., 2014; Jakeman et al., 2006). Model integration process follows a similar sequence and is also iterative (Holzworth and Huth, 2011; Holzworth et al., 2014), with a small portion of integration realized first, and more complex issues and functionalities incorporated through iterations. Keeping all this in mind, we divided model integration process into phases of pre-integration assessment, preparation of models for integration,

orchestration of participating models during simulation, data interoperability, and testing (Fig. 2.1).

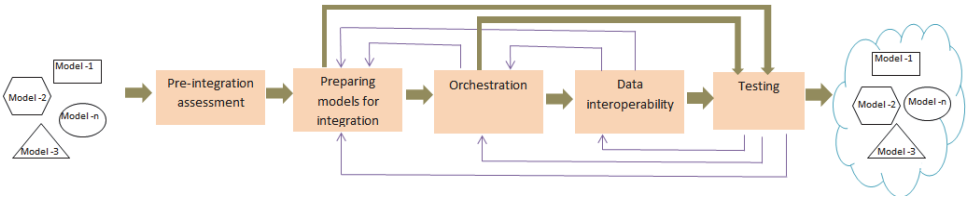


Figure 2.1 : Iterative relationship among model integration phases.

This process forms the basis for our review. As we conducted the review, it became clear that three topics deserve special attention due to their significant role: interface standards for interoperability (within the preparation of models for integration phase); performance in integrating models (within the orchestration phase); and discovery, accessibility, and ease of use which are process-wide considerations. A summary of this structure, along with brief descriptions of categories of methods employed, examples of existing modeling systems, and references are presented in Table 2.1.

We note that:

- Most papers emphasize only certain phases of model integration.
- Comparing all selected frameworks with all aspects of the integration process is challenging due to limitations of available information. We believe, however, that, collectively, the full range of methods is represented.
- The set of model integration frameworks in Table 2.1 represents a robust, albeit incomplete, representation of existing frameworks.

Table 2.1: Elements or aspects of model integration practices.

Elements or aspects of model integration	**Different approaches to implement elements**	**Description and example frameworks**	**References**
Pre-integration assessment	-	How selection of models and system conceptualization is done. Examples in: FRAMES, OpenMI, SEAMLESS	Butterfield et al.(2008); Janssen et al.(2009); Janssen et al.(2011); van Ittersum et al.(2008); Whelan et al. (2014)
Preparing models for integration	Component based	Models are presented as components for interoperability. Examples in: CSDMS, ESMF, FRAMES, OpenMI, SEAMLESS	DeLuca et al. (2012); Janssen et al.(2011); Peckham et al. (2013); van Ittersum et al.(2008); Whelan et al. (2014);
	Web service based	Models are presented as web services for interoperability. Examples in: AWARE, eHabitat, GEO Model Web	Dubois et al. (2013); Goodall et al. (2011) Granell et al. (2010); Geller and Melton (2008); Geller and Turner (2007); Nativi et al. (2013); Roman et al. (2009);
	Hybrid	Some models are presented as components and others as web services.	Castronova et al. (2013); Goodall et al. (2013);
	Tailor-made	Custom-made techniques for interoperability. Examples in: CMP, TIME	Moore et al. (2007); Rahman et al. (2003);
Interface standards for interoperability	-	Model interface standards designed by the modeling community. Examples in: CSDMS, ESMF, OpenMI, OGC WPS.	da Silva et al. (2003); DeLuca et al. (2012); Goodall et al. (2011); Hill et al. (2004); Moore and Tindall (2005); Peckham (2014); Schut and Whiteside (2007);
Orchestration of interaction of models during simulation	Component based	Orchestration of models is done using component-based development. Examples in: CSDMS, FRAMES, OpenMI, SEAMLESS	Gregersen et al. (2007); Janssen et al. (2011); van Ittersum et al. (2008); Peckham et al. (2013); Whelan et al. (2014);
	SOA based	SOA principles are used for orchestrating models. Examples in: AWARE, eHabitat, GEO Model Web.	Dubois et al. (2013); Butterfield et al. (2008); Goodall et al. (2011); Granell et al. (2010); Nativi et al. (2013);
	Hybrid	Both component-based and SOA-based approaches are used.	Castronova et al. (2013); Goodall et al. (2013);
	Tailor-made	An ad hoc collection of techniques for orchestration. Examples in: APSIM, CMP, OMS,	David et al. (2013); Holst (2013); Moore et al. (2007); Keating et al. (2003);
Performance in integrating models	-	How performance optimization is treated by integration frameworks. Examples in: ESMF, OMS	David et al. (2013); Hill et al. (2004); Knapen et al. (2013).

Elements or aspects of model integration	Different approaches to implement elements	Description and example frameworks	References
Data interoperability	Hard-coding	Hard-coding is used for data interoperability. Examples in: CMP, eHabitat	Dubois et al. (2013); Moore et al. (2007);
	Framework specific annotations	Application-specific annotations are used for data interoperability. Examples in: OMS, TIME	David et al. (2013); Rahman et al. (2003);
	Controlled vocabulary and ontology	Reusable vocabulary and ontology is used for data interoperability. Examples in: FRAMES, SEAMLESS	Argent (2004); Athanasiadis and Janssen (2008); Geller and Turner (2007); Nativi et al. (2013); Rizzoli et al. (2008); van Ittersum et al. (2008); Whelan et al. (2014);
Testing	Unit test, integration test, automated testing, test bench	How frameworks treat testing. Examples in: APSIM, FRAMES, ModCom, OpenMI, SIAT, TIME	Bertolino (2007); Bruggeman and Bolding (2014); Hillyer et al. (2003); Holzworth et al. (2014); Holzworth et al. (2015); Luo (2001); Moore and Tindall (2005); Rahman et al. (2003); Verweij et al. (2010); Whelan et al. (2014);
Discovery, accessibility, and ease of use	-	Efforts to improve discovery, accessibility, and ease of use of models. Examples in: BioMA, eHabitat, InsightMaker, iPlant, ModCom, SEEK, Systo,	Brooking and Hunter (2013); da Silva et al. (2003); DeLuca et al. (2012); Goecks et al.(2010); Goff et al. (2011); Hillyer et al. (2003); Wolstencroft et al.(2015);

Findings related to phases of integration: pre-integration assessment, preparation of models for integration, model orchestration, data interoperability, and testing are described in Sections 2.2 through 2.5. To avoid repetition of content, we treat preparation of models for integration and orchestration of participating models during simulation together in Section 2.3. Discoverability, accessibility, and ease of use of integrated systems are addressed in Section 2.6. Section 2.7 presents discussion as well as conclusions, and finally recommendations for increasing efficiency and effectiveness of the model integration process are presented in Section 2.8.

2.2 Pre-integration assessment

Pre-integration assessment combines science-based requirements of an environmental assessment with computer technologies to create an initial integrated software system workflow design. The assessment may be related to research, environmental analysis, or decision/policy support. Pre-integration represents a key aspect of model integration since it establishes the roles of and relationship between scientist (performing an environmental assessment, for example) and software engineer (designing and building the computer-based modeling system). These complementary roles have often been performed by the same person, but as science problems and required modeling systems have grown more complex, these roles necessarily

required a relationship between experts in the domain science and software engineering.

On the science side, pre-integration assessment includes a problem statement; articulation of the purpose and goals of the science-based modeling effort; conceptualization of the relevant system (including major components and interactions); characterization of scenarios and use cases; analysis of alternatives; design of a science-based approach; and a statement of any project resource constraints and solution criteria. Collectively, this information represents the requirements that must be satisfied by either an existing or new software system.

Ramamoorthy et al. (1992) point out that the main cost of integration is the effort required to detect and resolve inconsistencies in the coupled system, a cost that can be reduced if inconsistencies are detected and resolved prior to integration -- that is, during the process of transforming the science design to a software design. To aid transformation, the software engineer may utilize a set of tools recommended by the System-of-Systems Engineering (SoSE) approach (Butterfield et al., 2008), including: context views that define relationships, dependencies, and interactions between the system and its environment; use cases that list interactions between the system and the user; activity diagrams which show the flow of activities while using the system; and diagrams that provide a high-level description of system behavior.

In reviewing existing frameworks, we found only a few scientific papers that discuss how pre-integration assessment was performed. In SEAMLESS (Janssen et al., 2011; van Ittersum et al., 2008), pre-integration was performed through discussions that involved scientists, assessors, decision support specialists. The discussions focused first on defining scenarios and indicators (which would be tested by the integrated system), and then on analyzing system requirements. An assessment project ontology (Janssen et al., 2009) was developed to create a common understanding for people involved in the project. The discussions produced the overall architecture of the integrated system, specifications of components, and data to be exchanged between them.

For FRAMES, Whelan et al. (2014) intended to accommodate multiple modeling domains and assessment scenarios (workflows) by designing the framework to be independent of any specific set of models. They suggested that developing a framework capable of constructing and orchestrating integrated modeling systems required detailed analysis with respect to 19 aspects of design and implementation that they have identified. In summary, these requirements target metadata definition, standardized vocabularies,

component connectivity, data-transfer compatibility, system architecture and functionality, and model accessibility.

The OpenMI project provides a formal standard for data exchange among models, including specification of an application program interface that describes how models should request, and respond to requests for, data. Various frameworks, including the OpenMI software development kit, implement this standard to achieve model interoperability. OpenMI suggests that model-linking can begin only after identifying data that can potentially be exchanged, defining how the data will be exchanged, and schematizing deployment and running of the linked computation. Developers who want to use the OpenMI framework should ensure that components use the same platform (for example, do not mix Java and .Net components); that the same OpenMI version is implemented; and that input data not provided by other components will be provided from external data sources. This shows that pre-integration assessment must consider different levels of details in meeting integration goals.

The connection between integration frameworks and science domains cannot and should not be ignored, although there are various levels of 'affinity'. For example, SEAMLESS was entirely driven by agricultural science. FRAMES or OpenMI were intended to be domain-independent, however, they were initiated by specific science questions and domains. FRAMES is rooted in chemical fate and exposure modeling, but OpenMI is largely driven by hydrological applications. Similarly, ESMF had its origins in Earth systems modeling, while CSDMS was instigated by surface dynamics modeling. Such affinity to particular science domains is revealed by model components in the respective model repositories, in the types of applications where the frameworks have been tested, and also in the equations and methods that the frameworks typically handle.

From the literature review, we identified the following questions for facilitating pre-integration assessment:

(1) Why are we integrating? What are the objectives?
(2) What system is to be simulated? What scenarios are to be simulated using the integrated system?
(3) What indicator variables are to be tested using the integrated system? What are the options for calculation of the indicator values?
(4) Are the required modeling components from the same science domain? Are the semantics compatible? Are there frameworks that cover the relevant science domain, or you will need to fit into a 'foreign' framework or develop both the new framework and the models within this framework?

(5) How much data and information will the models exchange, such as one variable vs. hundreds of variables, how often, over what spatial grid and resolution? What processes require feedback loops?
(6) Is it important (or necessary) to keep participating components independent of one another, or will linking be established by creating a permanent dependency of one component upon another, effectively merging multiple components into a single one?
(7) What are the implications of programming languages, operating systems, and computer platforms used? Will language interoperability tools be required? Will individual components remain on their respective platforms (e.g., development environment or operating system) or they will they be migrated to a common platform in the integrated system?
(8) What programming skills are required and are they available? How much new software will be required to link models?
(9) How will system input data (i.e., data about the physical/ chemical/biological/social etc. system being modeled) be organized and distributed among components?
(10) Are data unit conversion functions needed?
(11) Is dataset translation, aggregation/disaggregation required? If so, are the tools available or must they be created?
(12) Are there specific performance requirements to satisfy (e.g., parallel computation)?

Consideration of the above-mentioned conceptual and technical questions is the basis for an initial design that will enable to evaluate feasibility of the integration and resource requirements. The result of the pre-integration assessment will be either to proceed with integration, not consider integration, or obtain additional information to make a better decision. If the decision is to continue with integration, the next questions are "How should the models be prepared for integration?", and "How should model orchestration be conducted?" These topics are addressed in the next section.

2.3 Technical integration of models: preparing models for integration and orchestration of participating models during simulation

Technical integration requires specific software strategies that prepare modeling components for linked execution within a framework designed to execute the integrated modeling system. Because the literature does not provide sufficient detail, we will not attempt a technical framework-to-framework comparison. We introduce concepts of model preparation and system orchestration in Sections 2.3.1 and 2.3.2; discuss model preparation strategies and requirements employed by existing frameworks in Section

2.3.3; and discuss two key issues related to model integration and orchestration, interface standards for interoperability and framework performance optimization, in Sections 2.3.4 and 2.3.5.

2.3.1 Preparing models for integration

From a software engineering perspective, preparing a component for integration depends on its native software environment (language and OS), manner in which inputs and outputs are organized (e.g., using files, databases, etc.), and integration approach used by the target system (framework). Consider a practical example of a model that produces climate change scenarios up to year 2100. The simulation start year is constant, but the end year is a user input. The model's output, as originally designed, is accessible only after the final time step. If we want to link this model with another so that data is exchanged at the end of each time step, we must make certain modifications. Modifications include preparing a function that returns the values of the required variables at the end of each time step, and a function that receives values of selected variables from the model to be linked. In achieving this linkage, we need to consider that the two models are developed using different languages, are deployed on different platforms, treat space and time differently, etc. The process of preparing models for integration can thus go through some combination of steps, as summarized in Fig 2.2. And if we want to use the model independently of the development environment, or need to protect intellectual property rights of developers, we may have to deliver a model in binary format as DLL or EXE file rather than source code (Moore et al., 2007).

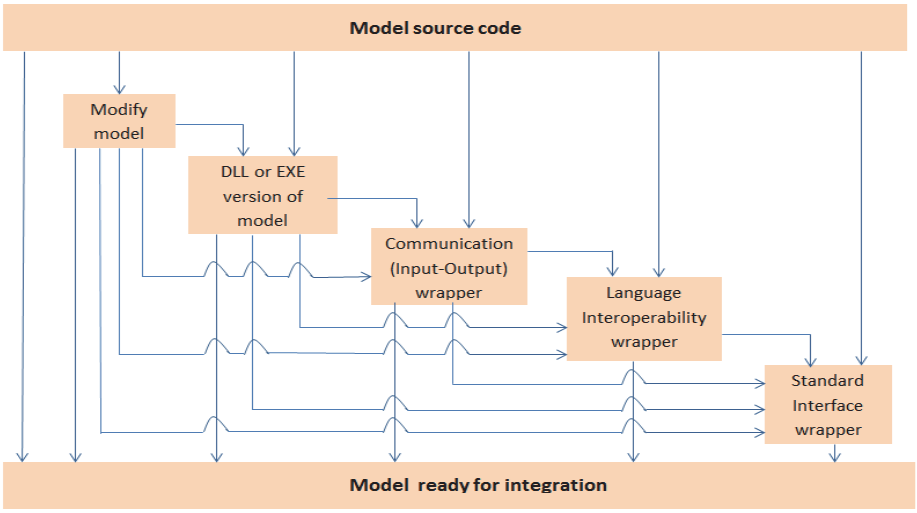


Figure 2.2: Alternative paths in preparing a model for integration.

Model modifications will sometimes be needed to enable automated input-output data exchange that conforms to a specific framework design. In such situations, small existing (legacy) models can be reprogrammed, or we may directly modify the model interface to conform to the framework data exchange protocol (Figure 2.3). For larger and more complex models, however, rewriting is time-consuming and error-prone (Peckham et al., 2013). To avoid rewriting, it is common to develop a communication wrapper (Van Ittersum et al., 2008) which is a thin layer of code on top of the native component that manages data exchanges between the native component and other components or the system. Wrappers could be developed to make a model language-interoperable; make a model accessible over a network; implement agreed-upon names for functions and variables; provide a custom interface with required input-output attributes; or address a combination of these objectives. Depending on the integration framework, a model wrapper can be developed following industry standards or "local" standards; some of these are discussed in Section 2.3.4. Generally, the work required to develop wrappers depends on how much we want to adapt the existing model interface.

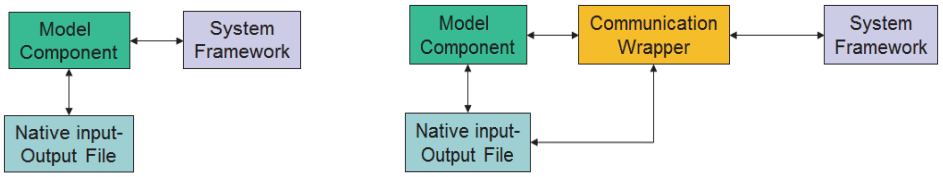


Figure 2.3: Common strategies to implement framework data exchange protocol on existing models are to modify the model interface directly to conform to the framework data exchange protocol or develop a wrapper as a separate component.

2.3.2 Orchestration of participating models during simulation

A complementary part of technical integration is orchestrating models during simulation which requires enabling and managing interactions between components (Madni and Sievers, 2014). The orchestration task includes: identifying components to be included; establishing linkages and overall workflow to be implemented; and managing execution of the workflow (initializing components, synchronizing execution, facilitating data exchange, error management, and system file management). Furthermore, the operational nature of the data exchange within an integrated system may be sequential (feed forward) or iterative (feedback). In feed forward systems, one component completes its execution and produces data for subsequent consumption. In feedback systems, components require the exchange of intermediate data values at certain time steps.

2.3.3 Commonly used approaches for technical integration of models

The actual methods employed for technical interoperability of models vary among development groups. From our literature review, we categorized them as component-based, web service-based, a hybrid of component and web service-based, and tailor-made (custom-made) techniques. These methods can be applied to both legacy and new science components. A brief comparison of the methods, which are further described in the following sections, is presented in Table 2.2.

Table 2.2: Brief comparison of technical integration methods commonly used by integration frameworks.

	Component-based	SOA-based	Hybrid of Component-based and SOA	Custom-made
Primary characteristics	Reusability of components and system level functionalities.	Loosely coupled systems; distributed services.	Models as components and as web services.	An ad hoc collection of methods/ techniques is used.
Advantages	Plug & play; separation of concern; minimizes context switching time of components.	Plug & play; location independence; platform independence; scalable.	Enables linkage to different frameworks; manage heterogeneity; reuse existing frameworks.	Developers can use ad hoc techniques for orchestration.
Disadvantages	Developing reusable orchestration engine is challenging.	May require heavy data exchange, and high availability.	Requires parsing data between components and web services.	May not support new version of models; lack of extensibility.
Examples	CSDMS, OpenMI, FRAMES, SEAMLESS	AWARE, eHabitat	Castronova et al. (2013), Goodall et al. (2013)	APSIM, CMP

Component-based approach for technical integration of models

The component-based approach is the design and construction of computer-based systems using reusable software components which are independent executable entities accessible only through its interface (Debayan, 2011). One main difference of component-based development from non-component-based is the emphasis on reusability of components (Crnkovic et al., 2006). This approach recognizes different levels of reusability: code-level, module-level, library-level, design pattern-level, and framework-level (Rekaby and Osama, 2012).

The component-based development lifecycle involves two main phases, component production and component integration or utilization (Rekaby and

Osama, 2012). In the component-based approach, components have standardized calling interfaces (DeLuca et al., 2012) or 'sufficient commonality' (Debayan, 2011) which enables developers to formulate predictable architectural patterns that manage interactions between them. As a result, the workflow-managing functionality can itself be developed as a reusable framework-level component.

The component-based approach is one of the most commonly used techniques to prepare models for integration and manage simulations. For example, the Earth System Modeling Framework (ESMF) uses a component-based approach to integrate models (DeLuca et al., 2012); ESMF is based on the principle that complicated applications are broken into smaller components with standard calling interfaces. A model component that implements the ESMF standard interface can communicate with the ESMF shell and inter-operate with other models. Similarly, the CSDMS (Peckham et al., 2013) requires models to be presented as components to join the integration framework. Component-based programming is chosen so components can be added or replaced at runtime, and to ensure that components written with different programming languages and hosted on remote locations can join the framework. The CSDMS framework requires models to implement the two-level component-interface standard provided. The CSDMS uses functionalities of the common component architecture (CCA) framework[2] to coordinate communication between components developed using different frameworks (Peckham et al., 2013). FRAMES (Whelan et al, 2014), a feed forward modeling framework, employs the component-based approach and incorporates data dictionaries for data exchange. Wrappers are written for each component to read and write data to the dictionaries. The framework then manages transfer of data between components during runtime through an inter-component communication API. The API consists of routines that read model inputs, write model outputs, handle language interoperability, manage unit conversions, and provide error handling.

Existing component-based frameworks mostly employ a local data exchange standard. The exception involves the OpenMI standard that provides plug-and-play model access by implementing a component-based software development paradigm. The OpenMI strategy is to access model input-output directly at run-time and not use files for data exchange. OpenMI-compliant components can exchange data without any supervising authority if the components have information about each other (Gregersen et al., 2007). If, however, the communication involves iterations -- that is, feedbacks and looping over components -- a separate controller component is required; the

[2] CCA - http://www.cca-forum.org/overview/

controller contains an iteration algorithm and a link to individual components involved in the iteration. The OpenMI standard has been implemented in some framework designs such as SEAMLESS-IF (Janssen et al., 2011; van Ittersum et al., 2008), SIAT (Verweij et al., 2010), NITROEUROPE (Sutton et al., 2009), EVOLTREE[3], and FluidEarth[4].

The component-based approach has a number of advantages for the modeling domain, including:

- Components can be added or deleted from a simulation at runtime (Armstrong et al., 1999; Peckham et al., 2013);
- It is suitable for rapid execution of workflows because when an application (in this case, the workflow manager) calls a component, the component can run within the process space occupied by the calling application, and it can share memory and processor time with the calling application. This makes the communication between the calling application and the component very fast;
- It enables models to be independent and, thus, reusable entities;
- It enables creation of both tightly- and loosely-coupled systems; and
- It facilitates sharing of models and possibly collaboration among modelers or workgroups who share components.

Although the component-based approach has many advantages, it also has many challenges:

(1) Communication between components developed using different programming languages requires language interoperability tool (Peckham et al., 2013). Some systems avoid this by requiring all components to be written in the same language, but some use a language interoperability tool-- for example, the CSDMS framework uses Babel[5] a language interoperability tool that manages communication between components written in C, C++, Java, Python, or FORTRAN;

(2) Communication between components hosted on different computers is commonly done via tight-coupling. For example, both CSDMS and FRAMES use Remote Method Invocation (RMI) for remote communications. RMI enables an object hosted on one computer to invoke an object hosted on another. RMI has disadvantages, however, since it operates on the object level (i.e., components are treated as objects rather than independent applications), and it creates tight- coupling between components. RMI is also slow and can be used only with Java, making it programming language-dependent (Tanenbaum and Van Steen, 2007);

(3) Due to the many unique implementations of data exchange protocols, the component-based approach has resulted in interoperability that is typically

[3] EVOLTREE - http://www.evoltree.eu/
[4] FluidEarth - http://fluidearth.net/default.aspx
[5] https://computation.llnl.gov/casc/components/index.html#page=home

constrained to a specific framework, versus a common standard (as proposed by OpenMI) that would facilitate global or universal reuse and interoperability.

Web services and Service-oriented approach for technical integration of models

Web services are web application components that can be accessed via the World Wide Web. Unlike the component-based approach, web services can interact independently of the underlying platform. They eliminate interoperability complications related to programming language, operating system, and computer platform (Castronova et al., 2013; Dubois et al., 2013; Goodall et al., 2011). Independently-built heterogeneous services can be linked to build integrated modeling systems using a Service-Oriented Architecture (SOA) design approach. SOA is based on design principles: standardized service contract or description; service loose-coupling; service abstraction (hiding unnecessary information); service reusability; service autonomy (keeping the service as an independent unit); service statelessness (minimizing the time in which the service will stay stateful); service discoverability (use meta information that accurately describe the service); and service composability (design services that will be composed with other services) (Erl, 2008b).

A main factor that differentiates the service-oriented approach from the component-based approach is that service-orientation implies distributed components will interoperate over a network (Huhns and Singh, 2005; Goodall et al., 2011). SOA enables building distributed applications by composing or putting together services while keeping them autonomous, regardless of their heterogeneity (Pessoa et al., 2008). In SOA, unlike a component-based approach, a langauge interoperability tool is not required for communication of hetrogeneous services. Services may also run on different operating systems, and can encapsulate legacy systems. Configuration of services into integrated modeling systems can occur dynamically (Erl, 2008a; Papazoglou, 2012) and is scalable (Nativi et al., 2013).

Model developers use web services for various reasons, and the service-based approach is becoming increasingly popular. For example, the Group on Earth Observation (GEO) Model Web initiative (Nativi et al., 2013) suggests that presenting models as a service will increase interoperability and facilitate interaction of models, databases, and websites. They use a SOA-based approach to orchestrate distributed components, introduce a broker component to facilitate discovery of services, intermediate (i.e., inter-component) services to adapt output content and format of one model to an appropriate format before it is passed to the next, and handle asynchronous

communication between remote servers. Similarly, the Model Web for Ecological models (Geller and Melton, 2008; Geller and Turner, 2007) suggests that an open-ended network of interoperable models and databases that is maintained, operated, and served independently can be realized with web services. The Model as Service approach (Roman et al., 2009), which is an extension of the Model Web initiative, notes that web services can migrate a stand-alone model into a service on the Web.

The AWARE[6] integration framework (Granell et al., 2010) uses the open geospatial services standard to make environmental models interoperable. They selected SOA because services can be published, discovered, aggregated, invoked, and reused independent of the underlying technology used to build them; service standards define how to interact with the available services in a uniform manner; the interoperability of web service provides the opportunity to present legacy models as services; and SOA enables creation of value-added functionality by linking existing services. Web services can be delivered to end users users for visual service-chaining by presenting them via Cloud computing, and Grid computing technologies, and service-oriented infrastructures. Goodall et al. (2011) recommends a service-oriented computing approach to orchestrate models in simulation since SOA views integrated modeling systems as an interconnected collection of distributed components. Similarly, eHabitat (Dubois et al., 2013) uses SOA with brokers to handle service composition. A discovery broker, access broker, and semantic discovery broker mediate communication between services and hide the heterogeneity of underlying services.

The web service approach is used mainly to enable cross-language interoperability and link components that reside on different operating systems. Its main disadvantages are that request and replay messages are based on XML, which is larger in size than binary protocol; and performance on the Internet varies from time to time and place to place. It may require high availability.

Hybrid of component-based and Web service-based approach for technical integration

Constructing an integrated modeling system using a specific software architecture or framework may not be optimal since a mix approaches may be more useful. Some modelers use a hybrid of component-based and service-oriented approaches to manage technical integration of models. Consider the following two cases where both used two-level wrapping to access a model hosted on a remote machine, as shown in Fig 2.4. Castronova et al. (2013) used an OGC WPS wrapper and an OpenMI component wrapper

[6] http://www.aware-eu.info/

to prepare the models for integration. Implementing the WPS on a model transforms it into a web service which can be accessed remotely via Internet. The linking of models is handled through the OpenMI-based HydroModeler environment, however, which requires wrapping models with the OpenMI interface standard. They developed an OpenMI compliant-wrapper that invokes WPS-based services.

Likewise, Goodall et al. (2013) used the hybrid approach to enable coupling across modeling frameworks. They linked two existing modeling frameworks -- ESMF and OpenMI -- using a service-oriented architectural approach. A climate model developed within ESMF and hosted on a high performance computing cluster was made available as a web service, and an OpenMI-based client that accesses the service was also built. The Soil and Water Assessment Tool (SWAT), a hydrological model, was hosted on a Windows-based personal computer and made OpenMI compliant. Communication between the two OpenMI-compliant components was organized using the OpenMI configuration editor which handled the transfer and translation of the exchanged items. From these two examples we can see that the service-oriented approach overcame heterogeneity due to computer architecture and programming language differences and the component-based framework was utilized to re-use the existing OpenMI integration tool.

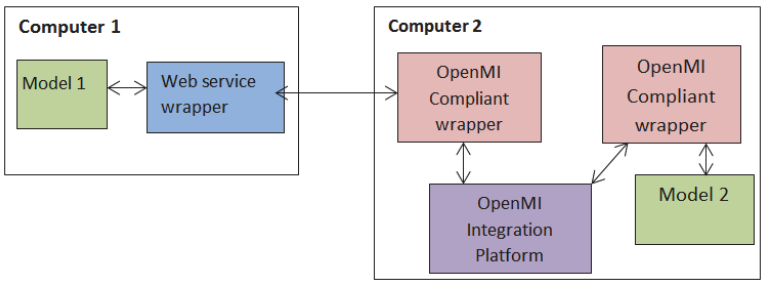


Figure 2.4: Two-level-wrapping for cross-platform interoperability: Web service and OpenMI-compliant component wrapper.

Tailor-made (custom-made) techniques for technical integration of models

Some integration frameworks use interoperability techniques that are unique and neither component-based nor web service-based. We use the term 'tailor-made' or 'custom-made' to group those frameworks into one category. In the custom-made approach, developers use an ad hoc collection of methods/techniques for technical interoperability. They use framework-specific 'principles' and this could hinder the continuity and extensibility of such frameworks. Each is quite different and has its own advantages and disadvantages.

In the Common Modeling Protocol (CMP), models are presented, for technical interoperability, as an executable file or dynamic link library (Moore et al., 2007). No further specification is required on the structure of the model interface. Simulation specification is a hierarchical arrangement of components (i.e., a system is a tree of subsystems) using the Simulation Description Markup Language (SDML). It uses message-passing system to coordinate communication between components. To run a simulation, the user must define the configuration of components and initial values of state variables in an XML document that conforms to the SDML schema. Similarly, the Agricultural Production System Simulator (APSIM) (Keating et al., 2003) modules are self-contained and a central message-passing system is used for inter-module communication. Every simulation in APSIM must be represented as an XML document. Users employ the GUI to specify a simulation configuration by selecting preferences from a set of toolboxes. On the other hand, the TIME modeling framework (Rahman et al., 2003) requires models to be developed as a class using any.NET language such as C#, Visual J#, Visual Basic .NET, or Fortran 95.NET because cross-language interoperability[7] between models is handled by the Common Language Runtime[8] environment of the .NET framework.

Domain Specific Languages (DSL) (Fowler, 2010), programming languages developed to meet specific needs of a domain, also manage orchestration of models in simulation (Holst and Belete, 2015). OMS3 provides a domain-specific language named 'Simulation DSL' to configure simulations (David et al., 2013). The DSL is developed independently of the framework and functions as a layer on top of participating components. Similarly, the Universal Simulator for Ecological Models – UniSim – (Holst, 2013) uses DSL to define orchestration of components in a simulation.

In the custom-made approach, we observe that frameworks do not impose explicit model interface standards for interoperability, which could imply they require custom code to access specific models. Incorporating new versions of models could also require modification of the custom code. Custom-made approaches thus have the disadvantages of lacking flexibility, maintainability, and extensibility.

2.3.4 Interface standards and interoperability

Wrapping of models is commonly done for technical interoperability. If the wrapper interface is developed using a set of standard names for functions and variables, however, semantic interpretation of their attributes will be uniform and software can be included to test the correctness of data

[7] https://msdn.microsoft.com/en-us/library/vstudio/a2c7tshk(v=vs.100).aspx
[8] https://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.110).aspx

exchanges. Interface standards can therefore serve both technical and semantic interoperability. Currently, a standard does not imply universal awareness, acceptance and use, but rather that a framework designer has either specified or adopted a standard for use; thus, different interface standards are available. Commonly interface standards have mandatory and optional parts. They can be applied in developing new models or in re-factoring of existing models. A brief comparison of existing interface standards is provided in Table 2.3.

Table 2.3: Some interface standards used by the modeling community.

Name of interface standards	Description	Some of its features	Reference
OpenMI standard	A collection of programming interfaces for components.	It consists of a list of function names or method signatures called *initialize*, *update*, *inputItems*, *status*, etc. that enable the model being wrapped to request data from other models and respond to requests for data.	Moore and Tindall (2005)
CSDMS standard	Developed to simplify conversion of an existing model to a reusable, plug-and-play model component.	It has two levels of specification: (1) BMI[9] developed to provide model metadata to the next level and (2) CMI[10] communicates with BMI functions as well as with Service Components and the CSDMS Framework	https://csdms.colorado.edu
OGC WPS	Specifications on how inputs and outputs of geospatial services are handled.	It has three mandatory operations that should be implemented by all services: *GetCapabilities*, *DescribeProcess*, and *Execute.*	Schut and Whiteside (2007)
ESMF standard	Developed to have standard interface for model components.	To make a model ESMF-compliant, it should include methods named *initialize*, *run*, and *finalize.*	da Silva et al. (2003); DeLuca et al. (2012); Hill et al.(2004)
Web service standard for the water domain	Developed to present models from water resources domain as web services.	It combines semantic features like where, what, and when of the exchanged data from the OpenMI with web service features of the OGC WPS standards.	Goodall et al (2011)

2.3.5 Performance optimization and related issues in orchestrating simulations

Performance of computer simulation is an important concern for all modeling and integrated modeling, in particular. In this context, performance

[9] BMI - http://csdms.colorado.edu/wiki/BMI_Description
[10] CMI - http://csdms.colorado.edu/wiki/CMI_Description

optimization is the process of tuning a software system to execute rapidly. Performance can be affected by the design of the software itself, the operating system, middleware, hardware or communication networks (Woodside et al., 2007). Here we discuss two performance optimization concepts applied by developers of integration frameworks: high performance computing (HPC) and parallel processing (multi-threading).

HPC is a cluster computing environment with a number of nodes containing both processing power and memory (Gavrilovska et al., 2007). The benefit of HPC is that data which are too large to process by one server will be distributed over a number of nodes. The HPC platform manages the distribution and processing. HPC has been used by modeling frameworks to simulate complex workflows such as by ESMF for climate simulations (Hill et al., 2004). Linking the integrated system with the HPC platform is an added value.

In designing integration, orchestration of models can be treated as a serial or parallel process. In reality, many natural processes represented by integrated models can happen in parallel, but such parallel processes are commonly represented using sequential execution of simulation models. In fact, almost all modern computers currently have multi-core CPUs, and the parallel execution approach provides effective utilization of available processing resources (David, et al., 2013). Parallel execution of processes can be achieved using multithreading techniques (Tanenbaum and Van Steen, 2007). This also improves performance in non-HPC environments because threads in a process share the same memory area which reduces the time spent for process context-swapping. For example, OMS3 uses a multithreading technique for parallel processing of components in one machine (David et al., 2013). On the other hand OpenMI's control flow is pull-driven -- that is, the succeeding component requests information from the preceding component in a chain. A component handles only one request at a time before acting upon another request; however, the called process can handle the request by using multiple threads or by a parallel computing sessions, depending on the developer's preference. In OpenMI, functionalities such as parallel, cloud or high performance computing are yet to be developed (Knapen et al., 2013).

For non-HPC environments, integration frameworks can increase performance by applying multithreading techniques over distributed computers. Multithreading has a significant effect on system performance since subsystems can be executed in parallel on different computers (Tanenbaum and Van Steen, 2007). And if each process takes longer to execute (compared to the time to exchange data), multithreading will generate particular rewards in performance gains (Wainer et al., 2008).

2.4 Data interoperability

2.4.1 Issues on data interoperability

Data interoperability is the ability to correctly interpret data that crosses system or organizational boundaries (Renner, 2001). At the core of integrated modeling simulations is the exchange of data between workflow components. Implicit in our previous discussion of component preparation and workflow orchestration aspects is the need to ensure that data being transferred between components is: semantically interoperable – that is, has unambiguous meaning and is correctly mapped and, if necessary, translated; and structurally interoperable – that is, the dataset is formatted in the required form. Semantic interoperability issues for data are the largest hurdles in model integration (Argent, 2004). Differences in entity naming, scales used to represent space and time, and ways we represent a concept in relationship to others are the most common causes of semantic heterogeneity among models.

It is often necessary to perform additional computations on data as it is transferred from one model to another. Data mediation requires identification of the WHAT, WHERE, and WHEN of data in the context of each model (Moore and Tindall, 2005). WHAT is the name, definition, and quantitative description of the data represented by the variable, including factors such as units, allowable minimum/ maximums, etc. WHERE is location information and spatial resolution. WHEN describes temporal information and resolution. While space and time are obvious contexts, other "dimensions" must be considered as well: for example, differences between how components categorize land use, or human/ecological species demographics, chemicals, etc. must be resolved. Thus, unlike technical integration of modules, developing a 'generic' module that will handle data mediation among different models is quite difficult.

A related issue is, which component will perform the data mediation? An independent module be dedicated to this purpose, or will the participant models themselves mediate the data. In most integration frameworks, a dedicated module for data interoperability is developed, but participating models can also be responsible. For example, in OpenMI, the producing (source) component is responsible for identifying and extracting the requested data, performing appropriate conversions, and determining semantic mapping to variables from the receiving component (Moore and Tindall, 2005). It is assumed that data conversion "can be done in the most optimal way by the source component" (Gregersen et al., 2007). This indicates that, if the source component can be linked to five components, it should contain implementations of dataset conversion for each of them. Such

architecture has two problems: a duplication of effort by implementing dataset conversion functions for each model and a lack of manageability when the number of components grows because the method may not be extensible.

Frameworks generally vary in the degree to which they codify the metadata and conversions related to data exchange. Data interoperability is achieved either by hard-coding, using an in-house developed standard and annotations, or via a controlled vocabulary and ontology, as shown in Table 2.4 (Athanasiadis and Janssen, 2008; David et al., 2013, Rahman et al., 2003). In the following subsections we discuss data interoperability techniques used in the frameworks we reviewed.

Table 2.4: Brief comparison of data interoperability techniques used by integration frameworks.

	Hard-coding	In-house developed standard and annotations	Controlled vocabulary and an ontology
Primary characteristics	Uses an explicit rather than a symbolic name.	Use metadata for mediation.	Use vocabulary or ontology for mediation.
Advantages	Easier to implement.	Flexible and extensible.	Flexible, extensible, and accommodate change.
Disadvantages	Lacks extensibility and flexibility.	Usage is limited to small group.	Difficult to construct the vocabulary or ontology.
Examples	CMP	TIME, OMS3	SEAMLESS, FRAMES

2.4.2 Hard-coding for data interoperability

Hard-coding uses an explicit rather than symbolic name for something that will likely be changed[11]. In the context of data interoperability, this means that mapping and transfer of variables and related metadata from one component to another is achieved by direct connection between components. One ramification is that the mapping and transfer code must be generated for every combination of components that may share the data, resulting in duplication of code and high maintenance costs. Hard-coding is the most frequently used technique for data mediation because it is relatively easy to implement, but its implementation varies among frameworks. For example, in CMP (Moore et al., 2007) designers stress that adopting a set of conventions on component interfaces to address semantic and dataset interoperability is difficult. To reconcile inconsistency of data representation

[11] Hard-coding - http://whatis.techtarget.com/definition/hardcode

between components, they suggest using the component that translates the meaning of attribute names, quantities, and units of one component to another, or writing code in one component about the other components and adapting its interface at run-time.

2.4.3 Framework specific annotations for data interoperability

An annotation is a form of metadata which could be a comment or an explanation that is attached to text, image, or data (Yu, 2011). The second method of data interoperability is using application of specific annotations. Data mediation operations are performed based on information fetched from the annotations. A major advantage is that annotations for a component are reusable. Its disadvantages include that it is framework-dependent and that information presented in the annotation is dependent on how the developer defined the annotation.

Consider that TIME (Rahman et al., 2003) uses .NET metadata tags for classifying and documenting different attributes of models such as input-output, minimum and maximum allowed values, and units. OMS3 (David et al., 2013) similarly provides metadata information of classes, fields, and methods as annotations which contain information about component execution, unit and range constraints of data, and documentation. We consider the extension of the OpenMI standard in this category. The basic definition of the standard does not provide conventions for naming quantities and data operations (Gregersen et al., 2007). Data is expressed only in terms of data types, such as double, integer or string. However, the extension of the standard provides protocols (i.e., IBaseExchangeItem interface) to define how quantities, space, and time are described in the specific context of the data. This information can be fetched by calling functions, which provide metadata about the component such as *ValueDefinition*, *SpatialDefinition*, and *TimeSet*.

2.4.4 Controlled vocabulary and ontology for data interoperability

A controlled vocabulary is a list of "preferred terms" and definitions. When those terms are organized using a layer of interrelationships, axioms, and classes the result is ontology. As we progress toward more technical integration and automated data exchange, the need for standardized data interoperability via shared terminology and ontology becomes more important (Geller and Turner, 2007; Villa et al., 2009).

For example, the CSDMS framework handles semantic mediation by using Standard Names (Peckham, 2014) in the BMI and CMI interfaces. Semantic information about components can be found by accessing Model Information Functions of the BMI interface such as grid type and time steps. Dataset conversion functions are designed as reusable service components and can be accessed by CMI functions. Standard Names standardize not only variable names but also 'model assumption names,' based on a set of naming conventions. The naming conventions are based on 'object name + quantity name' and assumption names can be associated with variable names using an XML-based <assume> tag. Modelers are thus expected to use existing standard names in developing new models and contribute to identification and documentation of new standard names. Another example is the FRAMES (Whelan et al., 2014) integrated modeling framework which uses Dictionary files (DICs) for standardized data exchange between models. DICs consist of a list of standard names for variables and related metadata that component-models are expected to write to and read from. For dataset mediation, FRAMES has a service component that brokers data movement into and out of DICs. The mediation component performs quality checks on the data and executes any necessary unit conversions.

The goal of standardizing names is an ontology for automatic semantic mediation (Papazoglou, 2008; Peckham, 2014) which preserves semantics of the data and eliminates the need for custom-developed code for semantic mediation (Papazoglou, 2008). An ontology enables higher levels of reuse (Wang et al. 2002) and also enables framework-independent representation of knowledge on data structures (Rizzoli et al., 2008). Building ontology to address semantic interoperability across multi-disciplinary models is not a task for a few individuals (Argent, 2004) -- constructing a multi-disciplinary ontology requires huge collaborative efforts by different disciplines, as in Janssen et al. (2009).

Our literature review indicates ontologies are not commonly included in model integration frameworks. The GEO Model Web initiative (Nativi et al., 2013) suggests that Semantic Web technologies (RDF[12], OWL[13], SKOS[14], SPARQL[15]) can create understanding between models since they provide machine-understandable semantics; it also suggests that automatic semantic mediation can be realized only if the naming of service properties is unambiguously defined. Standardized model metadata is a means to achieving semantic interoperability.

[12] Resource Description Framework - https://www.w3.org/RDF/
[13] Web Ontology Language - http://www.w3.org/TR/owl-features/
[14] Simple Knowledge Organization System - http://www.w3.org/2004/02/skos/
[15] SPARQL Query Language for RDF - http://www.w3.org/TR/rdf-sparql-query/

Although four models are linked In SEAMLESS, ontology was built to avoid the hard-coding of semantic mediation. The ontology enables semantic mediation among models, indicators and data sources (Janssen et al., 2009), and also makes mediation process extensible to incorporate new models. The SEAMLESS ontology and its content is stored in a knowledge base, and model output is written to a database; the database and the ontology are linked by a relational database schema. The framework has a dedicated Knowledge Manager (KM) module to maintain links between components, manipulate data sources, and facilitate data exchange (Athanasiadis and Janssen, 2008). The KM is based on an ontology specifying the data exchange among participating models. Inter-component communication occurs via underlying databases and is dependent on the KM. The KM operates as a mediator by receiving data from models and mapping them to appropriate database tables.

2.5 Testing Integration

Testing verifies or validates whether a system satisfies certain specified requirements and improves the integrated system's quality. Integration of models is error-prone since it requires both scientific and programming efforts (Bruggeman and Bolding, 2014). Even though individual components are tested independently and may perform correctly, integration output can still produce unexpected results due to different assumptions made during integration (Voinov and Cerco, 2010). Testing remains difficult regardless of the integration strategy used (Ramamoorthy et al., 1992). As we have seen, there are many potential sources of error in designing and implementing software to facilitate data-exchange within integrated modeling systems, including matching of variables, conversion of units, use of metadata to perform semantic operations, and translation of space and time dimensions. Identifying and reporting specific sources of error (debugging) in integrated systems is also a challenge.

Integration of complex data and models is likely to propagate uncertainty throughout the model chain (Dubois et al., 2013) and uncertainty tends to increase as more models are chained together (Geller and Turner, 2007). Quantifying and communicating uncertainty in a model chain is challenging but crucial if we want to ensure the integration output will be accepted by the larger user community. In addition, time-stepping and numeric integration methods selected for linked components have their own effect on error and uncertainty in integration output (Belete and Voinov, 2016).

From the perspective of software engineering, testing occurs at different stages of development including unit testing, integration testing, system testing, and acceptance testing (Luo, 2001). Unit testing is the lowest level and is performed to verify correct component function. Integration testing

assesses whether interfaces between different components of a system are communicating correctly. System testing attempts to check end-to-end functioning of the system, and acceptance testing is done to deliver the system to end users. We must also consider emergent properties that appear when we link components into integrated systems (Ammann and Offutt, 2008). Best practices indicate software testing can be improved by characterizing the Why, How, How Much, What, Where, and When (Bertolino, 2007):

- Why – Are we looking for errors or we are trying to improve the usability of the user interface?
- How – How do we select test samples?
- How Much – How big should the sample be that we will use for testing?
- What – Are we going to perform unit test, component/ subsystem test, or integration test?
- Where – Are we going to test the simulated environment or the final context?
- When – At which stage of the product life cycle will we perform the testing?

Testing does not always get the appropriate emphasis due to time or cost constraints. For instance, Holzworth et al. (2015) points that the agricultural modeling community fails to give appropriate attention to testing. However, some frameworks have applied various testing approaches. While the naming was different, they provided certain testing functionalities to evaluate the system or its components. For example, FRAMES provided a "testbed" with error archiving functions (Whelan et al., 2014), from which users can closely examine system errors. Similarly, TIME provided a generic "test bench" (Rahman et al., 2003) in which model-component developers can test and calibrate a model. In ModCom (Hillyer et al., 2003) the framework distribution incorporates a "test suit" that covers all functionalities of the framework, and which is mainly aimed at debugging the framework libraries. In APSIM (Holzworth et al. 2014), automated testing is incorporated into the system to improve stability and robustness; whenever a modified version of code is submitted, automated testing runs the test suit.

When integrating models, a test plan should be part of development. The test plan should clearly outline the scope of testing and incorporate test cases, the list of functionalities to be tested, and by whom and when the testing will be performed. Depending on the context, testing can be done by developers, software testers, or end users. OpenMI testing, for example, was done in two phases -- first by the design team and then by external experts (Moore and Tindall, 2005). Testing was performed by eight universities and companies with the aim of migrating and linking different models with the provided tools and guidelines (Gregersen et al., 2007). Testing of integration can be done

with bottom-up or top-down approaches (Myers et al., 2011). Bottom-up testing begins with testing system components at the lowest level and progresses to higher-level combinations of components. In top-down, testing begins with the highest-level components and continues to individual components at lower levels. Once the testing strategy is selected, testing procedures can be automated which has advantages of saving time and repeatability. The challenge is that it may not be possible to automate everything we would like to test, and automated testing cannot always replace manual testing (Berner et al., 2005). This requires us to respond to questions like: How do we scientifically know that it works and that we did not create a false positive? For example, SIAT's developers used automated unit testing for code but not for the GUI (Verweij et al., 2010), the reason being that they found automated unit testing technology for GUIs is immature.

2.6 Discovery, accessibility and ease of use of integrated systems

Discoverability and accessibility of a model is significant in model integration. Despite the large number of modeling components, users generally lack tools to locate them. And if discovered, the next challenge is to gain operational access and evaluate their relevance to the user's problem (Booth et al., 2011). Finally, ease of use is important when deciding about model incorporation into a user's integrated system. For example, ESMF has been developed with a goal of enhancing ease of use, portability, and reuse in coupling climate and weather prediction models (da Silva et al., 2003; DeLuca et al., 2012). One of design goal of ModCom (Hillyer et al., 2003) is to provide a visual tool that enables model linkage and execution. Similarly, the TIME modeling framework (Rahman et al., 2003) includes a 'visually rich user interface' and reusable data handling and visualization components for integration of models. In the SEAMLESS-IF integration framework, the graphical user interface is designed to assist users in performing integrated assessment for alternative policy options (van Ittersum et al., 2008). The GUI offers easy access and display of system functionalities and outputs.

Preparing software for easy use requires knowing the needs of target user groups of the framework. For example, the Biophysical Model Applications (BioMA[16]) framework was developed to rapidly bridge model development from prototypes to operational applications with re-usable components. In this case, target users are both application users and advanced users. Ease of use features for application users relate to running and viewing available

[16] BioMA - http://bioma.jrc.ec.europa.eu/index.htm

model simulations, while ease of use features for advanced users include creating and developing models.

With the advance of web technology, presenting model interfaces and integration frameworks in web pages has become more popular since it significantly increases model discoverability, accessibility and usability. Web-based user interfaces that target specific communities – e.g. scientists, policy makers, and the public – improve access to data and models and facilitate model reuse. "Apps for the environment"[17] developed by the U.S. Environmental Protection Agency is a good example. The introduction of a browser-based Web Modeling Tool[18] (WMT) by CSDMS to replace the older desktop-based Component Modeling Tool (CMT) also points in this direction. Improving the WMT's ease of use is a focus of current development. Likewise, Systo[19], a web-based model whose technical design is based on the desktop-based Simile[20], InsightMaker[21] and STELLA[22], is a successful effort to make a model more accessible and easier to use.

Providing models via the web is not without challenges. As Brooking and Hunter (2013) point out, these challenges include managing large volumes of data, complexity of the required software platform, and computational demand for model execution are barriers that limit sharing and re-use of models over the web. Similarly, Nativi et al. (2013) point out developing a flexible architecture to link distributed components, minimizing interoperability agreements, optimizing performance, and securing availability in the long term are challenges in presenting models on the web.

Presenting models using web pages also creates a challenge since model usage is constrained by how the web page presents it. For example, users may want to display model output as part of another application, which leads to providing a "model as a service", making it available as a web service (Geller and Melton, 2008; Geller and Turner, 2007; Roman et al., 2009). This approach makes models and their output more accessible, more comparable, more scalable, and can be implemented using a variety of approaches (Nativi et al., 2013). One example of such a platform is eHabitat (Dubois et al., 2013) which is part of the European Union Digital Observatory for Protected Areas (DOPA[23]) project. In eHabitat, models and data sources are presented as a pool of OGC WPS services; using a web-based interface, users can select and assemble them like Lego blocks. Another example is the iPlant

[17] Apps for the environment - http://www.epa.gov/mygreenapps/
[18] CSDMS WMT - http://csdms.colorado.edu/wiki/WMT_information
[19] Systo - http://www.systo.org/
[20] Simile - http://www.simulistics.com/
[21] InsightMaker - https://insightmaker.com/
[22] STELLA - http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx
[23] The Digital Observatory for Protected Areas - http://dopa.jrc.ec.europa.eu/

cyberinfrastructure (Goff et al., 2011) which provides several biology-related applications as a web service API so bioinformatics experts and software developers can embed them in their tools.

Some modeling frameworks use purposely-designed semantic technology features to improve discoverability of models and related resources. For example, Galaxy (Goecks et al., 2010) – a computational framework for life sciences – uses tagging (labeling) to describe resources. During simulation, the system automatically tracks input datasets, tools used, parameter values, output datasets, and a series of analysis steps that it stores as history. The user can add annotations about analysis steps. The authors report that tagging supports reproducibility of experiments. Similarly, the system biology data and model platform - SEEK (Wolstencroft et al., 2015) – provides a web-based environment for day-to-day collaboration and public access. It uses the Resource Description Framework (RDF) to store metadata of resources that facilitate data and model exploration. All of these efforts indicate that incorporation of purposely-designed semantic technology features can improve discoverability of models beyond simple search operations.

2.7 Discussion and conclusions

Our goal was to identify existing strategies and recommend additional ones that facilitate efficient reuse and interoperability of science-software components to enable cost-effective construction and execution of integrated multi-disciplinary modeling systems. We organized the review around the model integration process that includes pre-integration assessment, preparation of models for integration, orchestration, data interoperability, and testing. We highlight various methods employed by developers to achieve each process step. The following observations and conclusions emerged from this review. We expand this in the final section with recommendations intended to foster a community-wide conversation to improve implementation of the overall process, and design and deployment of modeling components in particular.

1) There is wide variation in strategies for implementing and documenting model integration across the development community. The process is driven by a science-based need to explore processes, problems, and solutions related to the environment that uses software-based tools to express the knowledge and execute modeling workflows. Science-based needs determine the requirements for the integrated software system. The variation in strategies is largely related to computer-based technologies (hardware and software) rather than differences in science-based requirements.

2) Frameworks that facilitate construction and orchestration of modeling workflows are complex, requiring significant resources to build and a steep learning curve to understand and utilize them for those outside the development and targeted user groups. We found that the literature describing the overall process and frameworks, in particular, varies widely in aspects of the integration process and frameworks that are discussed and in the level of detail in those discussions.

3) There is wide variation in strategies for orchestrating execution of workflows within the frameworks, and preparing modeling components for assimilation into the frameworks. With few exceptions, current practice is to design and implement integrated modeling systems to satisfy requirements of individual projects and development/user group preferences. This strategy has resulted in a large number of frameworks, but little reuse of the modeling components (and, thus, the embedded science) among the frameworks.

4) Data interoperability involves interpreting and converting data produced by one component to satisfy content and format requirements of components that will consume it. Again, there is wide variation in how this is implemented. Methods include explicit and hard-coded mapping of variables, use of metadata that describes data attributes (units, min/max range), use of controlled vocabularies and ontologies to ensure semantic consistency, and intermediate components designed to manipulate data (translate, interpolate, aggregate, disaggregate) to resolve dimensional conflicts (spatial/temporal grid conflicts). Such variation in component design creates significant barriers to interoperability and sharing of science, expressed as software components, among frameworks.

5) Testing of integrated modeling systems verifies operational integrity of the system design as implemented, i.e., the system correctly formulates and processes the overall workflow from system inputs, to component-level data exchanges, to generation of results. Testing is particularly challenging in integrated systems because of the increased potential for error and error propagation. The reviewed literature paid relatively little attention to details on this topic. It appears that formal testing (with documentation) is often limited due to resource constraints and some developers conduct and document testing from the component- to the system-level. Systems with limited testing are generally operated by the developers only, while systems with more formal testing are easier to transfer to users. This lack of documentation of testing limits acceptance of the system by developers outside the original team.

6) Discoverability, accessibility and ease of use are fundamental requirements for users. For existing systems, discoverability is typically limited to local knowledge within a group, the literature, and Google searches. As stated above, the literature often does not provide sufficient detail to determine important characteristics of the software required to make a decision about pursuing its reuse. On a positive note, we see a movement toward the web as a platform for deployment of modeling components which appears motivated by an emerging desire to address interoperability by making models available as a service, thus eliminating issues of incompatible computer languages and operating systems. As a secondary benefit, model discoverability is also enhanced with web-based deployment.

Our overall conclusion is that integrated modeling can be significantly enhanced if the global community of developers specifically focuses on cross-framework reuse and interoperability in design and implementation strategies. This will take us beyond sharing science knowledge via literature and move us toward sharing this knowledge more efficiently and cost-effectively in computer-based software form. Based on our review, we conclude that this is not only possible but that it is already occurring, although in an ad hoc rather than an organized manner. In the next section we make several recommendations to increase awareness of these possibilities and focus on specific efforts.

2.8 Recommendations

Using the perspective gained from this review we organized a list of recommendations for moving forward as a global community in developing integrated modeling systems. Note that the recommendations do not include details such as specific standards that should be utilized or developed, but rather where standards apply and what they should include (i.e., the science content). We chose this approach despite the fact that specific standards already exist (e.g., OpenMI, OGS WPS) because we believe it is more important to come to a community-wide awareness of the benefits of reuse and interoperability and to increase participation in them. Our recommendations reflect the software engineering concept of "separation of concerns" and the science-based concept of ontologies. Separation of concerns focuses on developing software components that perform a specific, limited set of functionalities. As more functionality is packaged in a single piece of software, its flexibility and use (and reuse) becomes more and more constrained in an integrated context. Ontologies are a structured way to represent knowledge about a domain; they define and describe concepts (things) included within a knowledge domain, as well as their related properties and relationships.

We begin by identifying three principal elements of integrated modeling systems: user interface, model integration infrastructure and tools (i.e., frameworks), and science-based components. We recommend properties of each and the communication-based relationship between them. Each element can be further separated into lower-order elements, properties, and relationships.

User Interface

The user interface should be "thin" and focus on specific problems or classes of problems that give users the ability to select or construct a scientific workflow involving data, modeling, and tool components relevant to the user-specified problem; and visualize and process results of workflow execution. Workflows should accommodate at least two levels of complexity: first, a straightforward linking of multiple data, model, and tool components to enable both feed forward and feedback interoperability; and second, it should accommodate special "system"-level functionality such as data acquisition and harmonization tools, ability to perform system-wide sensitivity and uncertainty analysis, and analytical tools for synthesizing intermediate and endpoint data processing. These workflows define the requirements for design of the model integration framework and its components.

Users should have access to a repository of components and sufficient metadata about them to determine which ones are relevant to a user-stated problem and are interoperable. Metadata about model components (aka meta-models) should be standardized to ensure machine-readability and processing. The repository may be local (to an organization or a development group, for example) but, ideally, would be linked to a global collection of socio-environmental science software components.

Output of the user interface should be a standalone (and, ideally, standards-based) description of the workflow to be executed. The interface should also provide access to tools that consume and process results of the integrated modeling execution.

Model Integration

The model integration framework implements workflow produced via the user interface. Important elements of the integration process, as gleaned from the review, include execution management, semantic mediation, dataset conversion, and error handling. Integration tools or complete frameworks may be designed to optimize model system execution on high performance computers, parallelization of execution (e.g., leveraging the cloud), and web-based integration.

Science Components

Science components should be designed and implemented in a framework-independent manner. We encourage individual scientists to publish software in a way that can be discovered, understood, and integrated within any user interface and model integration system. Essential features associated with the design of components to meet these high-level requirements include:

a. An explicit API. At the highest level, it enables initializing, running, and closing functions. At a secondary level, it describes and enables programmatic access to science knowledge and functionality expressed in the component.
b. Metadata for input/output/internal variables that describe at least the :
    i. Meaning of variables, including assumptions made about variables and processes
    ii. Measure/units unambiguously
    iii. Valid range of values, if appropriate
c. In the context of a) and b), the content should be described using a controlled vocabulary and, eventually, a full ontology to describe relevant concepts and relationships.
d. Software is open source and available via public access version control systems such as gitHUB.

To develop the global repository, components must be published with science content and functionality to enable discovery, understanding, and integration. Their publication must include metadata that fully describe inputs, outputs, and internal workings of the component (aka black box). This will require controlled vocabularies and an ontological framework for describing data and modeling science. To achieve this, the modeling community should reinforce collaboration on developing and improving standards which can be used across frameworks. Adoption of the OpenMI Standard v2.0 as an OGC standard is an example of one such effort.

Chapter 3

Designing the Distributed Model Integration Framework – DMIF*

* This chapter is based on:
Belete, G.F., Voinov, A., Morales, J. Designing the Distributed Model Integration Framework – DMIF. (In Review, Environmental Modelling and Software)

Abstract
Integration of models requires linking of components, which are developed by different teams, using different tools, methodologies, and assumptions. Participating models may even operate at different temporal and spatial scales. We describe and discuss the design and prototype of the Distributed Model Integration Framework (DMIF) that links models deployed on different hardware and software platforms. We used distributed computing and service-oriented software development approaches to address the different aspects of interoperability. To demonstrate technical interoperability of heterogeneous models, we developed reusable web service wrappers for models created in NetLogo and GAMS modeling languages. We investigated automated semantic mapping of text-based input-output data and attribute names of components using word overlap semantic matching algorithms, and using an openly available lexical database. We found that for short text-based input-output data and attribute names of components, word overlap semantic matching algorithms work better than applying a lexical database. This holds true for both standardized and non-standardized short text. This is mainly because such text does not include contextual information about data. Furthermore, word overlap semantic matching algorithms can be applied to search for other components that can possibly provide data for a given component (1) if a model repository uses standard names for attributes of components and (2) if metadata of components are made available through an API. We also demonstrated automated unit conversion in semantic mediation by using openly available ontologies. This technique helps to avoid significant amount of reinvention by framework developers, and opens up the modeling process for many stakeholders who are not prepared to deal with all the technical difficulties associated with installing, configuring, and running various models. It also offers functionality to further link to other appropriate models and datasets. As a proof of concept, we implemented our design to integrate climate-energy-economy models. Different modeling groups can apply our design to link a wide range of models, and it can improve the reusability of models by making them available on the web.

3.1 Introduction

Models are simplifications of reality and developed with the objective to understand a concept or system, to analyze what the future states and trends may look like, and if possible to come up with appropriate management decisions and mitigation or adaptation strategies. Thousands of computer models have been developed. However, complex problems such as climate mitigation require interdisciplinary knowledge and assessment from many domains including climate, hydrology, energy, economy, land use, behavioral sciences, etc. The complex and interrelated nature of such real-world problems requires holistic system-of-systems thinking (Laniak et al., 2013). In this case it is not practical, perhaps impossible, to construct a

single model that could simulate such complex processes (Gijsbers and Gregersen, 2005). Integration of models and tools may be a solution (Stoorvogel, 1995). It also reuses existing models and it is faster and less expensive than reengineering legacy systems (Madni and Sievers, 2014).

During integration of models, we should understand that models can be developed using different assumptions and semantics, different methodologies, tools and techniques, may function at different temporal and spatial scales, may have different levels of complexity, etc. Integration of models assumes linking such heterogeneous models together into an operational model chain (Knapen et al. 2013), or rather a network with loops and feedbacks, where one model down the chain can also feed input back into a model above. This requires addressing interoperability at technical, semantic, and dataset level (Belete et al., 2017). Based on this we define model integration framework as a set of software libraries, classes, and components that enables one to manage technical, semantic, and dataset aspects of interoperability.

Integration of models requires mediation that goes beyond merging information and data that use different schemas. Computer-based models contain sophisticated knowledge statements, which may be represented in different ways. Due to this integration of models requires understanding of the different contexts of the models involved. It also requires mechanisms to modify the incoming information so that it fits to the assumptions, conditions (rules), and processes in the data-receiving model. Best practice indicates that this can be achieved by providing dedicated components or modules that handle context-based interpretation and semantic mediation. For example, such a mediator component is known as the Knowledge Manager in SEAMLESS (Athanasiadis and Janssen, 2008) or the Semantic Discovery Broker in eHabitat web processing service (Dubois et al., 2013). One of the objectives of modeling is to provide information to decision makers. Despite the large number of available models, decision makers lack easy access to models to evaluate alternative scenarios (Booth et al., 2011). Models that do not require installation of special software, and that do not need special training, are more accessible but rare. With the advance of web technology, presenting models and integration frameworks on the web is becoming more popular since it can significantly increase model accessibility and sharing, i.e. when a model can run in a normal web browser requiring no additional software installation. However, the volume of data involved, complexity of the software platform required, computational demand for model execution are identified as barriers that prevent sharing and re-using models over the web (Brooking and Hunter, 2013).

Although availing models through web pages is useful, usage of models will still be constrained by the way the web page presents the model. For example, users may want to directly access model output and display it as part of another application. This leads to the idea of providing a "Model as a Service" (Geller and Turner, 2007; Geller and Melton, 2008; Roman et al., 2009). Presenting models as web services has the benefits of making models and their outputs more accessible, easier for model comparison, scalable, and implementable using a variety of approaches (Nativi et al., 2013;, 2013). A web service developed using one programming language can be accessed and consumed with applications developed by using a number of other programming languages, i.e. without requiring intermediary language interoperability tools. The Interoperable nature of services gives the opportunity to integrate legacy models by presenting them as web services (Goodall et al., 2013; Granell et al., 2010).

Currently, one of the main challenges facing the integrated environmental modeling community is lack of interoperability across independently built systems (Goodall et al., 2011; Laniak et al., 2013). This means that besides improving accessibility of models and data, we also need a mechanism for integration across disciplines and models developed using different platforms. There is an increasing need for methodology that enables to build a system-of-systems (Butterfield et al., 2008) by connecting independent component systems and making them interoperable.

The context in which a certain model is used can vary significantly. Besides, a model can be linked to a number of models in different ways. Users have their own integration requirements. Integration scenarios identified and designed by a certain group of modelers or developers may not satisfy integration requirements of the whole user community. Even one particular user can come up with a number of integration requirements. On the other hand, only a small subset of end users may have the programming skills needed to modify the source code of integration frameworks in accordance of their requirements. Due to this, there is a need for tools in which users can select certain models and link them without the need for additional design, coding, debugging, etc.

In this paper, we present the design methodology of a distributed model integration framework. We present the approach used to convert heterogeneous and independently built models into plug-and-play components, and the mechanisms used to automate some of semantic mediation tasks. In addition, we present a case study in semantic mediation using semantic matching algorithms and lexical databases. We also introduce interfaces that enable runtime access and integration of web service based models without the need of additional coding. We also discuss the limitations

of such approach. After all, models are always built for a purpose and there is no guarantee that when we reuse a model we will be using it in the same way as intended by the original model construction. This is how so called 'integronsters' (Voinov and Shugart, 2013) are created, which we certainly want to avoid. User mediation and pre-integration assessment is the only way that we can safeguard ourselves from unintended misuse of models. We explore how integration interfaces and semantic mediation can assist in such user guided model evaluation.

The remainder of the paper is organized as follows: Section 3.2 presents the design criteria of model integration frameworks. Section 3.3 describes the architecture of the Distributed Model Integration Framework (DMIF). Section 3.4 provides information on how we enabled technical interoperability by presenting models as web services. Section 3.5 describes methodology to build semantic mediation module of model integration frameworks. This section also presents algorithms for semantic matching of text-based input-output data and for searching of components using attribute names of components. Section 3.6 introduces runtime access and integration of web service based models at the GUI level. Section 3.7 discusses additional issues that we should consider in developing integration frameworks, and Section 3.8 provides our conclusions.

3.2 Design criteria for DMIF

Our aim is to provide a verifiable design of a model integration framework that helps to link multidisciplinary heterogeneous models distributed over various software and hardware platforms in a meaningful way. Design should follow user's requirements, and it should be verifiable against those users' requirements. There are many factors that a model integration framework should consider (Belete and Voinov, 2014; Belete et.al, 2014, 2017). There is no ideal integration technique, which best suites all kinds of model integration requirements. However, as a guideline we know that an integration framework should consider the following design criteria. The design should:

- support models developed using different programming languages,
- include models hosted on different hardware and software platforms,
- access models located anywhere on the Internet,
- keep independently developed models autonomous,
- avoid reinventing whenever reuse of resources is possible,
- provide functionalities as reusable components,
- be extensible without disturbing the existing system,
- be accessible on the web.

Given these criteria, our design focuses on interoperability of heterogeneous models. To realize this we need to establish a few well-known dependencies (Rosen et al., 2008) among such models that enable them to exchange data and to collaborate.

3.3 The DMIF framework

3.3.1 Background

To meet model integration requirements, we considered Best Practices of System Integration, Enterprise Application Integration, Distributed Systems, and Software Design Patterns (Erl et al., 2008a; Gamma et. al., 1994; Tanenbaum and Van Steen, 2007). We observed that web services have a good potential to transform independently built models into plug-and-play components. This is because web services are platform independent, self-contained, interoperable applications that are accessible over the web. Moreover, any piece of existing code can be transformed into a network available service (Papazoglou, 2008). Besides, those independently built service-based models can be linked into system-of-systems by using Service-Oriented Approach (SOA) of software development. SOA is a design approach in which services are linked together based on the following principles: participating services are expressed using standardized metadata, loose coupling of services, reusability of services, autonomy of services, distribution of services on different platforms, discoverability of services, and composability of services regardless of their size (Erl et al., 2009). Due to the nature of SOA, it inherently addresses integration issues like loose coupling, interoperability, and platform independence, as a result it helps to curb integration challenges (Erl, 2008b).

3.3.2 Architecture of the framework

The Distributed Model Integration Framework (DMIF) is designed with the aim to handle linking of various heterogeneous computer-based models. Prototype of DMIF can be accessed from the COMPLEX[24] project model repository page. In this section we start describing DMIF by presenting its architecture (Figure 3.1), which presents the logical organization of the system in terms of software components (Fowler, 2003). Figure 3.1 depicts the high-level view of the integration framework. Besides, software architecture demonstrates how concepts in the problem domain are handled by the solution domain (Butterfield, et al., 2008).

[24] DMIF - http://owsgip.itc.utwente.nl/projects/complex/index.php/2-uncategorised/46-model-integration-framework

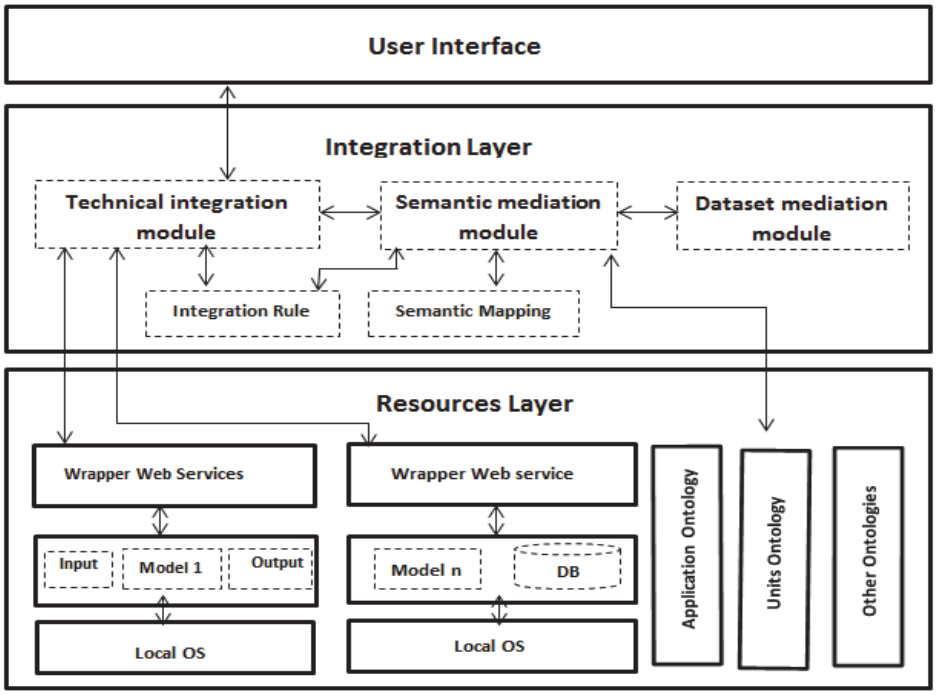


Figure 3.1: Architecture of the distributed model integration framework - DMIF.

The architecture of DMIF follows a layered structure based on principles of separation of concerns and distributed computing. Following the separation of concerns principle (Laplante, 2007), the system is a composition of a group of logically related elements. As a result, it relieves models from managing integration related tasks, for example, interoperability and data conversion. Following the distributed computing paradigm (Tanenbaum and Van Steen, 2007), a system is a collection of subsystems deployed on different heterogeneous platforms that communicate with each other by exchanging messages (Papazoglou, 2008). Distributed systems hide the fact that the resources and processes are distributed across multiple computers, and the system presents itself as if it were hosted only in a single computer. Due to this, models can be anywhere on the Internet (Nativi et al., 2013).

In the context of model integration frameworks, distributed computing paradigm provides the following benefits:

- Models can be developed using different programming languages and can be deployed in different operating systems,
- Participating models are kept autonomous,
- Models can run concurrently on different machines using multithreading techniques – which has significant performance gain in linking model with 'longer' execution time (Wainer et al., 2008),

- Distributed systems are scalable, more resources, such as, models, databases, files can be added through time,

As shown in Figure 3.1, a model may read its input from a file, database, or directly from the user interface. Web services developed on top of models so that the model can interoperate by exchanging message with each other. However, web service based approach has limitations since it uses verbose text-based protocol to exchange messages, which means web services use larger message size than binary protocols. Besides, web service based approach has challenge due to limited reliability of the communication between services over the Internet. Detailed discussion of different parts of the architecture is presented in the following sections.

3.4 Wrapping and technical interoperability

3.4.1 Enabling technical interoperability

Technical integration of models requires automating data exchange between models and making them jointly executable. As Knapen et al. (2013) indicate, technical integration of models can be achieved in five different approaches: soft linking, using scripts, proprietary monolithic, proprietary loosely linking, and with open standards. As mentioned earlier, the strategy we used to build the model integration framework is W3C open Web Service standards[25]. We assume that all participating models and tools can be presented as web services. Then the interaction between models is managed by the technical integration module using SOA principles. The technical integration module is the core for the whole integrated system, since it is responsible for establishing the connection between models, for routing, and messaging. It is the composition engine (Alonso, 2004) of service-based models.

Presenting models as web services will make them componentized. Here a software component is defined as an independent unit of functionality, with a well-defined interface, and internal states are not externally observable (Clemens et al., 1998). Componentization has the benefits of reusability, interoperability, limited dependency on other components, dynamic linking capability during runtime, etc. (Peckham et al. 2013, Rizzoli et al. 2008). Since we are integrating already existing models, we prefer to avoid rewriting them when presenting them as web services. We turn models into web services by wrapping them, and transforming them into plug-and-play autonomous components. A wrapper is an interface that encapsulates a model and provides access to the model in required way. For example, in

[25] http://www.w3.org/standards/webofservices/

CSDMS the BMI[26] wrapper is a set of functions named initialize, update, finalize, run_model, get_attribute, etc. that provide information about the model and run it.

Depending on the ease of implementation, wrapper web services can be developed by using different programming languages. For example, we developed a C# based wrapper for an economic model created using GAMS[27] programming language, and a Java based wrapper for a Household Energy Demand model created with NetLogo[28]. The challenge here is that models are developed independently, they are not designed to interoperate with other models (Nativi et. al., 2013) and they do not have knowledge about other member models (Butterfield et al., 2008). Yet for integration purposes, we have to come up with an interface that will help a model to act as a plug-and-play component.

By choosing the web service approach we offer the ultimate flexibility for using the models. As long as they are wrapped as web services they become available for further integration. Web service based wrapping addresses our main design goals: it supports models located anywhere on the Internet, developed using different programming languages, hosted on different hardware and software platforms, and it keeps independently developed models autonomous. The amount of work in developing wrappers depends on how much we need to adapt the existing model. If the model is built using a modular approach, developing wrappers may be straightforward; but if a complex model is implemented as a monolithic function, we may also need to modify the model code. Still, from the software development viewpoint, as long as the wrapper produces a web service, we do not need to impose any additional requirements on its functionality.

What still remains problematic is the descriptive part of the model and how it will be delivered by the web service. In fact, in addition to wrapping the model code to expose it as a web service, we need to put extra effort in properly documenting and exposing the model. Machine-readable description of web services, the WSDL[29] file, is designed to provide metadata information about the service. The contents of the WSDL file mainly focuses on technical issues such as data type definitions, set of operations or functions provided by the service, binding information, etc. However, the WSDL can also be used to provide semantic information about models that will be used during integration. For example, WSDL has optional Document element aimed for human readable documentation. This element can be used to incorporate

[26] BMI - http://csdms.colorado.edu/wiki/BMI_Description
[27] GAMS - https://www.gams.com/
[28] NetLogo - https://ccl.northwestern.edu/netlogo/
[29] WSDL - https://www.w3.org/TR/wsdl

standardized model metadata that will facilitate searching and integration of component models.

3.4.2 Web service orchestration

In DMIF, models do not interact with each other directly. Instead, the technical integration module handles the interaction between models. Service orchestration (Erl, 2008a; Papazoglou, 2008) is one of the main tasks of the technical integration module. It serves as a mediator (Gamma et al., 1994) that encapsulates the interaction between models. Based on the design principle of separation of concern, models are not expected to maintain information about 'where' other models are hosted and to support a protocol for 'how to' communicate with them. If we require models to maintain such information, then:

- a model should maintain a list of other models that could be linked to it,
- whenever a new model joins the framework, existing models should incorporate new information,
- when there is change in location or some modification in the interface of a certain model these changes should be reflected in the data maintained by existing models.

All this results in information duplication and hinders scalability.
In our approach, only the technical integration module maintains reference information about participating models. By using this information, the technical integration module can access both local and remotely hosted models. It also coordinates the communication between models so that the system gives a sense of a single aggregate service. Addition or removal of a component does not affect other existing components and this makes the framework extensible.

The workflow of orchestration of services varies depending on the type and number of models involved (Yang et al., 2012; Zhao et al., 2012). This is because the communication between models could be unidirectional, bidirectional or iterative. As a result, each case of integration will have its own workflow. However, as shown in Figure 3.1, in all cases integration starts at the user interface level. The user interface passes user inputs together with the sequence of execution to the technical integration module. The technical integration module then executes the models and receives their output. It also handles interaction with the semantic mediation and dataset conversion modules.

This design approach allows users to select the components they prefer to couple at the GUI level and this makes them responsible for the final decisions made about candidate models and their linkage mechanisms. User

defined coupling presents significant challenges since users are free to couple any available component models, which could possibly give birth to 'integronsters' (Voinov and Shugart, 2013), i.e., products that are valid in terms of component coupling but useless as models. On the one hand, during design time based on the logical and contextual information of participating components we can identify input-output data exchange between components, and make sure that the accompanying semantic mediation and dataset conversion functionalities are in place. We can document all such validated coupling potentials in a file and use them as integration rules (Belete et al., 2014). During run-time, the technical integration module will validate the user-defined coupling against the integration rule before proceeding to service orchestration. If the proposed coupling is not available in the integration rule, the system will notify the possibility of an 'integronster' but the user can override the suggestion and continue. On the other hand, this checking can become automated once adequate model documentation standards become accepted and implemented. Only when wrapped models contain sufficient semantic metadata to allow the system to determine whether the composition results make sense, the integration process will allow some level of automation. Until then the user will have to be involved for the final checking and quality assurance of the proposed integration schemes.

The other major issue we need to consider during orchestration is optimization of performance of the integrated model. In reality, many processes that are represented by integrated models happen in parallel. However, it is common practice to represent such parallel processes using sequential execution of simulation models. For better performance of simulation, our design approach supports parallel execution of models using the multi-threading technique (Tanenbaum & Van Steen, 2007) – i.e. the technical integration module calls the participating services as different threads of a single process but each of those services will run in parallel on their hosting machines. Currently, almost all modern computers have multi-core CPUs, and the parallel execution approach gives the opportunity for effective utilization of available processing resources (David et al., 2013). In distributed systems, the multi-threading technique has a significant effect in the performance of the system since subsystems can be executed in parallel on different computers. Especially if each of the processes are taking longer execution times, multi-threading will have rewards in performance (Wainer et al., 2008).

The other point in relation to performance optimization is the size of data to be exchanged, which could be dependent on the way we developed wrappers. A model may return several variables as output. In designing wrapper services, we can include either all these variables or only some of them as

the output of the wrapped model. If the wrapper includes only the required variables for the data exchange then we can minimize performance overhead due to exchange of large size of data over the network. However this at the same time can restrict the application of wrappers making them coupling specific: when a model is coupled with yet another model a new wrapper will be needed if other model output is to be exposed.

3.5 Semantic mediation

3.5.1 What is involved in semantic mediation

Semantic mediation is a means to link knowledge represented in different ways. In integrating models, semantic mediation is done based on the contextual information of participating models, and metadata information of models is the main source of contextual information. Depending on the extent of the semantic mediation automation, it requires providing a certain level of intelligent translation of concepts. This may include gathering and analyzing information based on the contexts involved. The challenges in semantic mediation are:

- How to minimize hard-coding in semantic mediation?
- How can we provide standardized semantic mediation?
- How can we accommodate the vast semantic difference among different models?

In semantic mediation, hard-coding or hard-wiring should be minimized (Uschold, 2003) since hard-coded semantic mediation is fragile whenever there is change in representation of model attributes, data structure, and data itself. Hard-coding can be minimized by standardizing the way we represent concepts and build structures that maintain the semantic relations between different concepts. Depending on the specific context, the level of detail in which a concept is represented varies. Based on this, such artifacts as Glossary, Lexicon, and Ontology have been used to organize concepts (Buccella et al., 2009; De Nicola, et al., 2009). Among these an ontology contains more organized information since it includes a hierarchy of concepts, relationships between concepts, and cardinality, which are not present in a glossary. Ontology preserves the semantics of the represented concept and this helps to eliminate the need for custom developed code for semantic mediation (Papazoglou, 2008), since ontologies are based on a set of formal expressions upon which a computer can reason (Janssen, 2009). They also enable higher level of reuse (Wang et al. 2002). In general, ontologies together with semantic mapping and translation techniques are crucial to minimize hard-coding in semantic mediation (Li et al., 2011; Uschold, 2003). In the availability of multi-disciplinary ontology, the metadata of participating models can be mapped and translated using semantic processing techniques.

However, building ontology to address semantic interoperability across multi-disciplinary models is not an easy task and requires concerted efforts of many people (Argent, 2004). Constructing a multi-disciplinary full-fledged ontology requires huge collaborative efforts among different disciplines. Again, we find a need for some model documentation standards and metadata for the provision of reusable and standardized data conversion functionalities.

In DMIF, we classified the semantic mediation task into two categories:
1) semantic matching which includes translating text data and mapping of variable names and,
2) unit conversion.

3.5.2 Semantic matching in integration of models

We developed a semantic matching module for text data and variable names used in models. For both cases, first, we used the word overlap matching (Canhasi, 2013) or token-based matching (Cohen, 2003) algorithm that does not consider hierarchical semantic relation between concepts, and then we used a lexicon database based matching algorithm that considers semantic relationships between concepts.

The semantic matching algorithms

Suppose we are linking model 1 with model 2. Our aim is to match a list of available text-based input data/variable names in the second model, with a list of text-based output data/variable names from the first model. While in many models we still see variable names that are quite semantically meaningless, there is a trend towards choosing informative naming conventions (Peckham, 2014). First, let us consider the word overlap matching algorithm that we developed. The algorithm has two phases. The first phase is direct text matching. The algorithm tries to find a text that matches exactly, and if it finds a match then it will assign matching index value as 1, which means the two texts match 100%. The next phase is to do token-based matching for those texts that did not match directly. In this phase, the algorithm will try to find a number of matching words in the two texts under consideration, and it manages the matching by using the following steps:

- split (tokenize) the first text into words,
- split (tokenize) the second text into words,
- count the number of matching words between the two texts,
- compute matching index

To compute the matching index, we use *Jaccard similarity coefficient* (Achananuparp et al., 2008; Sarawagi and Kirpal, 2004), since it measures the size of intersection between two given texts, it is suitable to represent

the word level similarity measurement (Niwattanakul et al., 2013). The Jaccard similarity coefficient (J) for texts A and B is computed as:

$$J(A, B) = \frac{|AnB|}{|AuB|} = \frac{|AnB|}{|A|+|B|-|AnB|} \qquad 0 \le J(A, B) \le 1$$

If *A* and *B* are both empty then *J* (*A*, *B*) = 1

Where the notation *n* stands for intersection and *u* is union between sets A and B.

However, the word overlap algorithm does not completely satisfy our needs, because it is common that different modelers can represent the same concept using different words and phrases, which have similarity in meaning. For example, suppose we want to check the similarity of 'car' and 'automobile' using the word overlap algorithm. The algorithm returns matching index of zero. Nevertheless, semantic mediation requires a matching algorithm that considers relations between texts based on meaning of words.

Due to this, we developed the second semantic matching algorithm that uses a lexicon database that considers hierarchical semantic relation between concepts. We used an openly available lexicon database called WordNet[30] that consists of over a hundred thousand words with meanings and a complex architecture of a network of words. In the database nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called synsets, each expressing a distinct concept. The main relation among words in WordNet is 'being synonymous', e.g. car and automobile are synonymous. WordNet computes similarity between two words using the following steps:

- find shortest path length (in links/edges) between the synsets containing the compared words w1 and w2,
- find the depth of the first common subsumer (or superset) of both synsets,
- compute the similarity index based on the path length and depth information.

Like the word overlap algorithm, the similarity index value for WordNet based algorithm ranges between zero and one. For example, if we compute similarity between 'car' and 'automobile' it returns one.
In general, applying the two semantic matching algorithms has challenges. The first challenge is that it is difficult to do the matching for acronyms and abbreviations that are commonly used by the modeling community. The second challenge is to decide where to put the margin for similarity index values after which we can consider matching as 'valid'. We observed that, in some cases matching index value of 0.8 may not be good enough to consider

[30] WordNet - https://wordnet.princeton.edu/

the matching as 'valid' and in some other cases an index value of 0.3 might be enough. For better results, setting the margins requires the decision of the human user. As test cases, we applied these two semantic matching algorithms in matching text data and variable names of models.

Semantic matching of text data

Case study 1: The first case study focuses on matching text-based input-output data to link an Airports information web service[31] with a Global Weather[32] model that is available as a web service. This example is a case of the integration of data as a service with a model as service. The objective of the linking is to generate an enhanced list of airports in a country, which includes current weather information such as *visibility, sky conditions, pressure, temperature, relative humidity, and dew point*. This can be done since the Airports web service provides a list of airports in a country, and the Global Weather web service provides weather information of a given city in a country. Name of cities are text data to be passed from one web service to the next web service. Consider, for example, the case of 'Netherlands', as being an input text data given to the two web services. Table 3.1 shows that the two web services represent the names of cities in different ways.

Then, given the list of names of cities, we can handle the semantic mediation by direct mapping between the names. The problem with this approach is that we are not controlling these web services while the owners can modify the content of the data anytime. Therefore, whenever the content of data is modified we have to modify the mapping. Due to this hard-coded semantic mapping is fragile. The solution is to develop an automatic semantic matching algorithm that manages the semantic linking between the text values.

[31] Airports information web service - http://www.webservicex.net/airport.asmx
[32] Weather web service - http://www.webservicex.net/globalweather.asmx

Table 3.1: Text data for semantic matching

List of airports in a country (source airports web service)	List of cities in a country (source weather web service)
AMSTERDAM SCHIPHOL	Amsterdam Airport Schiphol
BERGEN OP ZOOM WONSDRECHT	AntillesFlamingo Airport, Bonaire
BREDA GILZE RIJN	AntillesHato Airport, Curacao
DEN HELDER DE KOOY	AntillesJuliana Airport, Saint Maarten
EINDHOVEN	AntillesRoosevelt Airport Saint
ENSCHEDE TWENTE	Eustatius
GRONINGEN EELDE	Groningen Airport Eelde
LEEUWARDEN	Leeuwarden
LEIDEN VALKENBURG	Maastricht Airport Zuid Limburg
MAASTRICHT	Rotterdam Airport Zestienhoven
ROTTERDAM	De Bilt
SCHIPOL	Deelen
THE HAGUE	De Kooy
UDEN VOLKEL	Eindhoven
UTRECHT SOESTERBRG	Gilze-Rijen
WOENSDRECHT AB	Soesterberg
	Twenthe
	Valkenburg
	Volkel
	Vlieland
	Woensdrecht

The result of word overlap matching (Figure 3.2) shows that only 25% of the names match using direct text matching, and 43.75% of the names can be matched using word overlap matching technique. In this case, we observed that matching results (Table 3.2) with index ≥ 0.25 can be considered as 'valid' for our purpose. On the other hand, when we consider the results of matching using WordNet, we get matching for ENSCHEDE TWENTE, SCHIPOL, and UTRECHT SOESTERBRG, which is certainly wrong and was not identified using word overlap matching algorithm due to the difference in spelling. However, the results 1) BERGEN OP ZOOM WONSDRECHT match with De Kooy with index = 0.625 and 2) ROTTERDAM match with Eindhoven with index = 0.899999 are not 'valid' for our purpose. In terms of semantics, the matching could be correct but it did not solve our problem. This shows that for matching short text based input-output data, usage of lexicon database may not necessarily improve the results of the semantic mediation. This could be mainly because a short text provides very limited contextual information.

List of all airports

Data that match directly : 25 %

Data that match using word overlap matching : 43.75 %

Direct + word overlap matching : 68.75 %

Figure 3.2: Screen shoot of output of direct semantic matching for Netherlands

Table 3.2: Results of semantic matching using word overlaps algorithm and using WordNet

Airport name	**Word overlaps matching**		**WordNet based matching**	
	Matching city	**Matching index**	**Matching city**	**Matching index**
AMSTERDAM SCHIPHOL	Amsterdam Airport Schiphol	0.6666666	Amsterdam Airport Schiphol	0.863999
BERGEN OP ZOOM WONSDRECHT	-	0	De Kooy	0.625
BREDA GILZE RIJN	Gilze-Rijen	0.25	Gilze-Rijen	0.759999
DEN HELDER DE KOOY	De Kooy	0.5	De Kooy	0.966666
EINDHOVEN	Eindhoven	1	Eindhoven	1
ENSCHEDE TWENTE	-	0	Twenthe	0.6833336
GRONINGEN EELDE	Groningen Airport Eelde	0.66666666	Groningen Airport Eelde	0.8000000
LEEUWARDEN	Leeuwarden	1	Leeuwarden	1
LEIDEN VALKENBURG	Valkenburg	0.5	Valkenburg	0.7333333
MAASTRICHT	Maastricht Airport Zuid Limburg	0.25	Maastricht Airport Zuid Limburg	0.5
ROTTERDAM	Rotterdam Airport Zestienhoven	0.33333333	Eindhoven	0.8999999
SCHIPOL	-	0	Amsterdam Airport Schiphol	0.5124999
THE HAGUE	-	0	AntillesHato Airport, Curaçao	0.2119999
UDEN VOLKEL	Volkel	0.5	Volkel	0.7233332
UTRECHT SOESTERBRG	-	0	Soesterberg	0.6599999
WOENSDRECHT AB	Woensdrecht	0.5	Woensdrecht	0.6666666

Semantic matching of variable names of models

Case study 2: The second case study was to match variable names of components in the CSDMS model repository. The objective is: given a component (data-receiving component) that requires external input data find other possible components (data-providing components) that could provide input data for it. The CSDMS uses standard names (Peckham, 2014) to document component attributes, and we considered only models which are documented following the standard names. As a first step, we put all input variable names of components in one category and output variable names in another category. Then, we applied both word overlap matching and WordNet

based matching algorithms to find components that could provide data to a given component. In this case, we applied the WordNet based algorithm to investigate to what extent standardization will minimize the need for complex semantic processing.

The results of the semantic matching (Table 3.3) show that for standardized variable names the usage of lexicon-based semantic matching does not provide better results than the simplistic direct text matching, which clearly indicates the benefits of standardized names when searching for relevant components. In this specific case, we observed that, although lexicon-based semantic matching index values are close to 1 the texts we are comparing represent different concepts. For example, variables channel_inflow_end_water__ discharge and sea_water_surface_wave__period return matching index of 0.904. These matching values are misleading to perform automated mediation.

Table 3.3: Semantic matching in the CSDMS repository for standardized model variable names (a) using direct matching algorithm, (b) using WordNet based matching algorithm.

Data-receiving component -> Avulsion. Requires input data for variables:	Using Direct matching algorithm		
	Data-providing component	Output variable name	Matching index
channel_inflow_end_bed_load_sediment__mass_flow_rate	river	channel_inflow_end_bed_load_sediment__mass_flow_rate	1
channel_inflow_end_water__discharge	river	channel_inflow_end_water__discharge	1
surface__elevation	cem	surface__elevation	1
	child	surface__elevation	1
	sedflux2d	surface__elevation	1

(a)

Data-receiving component –> Avulsion. Requires input data for variables:	Using WordNet based matching algorithm		
	Data-providing component	Output variable name	Matching index
channel_inflow_end_bed_load_sediment__mass_flow_rate	child	bed_load__mass_flow_rate	0.9228572
	river	channel_inflow_end_bed_load_sediment__mass_flow_rate	1
channel_inflow_end_water__discharge	child	channel_water__discharge	0.90875
	river	channel_inflow_end_water__discharge	1
	waves	sea_water_surface_wave__period	0.904
	windwaves	sea_surface_water_wave__period	0.904
surface__elevation	cem	surface__elevation	1
	child	surface__elevation	1
	sedflux2d	bedrock_surface__elevation	0.924
	sedflux2d	surface__elevation	1

(b)

We also considered that instead of doing the semantic matching within processed data, i.e. within specific list of input and output variable names, what if the search for data-providing components is done on the full metadata of the components. Note that the CSDMS provides an API that returns standardized metadata of components in JSON[33] (JavaScript Object Notation) format. The metadata information from the APIs consists of data beyond the list of input-output variable names of components. Due to this, we need a functionality to extract the list of input and output variable names from the full metadata of the component. To realize the search for components on the full metadata we incorporated the following algorithm:

- Read the whole metadata of the data-receiving component,
- Read the whole metadata of all other components in the repository,
- Extract input variable names of the data-receiving component from its metadata information,
- Extract list of output variable names of all components from the metadata information,
- Do semantic matching,
- List components (together with the corresponding output variable names) that can possibly provide data to the data-receiving component.

The result of searching for components from the whole metadata (Figure 3.3) shows that using the semantic matching algorithms we can search for other components that could provide data for a given component. In this case, we also see that applying lexicon database does not improve the matching of standardized variable names, which confirms the 'Law of the Semantic Web' by Uschold (2003) – "the more agreement there is, the less it is necessary to have machine-processable semantics".

[33] JSON - http://www.json.org/

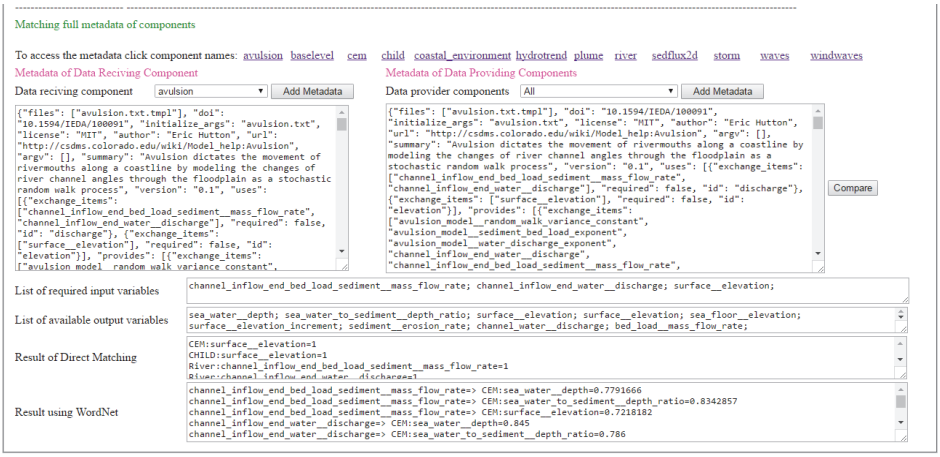


Figure 3.3: Screen shot of the user interface for searching for components that could provide data to a certain component from the CSDMS standardized metadata of components.

In general, the results of the semantic matching algorithms indicate that although we do not fully automate the semantic mediation process we can still provide useful information that could facilitate user guided semantic mediation.

3.5.3 Unit conversion in integrating models

Unit conversion is another semantic mediation task that we considered in DMIF. As part of our design criteria - avoid reinventing whenever reuse of resources is possible – we found that QUDT[34] or Quantities, Units, Dimensions and Data Types Ontologies is a good resource for unit conversion in semantic mediation. As indicated in its website, QUDT, developed by TopQuadrant[35] and NASA,[36] has the objective of improving the quality of software interfaces, web services, and data interoperability by providing consistent terms and constructs in defining attributes of datasets and messages. The QUDT ontology focuses on quantities, units, dimensions, data types, enumerations, and structures. Besides, QUDT ontology is freely available under Creative Commons Attribution-Share Alike 3.0 United States License[37]. The QUDT ontology is available in both OWL and TURTLE formats (Yu, 2011) and these standardized data structures help to build generic semantic mediation functionalities.

[34] QUDT - http://www.qudt.org/
[35] http://www.topquadrant.com/
[36] http://www.nasa.gov/
[37] https://creativecommons.org/licenses/by-sa/3.0/us/

The Units ontology is organized into categories of units, for example, SIUnit, SIDerivedUnit, DerivedUnit, NotUsedWithSIUnit, NonSIUnit. These categories are further organized into types, such as AngelUnit, ForceUnit, CurrencyUnit, MolarEnergyUnit, TimeUnit, etc. Besides, each unit has a number of attributes that will help to perform semantic translations. For example, the unit kilometerPerHour has attributes: Label = 'Kilometer per Hour', Abbreviation = 'km/hr', Symbol = 'km/hr', Type = 'LinearVelocityUnit', ConversionMultiplier = '0.2777777777777778e0'.

For a given unit, the semantic mediation module queries the ontology and provides a list of possible matching units in which a given unit can be translated. Consider Figure 3.4 for graphical demonstration of how we handled unit conversion using QUDT ontology. From the list of available units, assume that the first model is using kilometerPerHour to express a variable. Then, the semantic mediation module will search in QUDT for the Type attribute of the selected unit; in this specific case, kilometerPerHour belongs to the type LinearVelocityUnit. By using the fetched value of Type attribute (i.e. LinearVelocityUnit), the semantic mediation module will infer and provide a list of available LinearVelocityUnit type of units, for example, CentimeterPerSecond, FootPerHour, Knot, MeterPerSecond. Let us assume that the second model uses FootPerMinute to express linear velocity, which has conversion multiplier of 0.00508e0. Then the semantic mediation module will automatically derive the conversionMultiplier from km/hr to ft/min by dividing the conversionMultiplier of the first model by the conversionMultiplier of the second model.

The semantic module uses dotNetRDF[38] API to query the QUDT ontology. The API is an Open Source .Net Library[39] that enables to work with RDF, SPARQL and the Semantic Web. The API provides full support for SPARQL 1.1 Query and Update together with inferencing capability. Due to these features, it is possible to extend the semantic mediation module by adding a number of SPARQL queries.

[38] http://dotnetrdf.org/
[39] http://www.w3.org/2001/sw/wiki/DotNetRDF

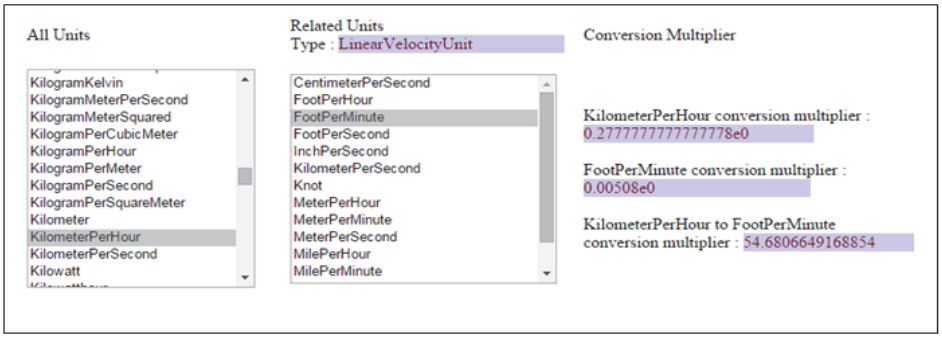


Figure 3.4: Graphic representation of semantic mediation in converting units.

3.6 Runtime integration of models

We define runtime integration as an integration method in which users can select and integrate models using a graphical user interface, during the time of usage. This can be achieved by following a two steps process: first, the models should be made available based on some standard, and second, there should be a 'generic' user interface that enables to link models and run simulations. Note that the 'generic' interface is built by assuming the defined standard. Based on this, we developed a prototype of a 'generic' interface that provides: 1) Runtime access interface (Figure 3.5) - to run stand-alone web service based models, and 2) Runtime integration interface (Figure 3.6) - to integrate web service based models without the need for additional design, coding, debugging, etc. To run stand-alone web service based models, the user needs to provide a URL of the service description (WSDL[40]), then the system will fetch the properties of the underlying service and it will generate appropriate GUI controls to receive input data. By using this interface, we can access different web service based models regardless of their underlying software and hardware platforms, and their location. For example, we can fetch models from UV Index and Alert forecast service on EPA[41], Global weather prediction services[42], U.S. neighborhood level demographic services[43], etc. The models may return simple or complex datasets and the output will be displayed in the user interface.

[40] WSDL - https://www.w3.org/TR/wsdl
[41] EPA UV alert service - https://www.epa.gov/enviro/web-services#hourlyzip
[42] http://www.webservicex.net/ws/default.aspx
[43] http://ws.cdyne.com/DemographixWS/DemographixQuery.asmx

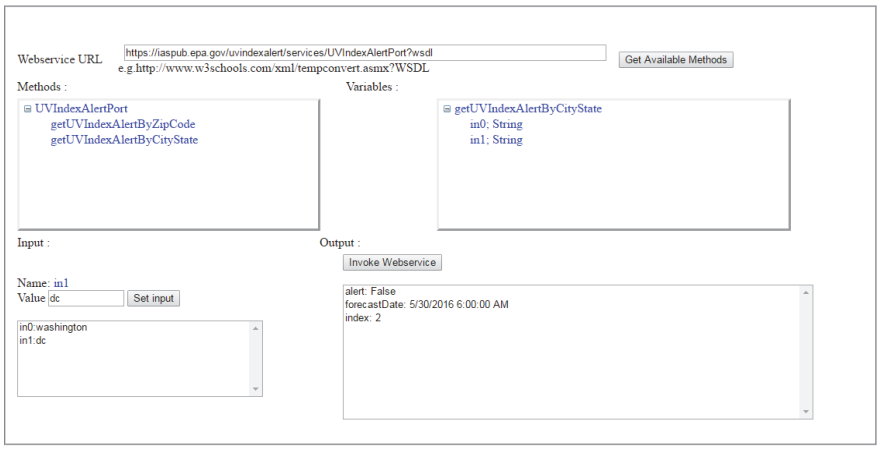


Figure 3.5: User interface for runtime access of web service based models, the case of EPA's UV alert web service.

The runtime integration interface has the following main parts (Figure 3.6):

1) The first part allows to provide the URL of the WSDL of web services to be linked. The user should provide the URLs, and then the available functions or methods of the services will be fetched by the system. Note that the participating services can be models or data sets organized in various ways, for example, sensor observation data (Regueiro et al., 2015). Individual web services can have one or more functions and the user has to select which functions are to be involved in the integration. Besides, the user can also provide input data or scenarios if there is any input that the model requires.

2) The second part provides access to a data conversion service if needed. If the output of one service can go directly as an input for the second service, then the user will skip this part. However, if data conversion is required and if the data conversion functionality is available as web service, then the user should provide the URL of the data conversion service. If data conversion service is not available, skilled users can build data conversion services and wider user community can reuse it.

3) The third part is the integration builder. In this part, the user should specify the workflow and input-output data exchange pattern between the services. The integration builder has two list or combo boxes that will be automatically populated when the user performs the operations described in steps (1) and (2). The first combo box is called *Output functions*, i.e. to indicate list of functions that produce output, and it maintains the concatenated value of service name with semicolon and then with the method name. For example, a service named `servicex` with a method called `methodx` will be listed as `servicex:methodx`. If

servicex has also an additional method called methody then this method will be listed as servicex:methody. The second combo box, called *Input variables*, maintains a list of input variables together with the corresponding method and service names. It contains a concatenated value of the service name, method name and input variable names. For example, if methodx of servicex have input variables varn and varm, then it will be maintained as two entries servicex:methodx:varn and servicex:methodx:varm. Then, the user can define the data exchange pattern between the services by matching the values listed in the combo boxes.

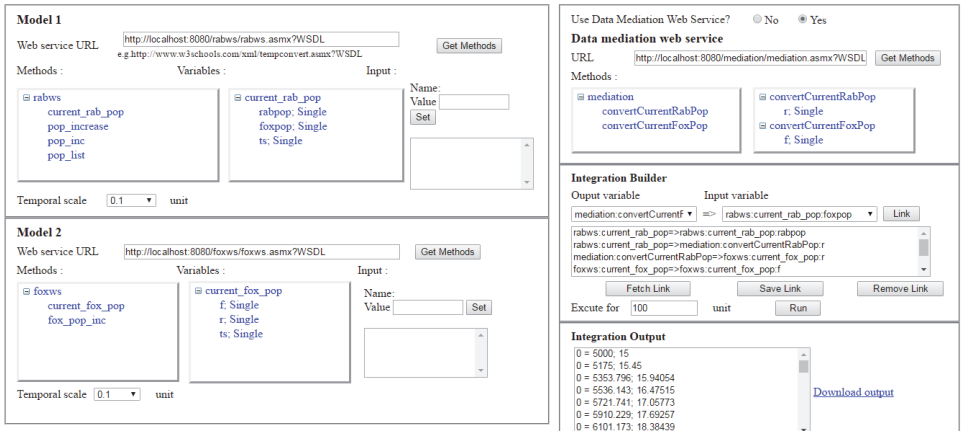


Figure 3.6: User interface for runtime linking of web service based models.

To illustrate how a user can define the linking between models in the GUI, consider the following example. Assume that we are integrating model M1 with model M2 to simulate a certain scenario. For example, M1 is predator's population dynamics model and M2 is the prey's population dynamics model so that the integration of M1 and M2 will simulate the classic predator-prey dynamics. Suppose M1 has a function named function1 with input variables var1 and var2, and after execution it produces the value of var1 (i.e. var1') as an output. Similarly, M2 has input variables var3 and var4, and after execution it produces the value of var4 (i.e. var4') as an output. Besides, suppose that var1 and var3 represent the same variable, except it is named differently, and the same applies for var2 and var4. As we see in Figure 3.7(a), when the integrated model runs for more than one cycle the outputs of M1 will be used as input for M1 and M2; and similarly the output of M2 will be used as input for M1 and M2. The user can define the linking by selecting the data-providing item from Output functions combo box and data-receiving item from the Input variables combo box and then click the 'Link' button, and this will generate data exchange commands as follows:

```
M1: function1 => M1: function1:var1
M1: function1 => M2: function2:var3
M2: function2 => M2: function2:var4
M2: function2 => M1: function1:var2
```

The listing represents data exchange pattern shown in Figure 3.7(a). The first line in the expression is to mean 'output produced from `function1` of model `M1` will be used as input by variable `var1` of `function1` of model `M1`'. Similarly, the second line is to mean 'output produced from `function1` of model `M1` will be used as input by variable `var3` of `function2` of model `M2`'. Once the linking is defined, the user can set the number of cycles the models should run, and then run the models. In every simulation cycle, the system executes all the linkages defined by the user.

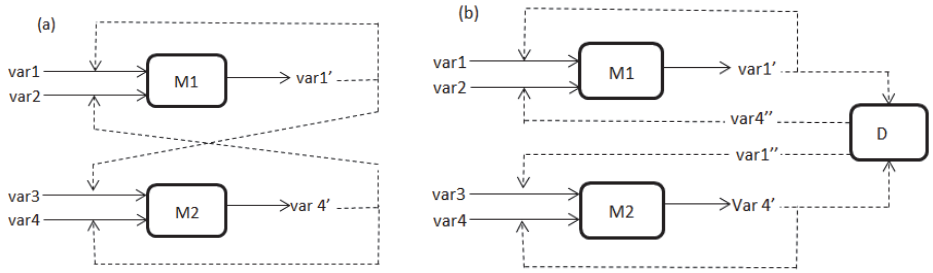


Figure 3.7: Variable mapping in integrating model M1 with model M2. M1 takes inputs var1 and var2 and produces output var1'. M2 takes inputs var3 and var4 and produces output var4'. (a) When Var1 and Var3 are the same except they are named differently. The same applies for `var2` and `var4` and the models are linked directly; (b) When data conversion is required. That is model `D` has a function that converts `var1'` to `var1''` and another function that converts `var4'` to `var4''`.

However, in most of the cases during integration the output of one model may require a conversion to the appropriate format before it is passed to the next model. Consider Figure 3.7(b), where we include a data conversion web service, `D`, which has at least two functions, let us say `function3` and `function4`. The purpose of `function3` is to take output data from `M1`, and convert it so that it can be passed to `M2`. Similarly, `function4` takes output from `M2`, converts it and so that it can be sent to `M1`. The user can define such kind of data exchange pattern from the user interface as follows:

```
M1: function1 => M1: function1:var1
M1: function1 => D: function3:var1
D: function3 => M2: function2:var3
M2: function2 => M2: function2:var4
M2: function2 => D: function4:var4
D: function4 => M1: function1:var2
```

In addition, the user can set the temporal scales for each model and the duration for which the integrated system will run. Based on these inputs all data exchange commands defined by the user will be executed. This kind of interfaces can resolve:

- access right issues since we do not need to acquire models and web service based models can be hosted on the owners' servers,
- the issues related to deployment and configuration of models on end users' computers,
- the issues related to the requirements of writing the code that links the participating models.

In addition, it will favor accessibility and reusability of models, and collaboration among modelers. The interface can be improved further by providing alternative data visualization features like maps, tables, and graphs. The usability will be certainly much improved if the user can select the list of output variables and the format in which the output will be displayed.

3.7 Discussion

Making existing independently developed models interoperable can be difficult (Galler and Turner 2007; Nativi et al., 2013). To contribute towards a solution to this global challenge, first, we have set our design goals for a model integration framework and then we treated interoperability at different levels.

As a proof of concept of our design approach, we implemented it in a case study that aimed to link three models from the COMPLEX[44] project. The first model is GCAM, a dynamic-recursive partial equilibrium model that represents the economy, energy sector and land use. The model is developed using C++ and it uses a number of XML based configuration files. The second model is EXIOMOD; a macroeconomic model developed using GAMS programming language. The third model is NIROO, an agent based model that focus on households' energy use and potential behavioral changes, and it is developed using NetLogo. By linking the three models, we are able to build system-of-systems that could simulate climate-change mitigation policy scenarios (Belete et al, in revision).

The main contribution of our work is that we developed and described the internal details of a distributed model integration framework that could link multidisciplinary heterogeneous models that could be deployed on different hardware and software platforms. Given the URL of a web service based model, it can be incorporated into the DMIF integration framework regardless

[44] COMPLEX - http://owsgip.itc.utwente.nl/projects/complex/

of its platform and location. As a result, a model deployed in a UNIX environment and another model operating under Windows can interoperate by using our design approach. A model developed with stakeholder participation using the user-friendliness of the NetLogo interface can be connected to some sophisticated climatic or economic models. Designing and developing applications that can integrate and run models on the web is a way forward for integrated environmental modeling (Laniak et al., 2013) and we consider our effort as a step in that direction.

In developing wrappers for models, we observed different challenges that require further careful consideration. Many of legacy models are developed as one 'main' function that read its inputs from a file, manipulates several variables, writes its outputs to file, and returns nothing. In such a case, the wrapper should invoke the 'main' function, but the list of input-output variables that the wrapper should implement varies depending on the specific context of the models involved in the coupling. There is also a possibility of the need to develop more than one wrapper for a single model or component. For example, in our case study we observed that, if we let the wrapper to return all model output then it would have its effect on system performance. Due to this, for example in developing wrapper for GCAM, even though GCAM produces several outputs, we designed it to return only two variables. This helped us to improve system performance. However, if we want to link GCAM with another model that requires access to other output variables of GCAM, then it may be better to develop new wrapper function/method than modifying and the existing one. In fact, in developing another wrapper for a model, we can reuse significant amount of code and we need to change only the variables that should be included as the model output.

The other identified challenge we observed in relation to wrapping and synchronizing the communication of models is error and exception handling. Error and exception can arise from the wrappers, mediation module, or from the underlying models. Error and exception that arise from the mediation module can be handled using the usual error and exception handling strategies. However, error and exception generated by the underlying models varies depending on the implementations of the models. Sometimes error generated by a model may seem to originate from the wrapper. Unless we do not have detailed documentation of the models, it is very difficult to list the possible kinds of errors and exceptions that could be generated. We may have access to the source code of the models but it could be very difficult to list errors and exceptions, especially run-time errors. Which kind of errors and exceptions can we tolerate? Which are not tolerable? There is no general rule to manage error and exception generated by models, but we have to decide based on individual cases and this makes the way towards generic integration framework more difficult.

With the advances in web technology, presenting models as web services is becoming more prevailing (Castronova et al., 2013; Geller and Melton, 2008; Goodall et al., 2011). The effort by GEO Model Web Initiative (Nativi et al., 2013) and Apps for the environment[45] by the US Environmental Protection Agency are also exemplary initiatives towards models as services. Nevertheless, discovery and integration of service-based models will be challenging unless the naming of service properties is unambiguously defined (Nativi et al., 2013). The CSDMS Standard Names (Peckham, 2014) is an exemplary step towards standardizing names for semantic mediation. However, providing generic semantic mediation requires going beyond standardizing names, it requires a full-fledged ontology. Again the challenge is to build such an ontology to address semantic interoperability across multidisciplinary models, which is not a simple task.

We demonstrated that the semantic mediation in unit conversion among models can be managed using an openly available ontology. Following this, we are able to avoid the need for custom developed code to convert, say, between SI and derived units. However, models can represent a quantity in a number of different ways, for example, concatenated units that are not included in the ontology. Other freely available ontologies, like Semantic Web for Earth and Environmental Terminology - SWEET[46] ontology that currently contains around 6000 concepts in 200 separate ontologies. These could broaden the use of ontology beyond unit conversion.

To minimize the custom coding and hard-coding in semantic mediation, metadata of model interface such as input-output variable names, units, scales, etc. should be connected to the ontology (Papazoglou, 2008). To realize this we have to use either the terminologies of the ontology in wrapper model interfaces, or we have to incorporate model interface metadata into an existing ontology. The first option requires understanding and consensus among model developers. However, the second option can be done relatively easily without seeking huge collaborations among modeling communities. It requires only designing the appropriate structure to accommodate model interface ontology in the existing ontology, in our specific case to QUDT ontology. For example, a model interface has input and output variables; and each of those input and output variables has name and unit attributes. We can use semantic relationships like 'is-a', 'equivalent', etc. to establish links between names of variables, which are found in different models. Similar relations can be established between the units used by the model and the corresponding available units in QUDT ontology. Once we defined the structure to accommodate model interface ontology then we can register attributes of participating models in the ontology. This can make

[45] http://www.epa.gov/mygreenapps/
[46] https://sweet.jpl.nasa.gov/

variable mapping and unit conversion between models fully automatic. But this is not going to happen until model documentation standards will be elaborated and implemented.

The use of distributed modeling approach enabled us to construct runtime access and integration tools. We showed that the technical aspect of integration could be standardized and developed as a 'generic' utility. The availability of such tools will help to focus on providing automatic data mediation functionalities so that users may not need to build data mediation services. Basically, data mediation requires to identify WHAT, WHERE, and WHEN of data in the context of each model (Moore & Tindall, 2005). The WHAT is a quantity description of the data represented by the variable; WHERE is about location information; and WHEN describes temporal information about data. Sometimes the WHERE and WHEN can be optional depending on the context of use of the data. E.g. if both model 1 and model 2 are referring to the same spatial location and temporal instance then the WHERE and WHEN may be considered as optional to perform data conversion. Due to this, unlike the technical integration, it is difficult to develop a 'generic' module that will handle the data mediation of all incoming models, until appropriate model documentation standards are in place and are adopted by the modeling community. Automatic data mediation functionalities can act on the provided data based on the specific context of the participating models. This may require building artificially intelligent agents that fit such purposes. This requires further enhancement of the semantic mediation features and embedding learning capabilities into the data conversion module. We hope that this will be the future direction of model integration frameworks.

3.8 Conclusion

In the year 2007, The Ecological Forecasting Program at NASA has set the vision for 5-10 years, calling it "The Model Web Concept" (Galler and Turner, 2007). One of its aspirations was to have distributed, networked, and interoperating models, datasets and sensors. In this paper, we present the design of a modular, distributed, scalable, and web-enabled model integration framework that enables linking of a wide variety of models. We also illustrate some of our ideas by implementing a prototype application and by performing a pilot case study. This is hardly a complete solution to all model integration challenges. Our aim is to demonstrate how heterogeneous models can be linked, regardless of their nature and their location.

By offering models as web services and providing an interface that works in a regular browser we make models more accessible and expect them to be more widely used both on their own as stand-alone simulation tools and in

integration with other models, serving various needs of decision support. The fact that to run a model we no longer need to download and install it, and setup all the input-output interfaces, very much simplifies its use and creates opportunities for a wider participatory use of models for learning and decision making (Voinov et al., 2016). This also delivers much needed functionality that can make models available in mobile devices and can facilitate co-learning and co-management with large numbers of stakeholders becoming involved in the process. However, further integration and meaningful use of models will become possible only when appropriate model documentation standards will become available and implemented. Standards that can convey the contextual information of the participating models can help consistent interpretation of the represented concepts. This will also facilitate the provision of reusable and standardized data conversion functionalities by grouping available data formats and by developing algorithms that serve specific contexts. Until then we are always running the risk of misrepresenting the ideas, assumptions, conventions, and intentions that are part of any model definition, and as result misusing models. The involvement of experts in the process of model integration will still be essential for some final checks and balances when deciding how models can be coupled and run.

Chapter 4

Exploring temporal and functional synchronization in integrating models: a sensitivity analysis*

* This chapter is based on:
Belete, G.F., Voinov, A., 2016. Exploring temporal and functional synchronization in integrating models: A sensitivity analysis. Computers & Geosciences 90, 162-171.

Abstract
When integrating independently built models, we may encounter components that describe the same processes or groups of processes using different assumptions and formalizations. The time stepping in component models can also be very different depending upon the temporal resolution chosen. Even if this time stepping is handled outside of the components (as assumed by good practice of component building) the use of inappropriate temporal synchronization can produce either major run-time redundancy or loss of model accuracy. While components may need to be run asynchronously, finding the right times for them to communicate and exchange information becomes a challenge. We are illustrating this by experimenting with a couple of simple component models connected by means of Web services to explore how the timing of their input-output data exchange affects the performance of the overall integrated model. We have also considered how to best communicate information between components that use a different formalism for the same processes. Currently there are no generic recommendations for component synchronization but including sensitivity analysis for temporal and functional synchronization should be recommended as an essential part of integrated modeling.

4.1 Introduction

In integrated modeling we may need to link component models, which are built under different disciplinary paradigms and assumptions, use different temporal and spatial scales, as well as different numeric schemes and methods (Laniak et al., 2013; Peckham et al., 2013). The fact that we are linking and synchronizing potentially very different components, designed to be treated under different spatio-temporal settings, may only add to the uncertainty and variability that can emerge from the integration process itself. The way space and time are treated can be further complicated by the different numeric methods used in components. Using higher order numeric approximations may compensate for some coarser time and space stepping, but may make it more difficult to define appropriate synchronization times and boundaries. Furthermore, components may assume different functional responses when modeling the same processes. Within certain domains these functions may be producing quite similar output, however eventually they can diverge quite significantly only adding to the overall uncertainty of the integration effort.

The investigation of this uncertainty and its impacts on model results can be handled using a kind of sensitivity analysis (Wainwright et al., 2014). In most of the traditional sensitivity studies the focus is on model parameters, including initial conditions (Hamby, 1995). In this research we are specifically looking at sensitivity to model characteristics that are related to integration,

to module coupling procedures. As such we will be analyzing the sensitivity of the integrated model to:

1. Variations in time stepping in components and the timing of their synchronization;
2. Changes in numerical methods used in components;
3. Changes in functional responses assumed in components to describe the same processes.

The experiments are conducted by varying only one characteristic at a time and keeping all other controls the same. The observations reported and conclusions drawn from this research can serve as a starting point to perform further sensitivity analysis in integrating models. Our analysis is largely for demonstration purposes to show what we should expect from model coupling and what are the possible problems that we may run into. We have used a very simple, classical model, which we split into components to see how the output will change depending on how the components are run. While there are some good methods for parametric sensitivity analysis, including global sensitivity treatment (Saltelli et al., 2008), these methods hardly apply in our case when testing sensitivity to how components are organized and coupled. Therefore we had to resort to the trial-and-error type of analysis, simply running the model under different arrangements and reporting the differences observed.

The other reason for doing this analysis is because when modules are linked using data or message exchange approaches, there is always overhead involved. Results from one module have to be collected, packed, sent to another module, unpacked and included in further calculations. This takes time. For example, in the analyses that we present below, the split version of the model runs 13 times slower than when the model is treated as a whole. Clearly we want, when possible, to minimize the interaction between components. When doing that, we want to know what can be gained and what can be lost in terms of accuracy vs. performance.

The paper is organized as follows. Section 4.2 provides description of the models and integration framework used to perform the sensitivity analysis experiments. In section 4.3 three categories of sensitivity analysis experiments with the corresponding observations are presented. Section 4.4 presents discussion followed by conclusions based on the experiments and observations.

4.2 The models and the integration framework

Two individual component models and a model integration framework are used to perform this research. The two components are developed based on

the classic Lotka-Volterra predator-prey model (Volterra, 1926; Lotka, 1956; Voinov, 2008). The original model mathematically is expressed as:

$$dX/dt = aX - V(X)Y \quad \ldots\ldots\ldots\ldots (1)$$

$$dY/dt = cV(X)Y - dY \quad \ldots\ldots\ldots\ldots (2)$$

where X = size (or total biomass) of the prey population; Y = size (or total biomass) of the predator population; a = birth rate or number of offspring per individual per year; V(X) = so called trophic function that describes the hunting strategy of the prey; c = economic coefficient or efficiency of conversion of prey consumed into new predators; d = mortality rate or proportion of predator population dying per year. In the simplest case V(X) = bx, where b = proportion of the prey population consumed by one predator per year.

To convert this model into an integrated, coupled one, we implemented equation (1) as an independent rabbit model and equation (2) as the fox model. In the first model Y is assumed constant and enters as a parameter, in the second model, similarly, X is constant and is a parameter. When the two models are run in concert they periodically exchange information about X and Y using the most recent value of the variable that is calculated in one model and substituting it for the parameter in the other model. For example, as shown in Figure 4.1, if the rabbit model runs with time step 0.4 and fox model runs with time step 0.5, then whenever the time is a multiple of 0.4 the rabbit model will receive the last calculated value of Y from the fox model, and, similarly whenever the step is a multiple of 0.5 the fox model will get the latest reported values of X. Recent values of X and Y are maintained by the model integration framework. When these two components are run with the same time steps, and variables are updated on every time step of the model run, the results are the same as in the original two-variable Lotka-Volterra model solved simultaneously as a system of ordinary differential equations.

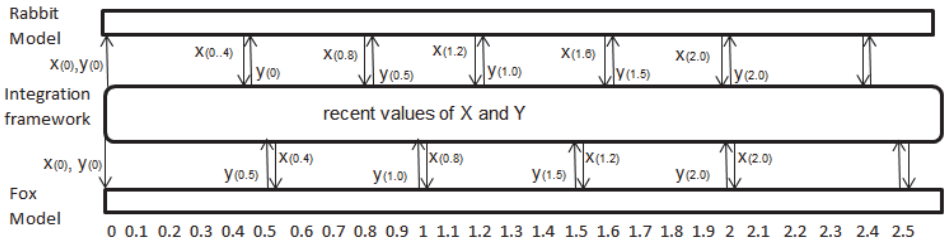


Figure 4.1: Data exchange pattern between component models. Rabbit model runs with time step 0.4 and fox model runs with time step 0.5.

The rabbit population dynamics model was built using C++ and the fox model was programmed using Java. Both models are wrapped using Web services so as to enable message-based communication between them (Figure 4.2).

The web-based model integration framework is built to capture model inputs, to facilitate the communication between them, to manage time steps used, to manage integration types used, and to display the results. Whenever computation by the two models is needed, the input data has to traverse from the integration framework to the Web service wrappers, then to the C++ and Java based implementation of the models, then finally back to the integration framework. The integration framework described above is available at https://github.com/getachewf/mdmif.

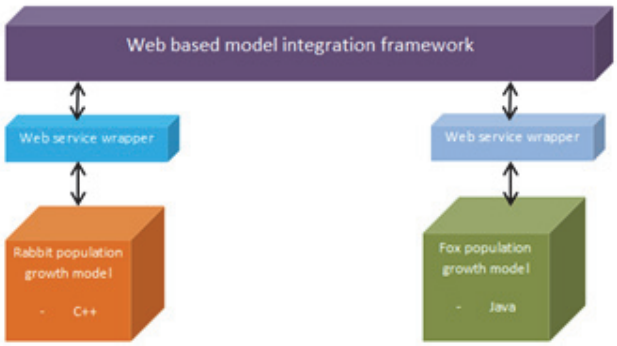


Figure 4.2: Models, Web service wrappers, and integration framework.

As mentioned above, our goal is to study the possible effects of asynchronous and mismatched coupling in a qualitative way, to see what can be potentially expected. In real-life models, which will be certainly of much higher levels of complexity than our simple model, we may be observing other types of behavior. However, even with this simple analysis we can observe some features that are worth mentioning and worth being aware of when coupling model components.

4.3 Temporal and functional sensitivity analysis in integrating models

To conduct a simulation we have to set parameter values for the model equations 1 and 2 described in the previous section. In setting the parameters we have adopted the parameter values used in the Simile® documentation[47], and have chosen:

- birth rate for prey, a=0.5,
- proportion of the prey population consumed by one predator per year, b=0.01,
- conversion coefficient of one prey consumed into new predators, c=0.01, i.e. 100 units of rabbit biomass consumed produces one unit of fox biomass,

[47] http://www.simulistics.com/

- mortality rate for predator, d=0.02

Additionally, for most of simulation runs, we have chosen the following initial values: X_0 = 5,000 and Y_0 = 45.

In performing the sensitivity analysis we followed the simple trial-and-error approach. Our sensitivity experiments were mainly grouped into three sets: (1) classic model, same integration and functional schemes in both components, (2) classic model, but different numerical integration methods in component models, e.g., Euler vs. Runge-Kutta methods, and (3) modified model using different trophic functions in component models (functional desynchronization). The experiments were conducted by selecting different time steps and initial population values for the models.

In addition to visual comparison of model output we have used quantitative measures to record sensitivity. Out of the many different techniques used for characterizing model performance (Bennett et al., 2013) we chose the Mean Absolute Error (MAE) and the R squared (R^2) indices. We selected MAE and R^2 methods since both of them are based on the difference between the base and predicted values at each point of the time series; they enable us to quantify the difference in data patterns between the base and predicted values. MAE measures how close data produced from a model are to the observed values, and is computed as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{i=n} |y_i - y_{bi}|$$

where

n= number of values considered in the computation,
y_i = value of model output for the i^{th} time step,
y_{bi} = value of base (observed) data for i^{th} time step;

Similarly R^2 is used to measure how close two data sets are, and is computed as

$$R^2 = 1 - \frac{\text{sum of squared distance between the actual and predicted values}}{\text{sum of squared distance between the actual values and their mean}}$$

An R^2 value close to 1 indicates that there is a good correlation between the two data sets, whereas a value close to 0 indicates that they are quite different. MAE values range from zero to infinity, making this index especially useful for comparison of data sets which are highly correlated ($R^2 \sim 1$) but quantitatively different.

The observations from the experiments are summarized in the following three subsections. We have used the following notations in the discussion and graphs: ts=time step; r_ts= rabbit model time step; f_ts=fox model time step; r_init_pop=rabbit initial population; f_init_pop=fox initial population.

4.3.1 Sensitivity to asynchronous time stepping

The experiments under this section were conducted to answer the following question: in linking models with different time steps how sensitive is the integration output to the difference in the time steps in component models and to the frequency of information exchange between the modules (the coupling frequency)? To investigate this we made several model runs with different combinations of time steps used in components. As it could be expected, with larger time steps in the component models the irregularities in model output also increased. Bigger time steps generally tend to crash the model faster. However, employing a smaller time step even in one of the models does have a stabilizing effect. Consider the following cases: (1) Figure 4.3(b) where r_ts=0.001 and f_ts=3, (2) Figure 4.3(c) where r_ts=0.1 and f_ts=0.01, and (3) Figure 4.3(d) where r_ts=2 and f_ts=3. The base trajectory for this experiment is the output of the experiment with r_ts=f_ts=0.001 in Figure 4.3(a), which produced the same results as the original two-variable Lotka-Volterra model run with ts=0.001.

The difference in time steps in two component models can be computed as $\Delta t = |r\_ts - f\_ts|$. For the three cases mentioned before, the differences in time steps can be compared as $|\Delta t_1| > |\Delta t_2| > |\Delta t_3|$. The graph of the base trajectory and the graphs in case (1) follow a similar pattern even after 1,000 time steps, but the graphs in (2) and in (3) go to zero after time = 708.3 and 36, respectively. This indicates that (1) smaller differences between time steps of participating models do not guarantee better accuracy in the overall performance; and (2) it is not the magnitude of the difference between time steps that bring irregularities in the output, but the actual size of the time steps used by the models that cause the change in behavior. So, in some cases, running one model with a very small time step while the other model uses a large one does not really help. To demonstrate the effect of using smaller time steps in one of the models consider the scenarios shown in Table 4.1.

Table 4.1: MAE and R2 based comparison of different scenarios where (a) the base scenario is r_ts = f_ts = 0.001; (b) base scenarios are different and case specific. All scenarios use r_init_pop=5,000 and f_init_pop=45 and the model run until t=50 units.

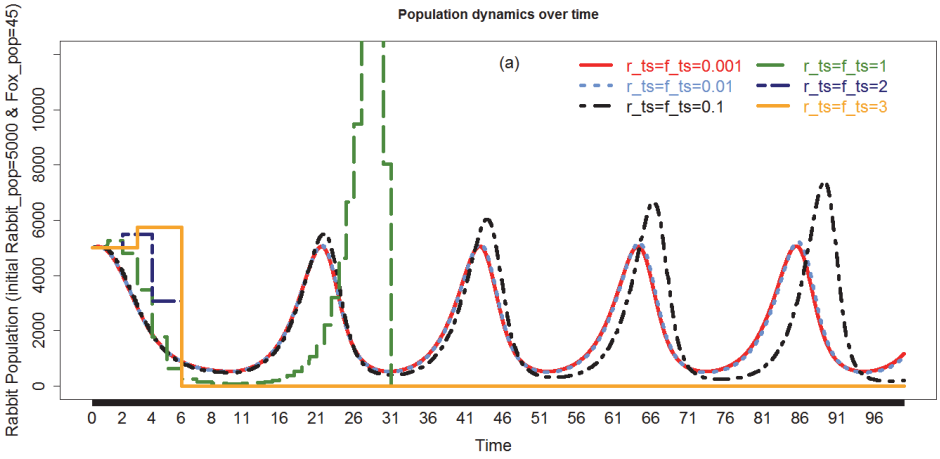
	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
r_ts	0.001	0.1	0.001	0.5	0.001	1
f_ts	0.1	0.1	0.5	0.5	1	1
MAE	17.71	268.90	85.12	1393.96	158.91	model crash
R^2	0.9997	0.9467	0.9954	0.2281	0.9857	model crash

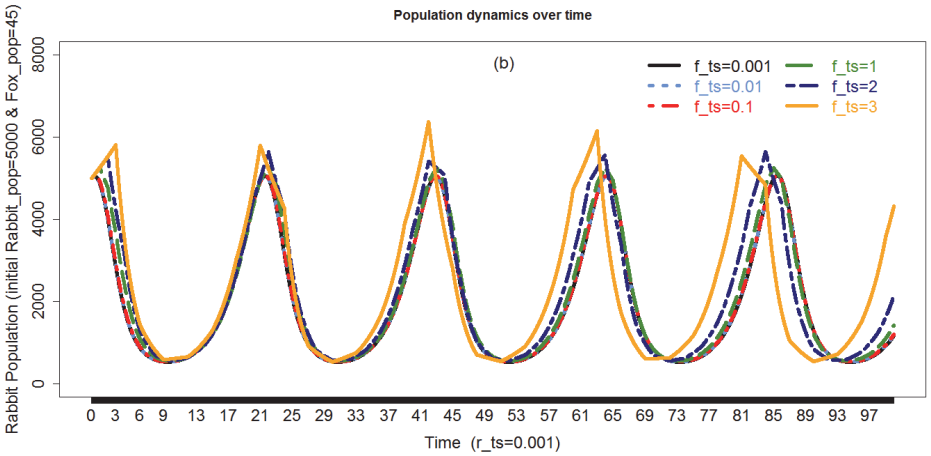
(a)

	Scenario 7	Scenario 8	Scenario 9
r_ts	0.001	0.001	0.001
f_ts	0.1	0.5	1
Base scenario for MAE computation	r_ts=f_ts= 0.1	r_ts=f_ts= 0.5	r_ts=f_ts= 1
MAE	261.27	1372.59	The base model crash
R^2	0.9495	0.2571	The base model crash

(b)

For the scenarios listed in the table the MAE values indicate that usage of smaller time steps in one of the coupled models improves the accuracy significantly. We can also observe that scenario 3 has better accuracy than scenario 2. Consider the MAE values of scenario 1 in (a) and scenario 7 in (b): clearly the scenario where r_ts=0.001 and f_ts=0.1 is closer to r_ts=f_ts=0.001 than to r_ts=f_ts=0.1. The same applies to scenarios 3 and 5. This shows that increasing frequency of information exchange between models has significant contribution for getting better accuracy. However, this comes at a price of longer model runs; e.g. scenario 1 took approximately 61.85 times longer than scenario 2.


Population dynamics over time
(a)
r_ts=f_ts=0.001
r_ts=f_ts=0.01
r_ts=f_ts=0.1
r_ts=f_ts=1
r_ts=f_ts=2
r_ts=f_ts=3
Rabbit Population (initial Rabbit_pop=5000 & Fox_pop=45)
0 2000 4000 6000 8000 10000
0 2 4 6 8 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96
Time


Population dynamics over time
(b)
f_ts=0.001
f_ts=0.01
f_ts=0.1
f_ts=1
f_ts=2
f_ts=3
Rabbit Population (initial Rabbit_pop=5000 & Fox_pop=45)
0 2000 4000 6000 8000
0 3 6 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Time (r_ts=0.001)

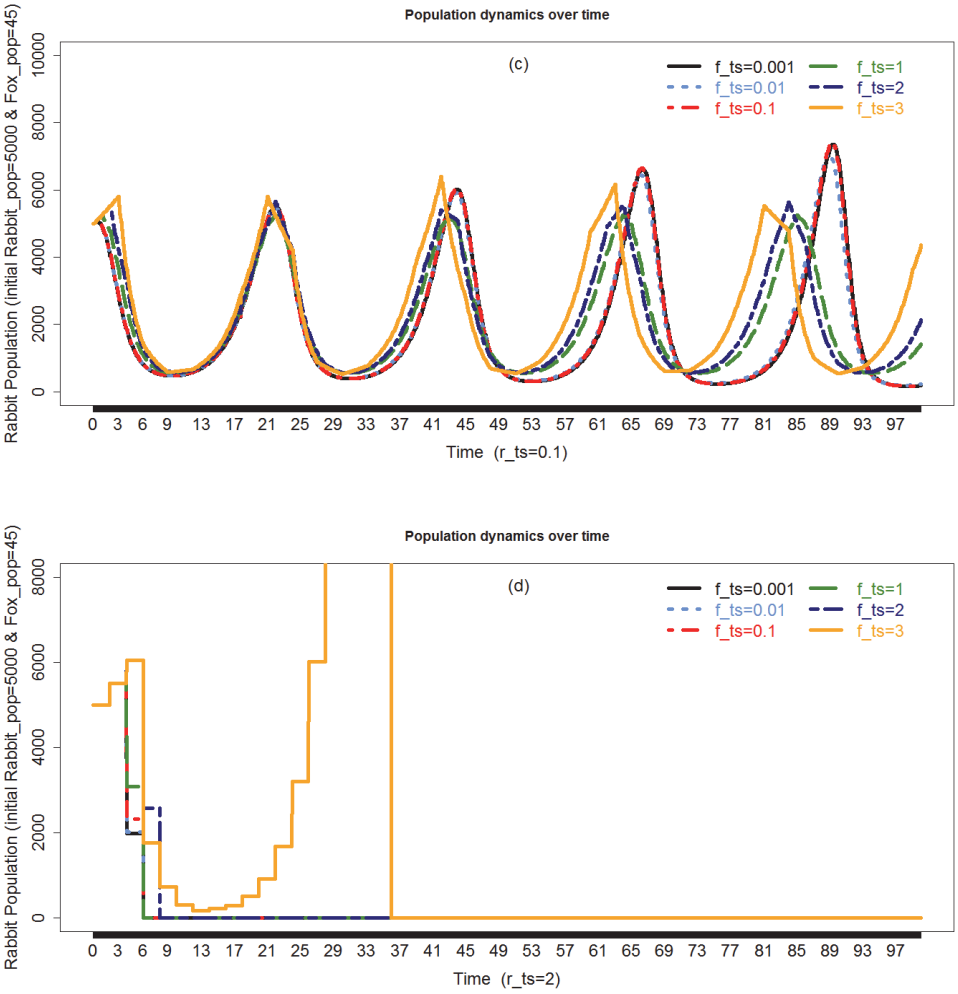


Figure 4.3: Rabbit (prey) population dynamics when (a) r_ts=f_ts; (b) r_ts=0.001; (c) r_ts=0.1; (d) r_ts=2. In all cases initial population of rabbit is 5,000 and initial population of fox (predator) is 45. The base trajectory for this experiment is the scenario with r_ts=f_ts=0.001.

The other issue that we investigated here, when using different time steps for the coupled models, is whether our decision to use a smaller time step for model A and a bigger time step for model B, or vice versa, has any effect on the results? Do the two model components (which look quite similar in terms of the equations used) have a similar effect on the overall accuracy of the simulation when they are run at finer time steps? As follows from Figure 4.4 we do see a difference: the rabbit population dynamics in case (a) and the fox population dynamics in case (b) when r_ts=0.1 and f_ts=1 have more or less regular patterns. However, the model crashes if we decide to swap the time steps between the components and make r_ts=1 and f_ts=0.1. This

means that choosing a bigger time step for model A and a smaller time step for B is not the same as vice versa. It appears that this decision is not symmetrical. This can probably be explained by the fact that the fox numbers are two orders of magnitude lower than the rabbit numbers. A larger time step in a model with larger sizes of the variables is more likely to cause the model to crash. The lesson is that time steps should be chosen taking into account the rates of change that are calculated in the component models: the models with larger variables and higher rates deserve smaller time steps.

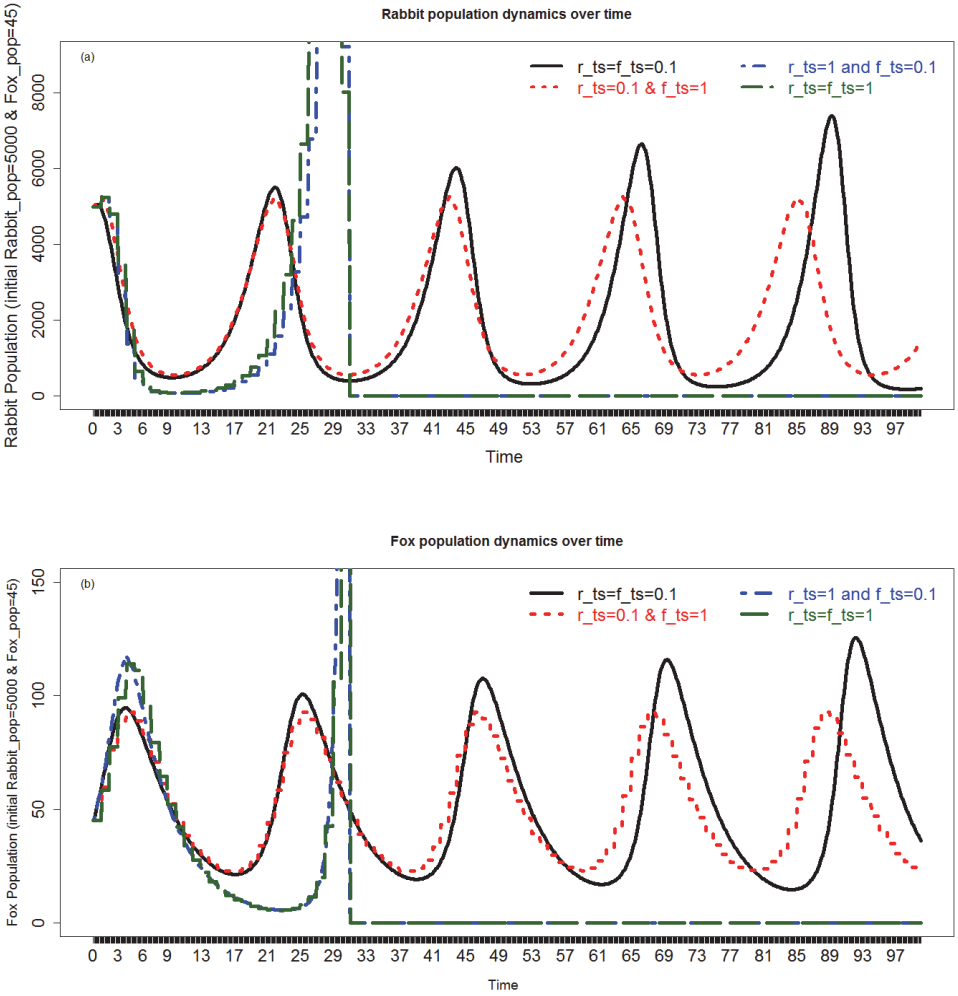


Figure 4.4: (a) rabbit population dynamics; (b) fox population dynamics, assuming initial population of rabbit is 5,000 and initial population of fox is 45.

Generally we see that integration of models with different time steps can be (1) highly sensitive to the size of the time steps chosen; (2) quite sensitive to

the size of the time steps chosen in particular component models, and (3) relatively less sensitive to the difference between the time steps in component models.

4.3.2 Sensitivity to numerical methods in component models

The experiments under this section were conducted to determine the sensitivity of integration output to different integration methods in coupled models. As we know, higher order numerical methods are more accurate, but require additional computation time; however, less computation time is needed than if we try to achieve same accuracy by only decreasing the time steps. Consider a comparison experiment. In the first case, we set the time step to 1 and use Euler integration for both models, running the simulation for 100 time steps. In the second case, to get better accuracy, we change the time step to 0.1, still using Euler integration for both models. In the third case we kept the time step of 0.1 but use the Runge-Kutta method for both models. We observe that the second and the third cases, respectively, took 9.76 and 59.37 times more run time than the first case. However, we need both better accuracy and better performance. Would it help if we use a more accurate method in only one module, while the other module is run with a lower order method? Can the more accurate calculations in one module help to correct and improve the accuracy in the other module?

Here we implemented the Euler method in one model and Runge-Kutta method in the other. Experiments were conducted for three cases: first both models used Euler method, same as in the base run. In the second case the rabbit model was solved with Euler method and the fox model used the Runge-Kutta method. In the third case, vice versa, the rabbit model used the Runge-Kutta method and fox model the Euler method. The results from these experiments are shown in Table 4.2.

Table 4.2: Effect of using Euler method in one model and Runge-Kutta method in the other model. We have used r_init_pop=5,000, f_init_pop=45, and the models were run for 1,000 time steps. The results are compared to the original classic model with Runge-Kutta method run at time step 0.1. Rr stands for Runge-Kutta method in rabbit model, Fr is Runge-Kutta method in fox model, Re - Euler method in rabbit model, and Fe - Euler method in fox model.

	r_ts = f_ts	Re-Fe		Re-Fr		Rr-Fe	
		MAE	R^2	MAE	R^2	MAE	R^2
Scenario 1	0.1	695.74	0.60	370.50	0.87	246.55	0.94
Scenario 2	0.3	1892.81	0.012	1544.56	0.06	1003.94	0.34
Scenario 3	0.7	model crash		1981.56	0.0037	1882.37	0.016
Scenario 4	1	model crash		model crash		2517.03	0.0005
Scenario 5	1.5	model crash		model crash		model crash	

We can see that usage of Runge-Kutta method in even one of the coupled models can improve overall accuracy. However this effect quickly deteriorates if the time step is increased. Usage of different methods in component models is not symmetrical: we get different accuracy when interchanging the methods. In the situation where the usage of bigger time steps with Euler integration crashes, using the Runge-Kutta method even in one of the models can extend the model run for a longer time, e.g., consider scenario 3 and 4 in Table 4.2.

4.3.3 Sensitivity to functional responses

Our objective in this subsection is to investigate the sensitivity of coupling models when the two participating models use different mathematical expressions to describe the same processes. This can easily be the case when integrating real models, which have been developed at different times by different teams, using different assumptions and formalizations for the same processes represented. For example, in our simple model both modules use a linear trophic function (Svirezhev and Logofet, 1983) to describe the interaction between two species, $V = V(x) = bx$, which assumes that there is a linear relationship between the number of rabbits and the rate of predation: the more rabbits there are, the more will be eaten. In many real-life situations this is not the best approximation because, for example, there is a certain saturation level for foxes' appetites, after which they simply cannot

continue consuming even when there are more rabbits. For this reason, the classic Lotka-Volterra model can be modified to use more realistic trophic functions. What will be the behavior of the integrated model if one module uses one (say linear) trophic function, while the other model uses another (say, s-shaped) trophic function (Arditi and Ginzburg, 1989; Voinov, 2008).

Let us assume that now in one of the models we use a Holling type III function $V(x)=ax^2/(h^2+x^2)$. For relatively small values of x both trophic functions, the s-shaped and the linear ones, can be approximately identical. The two functions will have equal values when x=h, which gives us b=a/2h. On the other hand, we have already set that b=0.01, so for example we can let a=100, and then h=5000.

Again, our goal is to see, if through coupling, one model can 'correct' the other when they are exchanging information. For ease of comparison let us call the rabbit and fox components that use the linear trophic function, Rl and Fl, respectively, and, similarly, the models that use the s-shaped function, Rs and Fs. As shown in Figure 4.5 for the parameters chosen above the difference between the linear and s-shaped formalization is quite small when x < 8,000, but then starts to increase. We may expect that models with different trophic functions should give approximately similar results, at least within the range where the gap between the two functions is small. Beyond this range we would like to see how one model will be 'correcting' the results due to the information exchange between the two models.

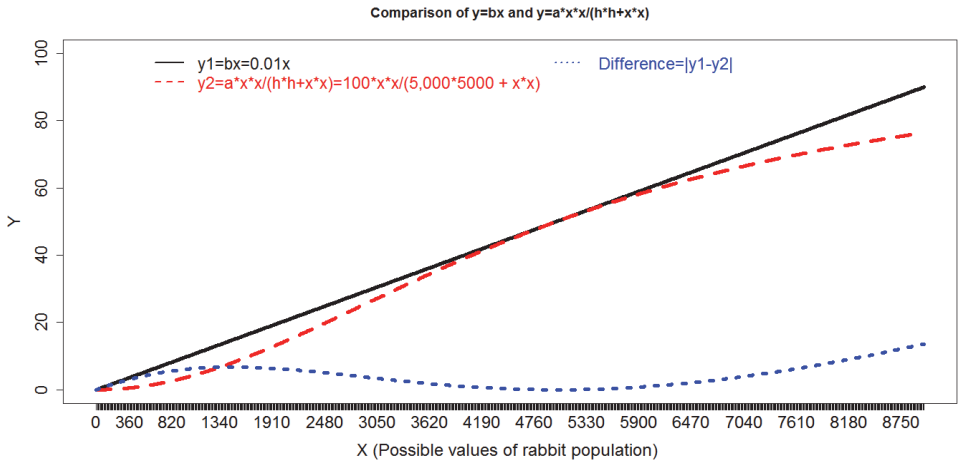


Figure 4.5: Comparison of trophic functions y = bx = 0.01x and $y = ax^2/(h^2+x^2) = 100x^2/(5{,}000^2+x^2)$.

As known from theory, unlike the linear trophic function case, which produces oscillations in populations of predator and prey, the s-shaped trophic function

results in the population equilibrating after a few cycles (Svirezhev, Logofet, 1983; Voinov, 2008). This is also what we can see in Figure 4.6(a) and (b) where graph (1) represents the dynamics in the Rl - Fl type of coupling, and graph (2) shows the results from the Rs - Fs coupling. We will use these graphs as base runs for further comparisons. If we consider a mix of trophic functions in component models we find that the Rl - Fs coupling (graph 3) produces a trajectory qualitatively similar to graph (1). On the other hand the Rs - Fl coupling (graph 4) appears similar to graph (2).

The phase portrait diagram for both the rabbit and fox populations are shown in Figure 4.6(c). As expected, we get a spiral instead of an oval that comes from the analytical solution, because of the integration error produced by the Euler method we used. Here we can say that even within the range of Rabbit numbers, where the two trophic functions are close enough (Figure 4.5), the difference apparently is large enough to produce a qualitatively different behavior.

We see that one of the trophic functions plays a dominant role in determining the trajectory, which is communicated through some 'correction' during information exchange between the two components. It also appears that the trophic function that is used for the Rabbit population is the one that determines the overall performance of the linked model. We can observe that in Rl-Fs type of linking the trajectory is similar to Rl-Fl, and in Rs-Fl type of linking the trajectory is similar to Rs-Fs. Why is that and will this be true in other cases?

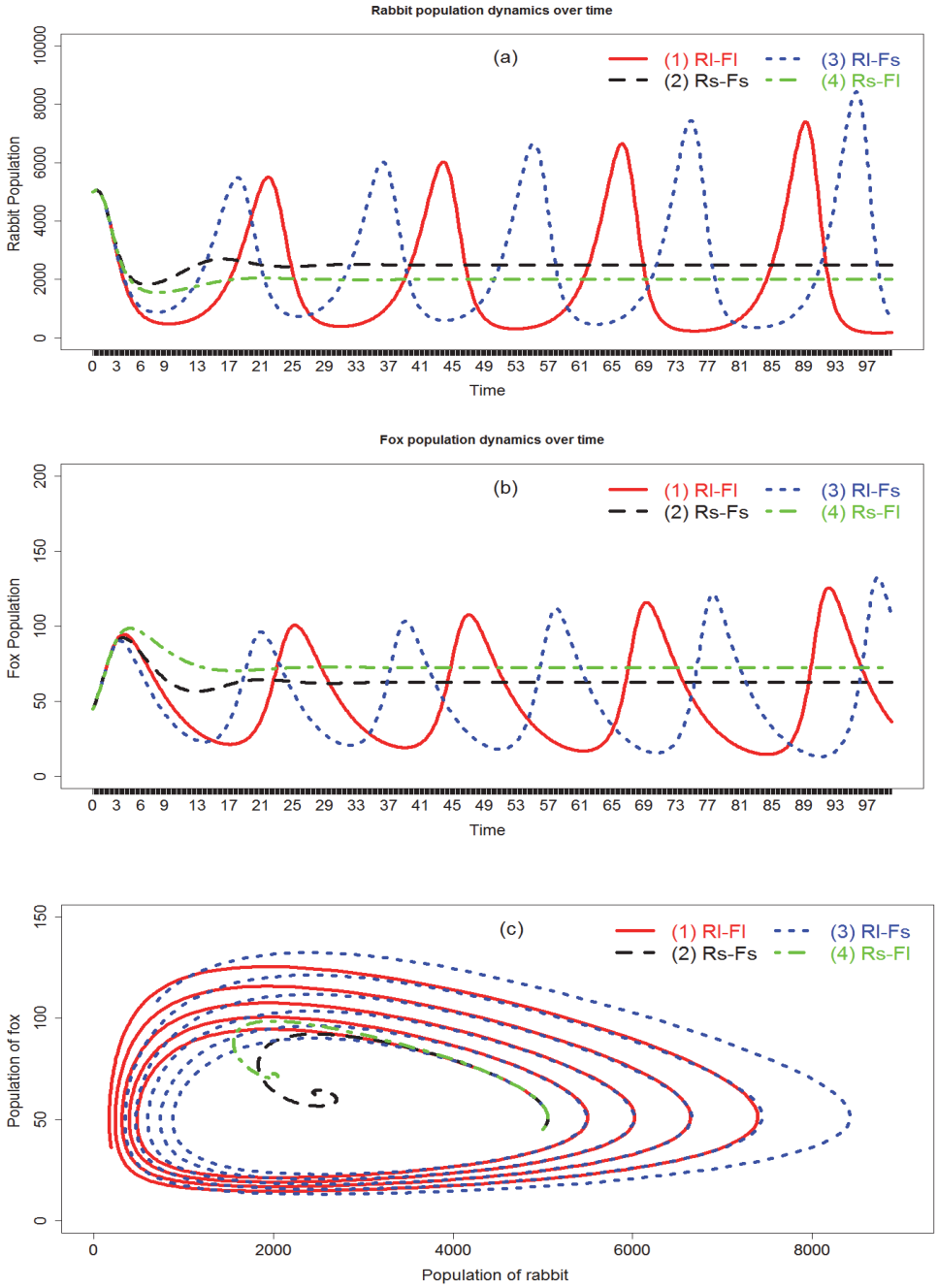
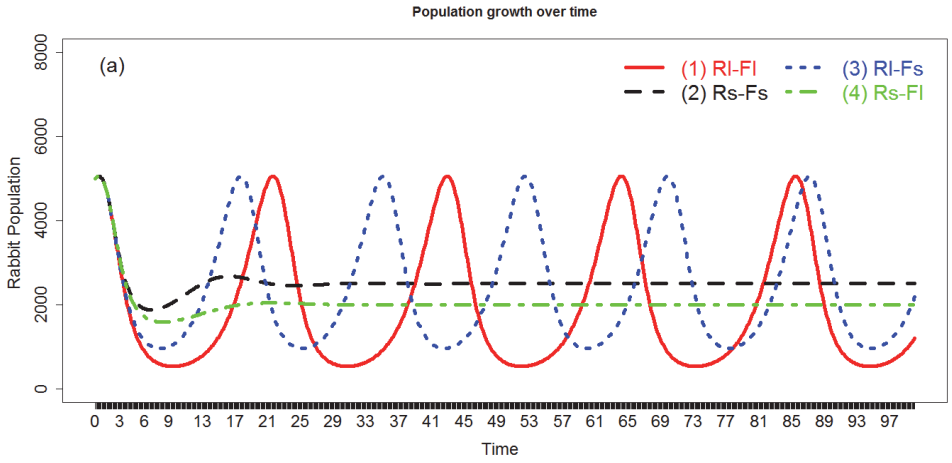


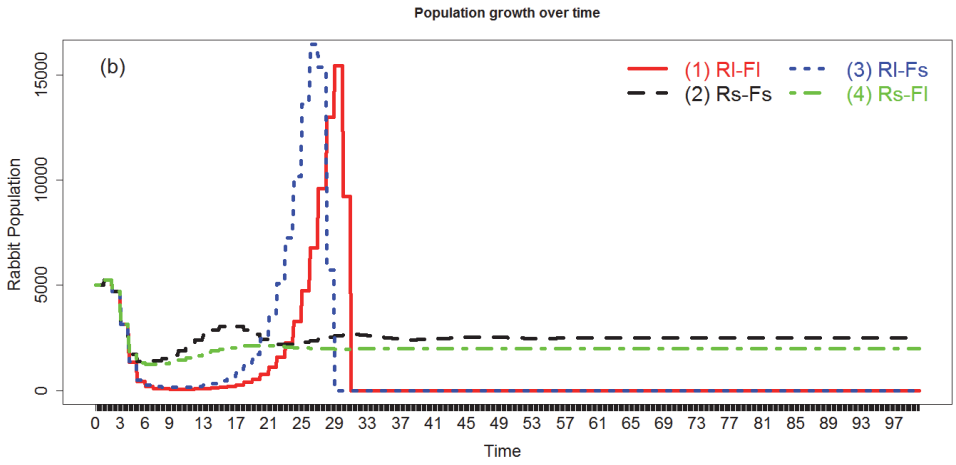
Figure 4.6: Comparison of different functional synchronizations between models; r_init_pop = 5,000, f_init_pop = 45 and r_ts = f_ts = 0.1. (a) Rabbit population dynamics over time. (b) Fox population dynamics over time. (c) Phase portrait for the population dynamics diagram shown in (a) and (b). Euler integration method is used in all cases.

First we investigate the coupling of modules that use different time steps. Consider graphs in Figure 4.7 where (a) r_ts=0.1 and f_ts=1, and (b) r_ts=1 and f_ts=0.1. In linking components with linear and s-shaped trophic functions, one 'corrects' the data towards the Rl-Fl pattern (oscillations) (graph 1), and the other 'corrects' towards the Rs-Fs pattern (equilibrium) (graph 2). The results show that graph (3), which is produced by the Rl-Fs coupling, is similar to the Rl-Fl pattern (graph 1). Likewise, graph (4) which is the result of the Rs-Fl combination has a trajectory similar to graph (2). Apparently, when linking components with different trophic functions, whether we use the same or different time steps in participating models, the data 'correction' process of one of the components will dominate in deciding the pattern of the integration output. At the same time we see that going to larger time steps was safe when done for the Fox model (in fact it appears that the accuracy has even increased since we see less amplification with time in the oscillations), but not so when increasing the time step in the Rabbit component: here we see that the model crashed after a few oscillations.

It appears that it is the Rabbit population model that 'drives' the overall dynamics: when the Rabbit population uses the linear function, the overall community model behaves as the linear function dictates; when the Rabbit population assumes a function with saturation, the community model also switches to the steady-state dynamics. May this be because the Rabbits have much higher population numbers, which therefore makes their dynamics dominant in the overall model? In all the previous cases the initial population of rabbits was 5,000 and the initial population of foxes was substantially lower - 45. This is of course driven by the ecological considerations behind these models. However perhaps that is what explains the dynamics of the community model. To test this we have simulated a community with initial population values of 100 for both rabbits and foxes. As seen from Figure 4.7(c) now replacing the trophic function for one of the species does not substantially affect the overall community dynamics: for both Rl-Fs and Rs-Fl models we get trajectories similar to the linear Rl-Fl type. So apparently the initial population values do matter, but it is yet to be seen why the linear type of performance (oscillations) appears to be dominant and what exactly it takes to switch to the saturated type of dynamics (equilibrium).

What we see is that, when integrating components that use different formalizations for the same concept we should also perform sensitivity analysis to understand how these differences play out in the overall dynamics.


Population growth over time
(a)
(1) RI-FI
(2) Rs-Fs
(3) RI-Fs
(4) Rs-FI
Rabbit Population
0
2000
4000
6000
8000
0 3 6 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Time


Population growth over time
(b)
(1) RI-FI
(2) Rs-Fs
(3) RI-Fs
(4) Rs-FI
Rabbit Population
0
5000
10000
15000
0 3 6 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
Time

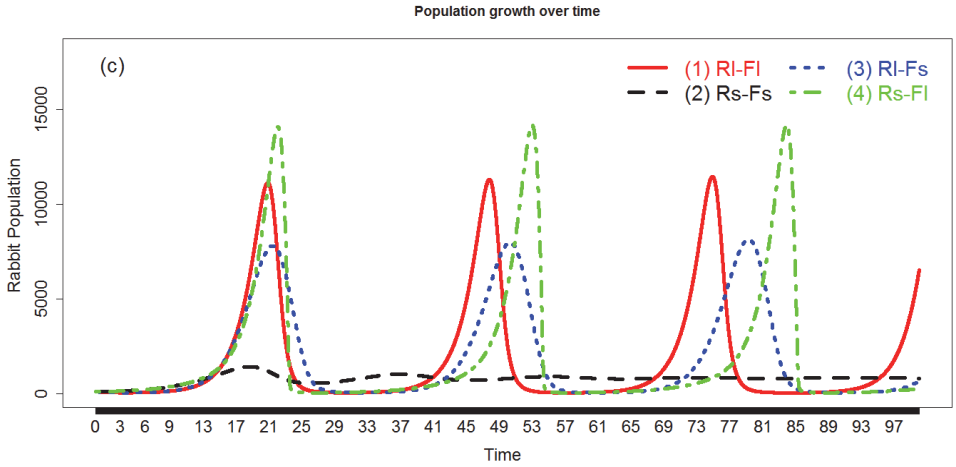


Figure 4.7: Comparison of functional synchronization between models over different time steps and population values (a) r_init_pop=5,000, f_init_pop= 45, r_ts=0.1 and f_ts=1; (b) r_init_pop = 5,000, f_init_pop= 45, r_ts=1 and f_ts=0.1, (c) r_init_pop = f_init_pop =100, and r_ts=f_ts=0.1. Euler integration method is used in all cases.

4.4 Discussion

Model integration can result in some unexpected and unintended results (Voinov and Shugart, 2013). In addition to various issues with model assumptions, semantics, scale, resolution, etc., we do need to keep in mind that when linking component models we may be also adding several degrees of freedom to the overall coupled system from the various combinations of time-stepping assumed in the components, as well as from the various numeric methods and functional responses implemented in them.

When we simulate a system, the way we manage the time steps will ultimately influence the output of the simulation (Cellier et.al. 2006). Using smaller time steps, even in one of the models, can provide better accuracy in the output. The model results turn out to be quite sensitive to the choice of combinations of time stepping applied and care should be taken when deciding what combination is most efficient.

Similarly, introducing higher order numerical integration methods in one of the coupled models can give us better accuracy in the overall output, but we also need to identify to which model we apply a higher order integration method. Common sense tells us that applying more accurate numerical schemes has a higher payback when used in the context of variables that have larger values and therefore produce higher rates of change in the overall calculations. This was indeed observed in our experiments, but not

always. Careful sensitivity testing under different combinations of parameters and initial conditions may be the only solution in this case.

This dependency on the size of the state variables involved was also clearly observed in our analysis of sensitivity due to functional synchronization of models. Switching from one functional response to another, assumed in one model or another, can substantially impact the overall results, producing even a totally different qualitative behavior of the system. Generally, the sensitivity analysis we performed was rather qualitative and did not quantify by how much one given factor is more important than another one (Saltelli et al., 2004).

We should definitely keep these findings in mind when doing research on integration of real models, like for example the integration of macroeconomic Computable General Equilibrium (CGE) model with agent based energy market model (ABM) in the on-going COMPLEX[48] project, which actually led us to the analysis described above. The CGE model simulates interaction of many economic sectors and it operates at a yearly time step. The model is written in GAMS[49], while the ABM is developed in NetLogo[50]. The ABM focuses on the residential energy demand and operates quarterly. In its current state the integrated CGE-ABM model studies only the electricity consumption dynamics and its impact on market prices, while an extension of the models towards gas consumption is envisioned.

At initialization the ABM receives information from the CGE about distribution of household incomes, aggregate shares of grey vs. green energy, energy consumption of sectors other than residential, and aggregate supply of both types of electricity and their prices. As the simulation goes on, the household agents in the ABM consider several decisions that influence their energy use, e.g., switching between green and gray electricity, buying energy efficient equipment and bulbs, or their actual behavioral, e.g., switching off the lights when leaving a room. During each 4th step of the ABM the electricity market as a whole is taken into account. The total residential electricity demand is summed up with those coming from other sectors considered by the CGE, and matched with the electricity supply, separately for green and grey electricity. Prices for both types of electricity are determined based on the prices in the previous period adjusted to the excess of supply or demand for each electricity type (Niamir and Filatova, 2015). Currently, the supply of each type of electricity comes from CGE but it is expected that the supply side will also be disaggregated in the ABM with a possibility to model technology diffusion. The new prices are returned to the CGE, which in turn

[48] http://www.complex.ac.uk
[49] http://www.gams.com/
[50] https://ccl.northwestern.edu/netlogo/

spreads the changes across all other sectors, re-estimates electricity needs for each sector and calculates the corresponding CO2 emissions, changes in sectors productivity (Filatova et al., 2014) and in incomes of households that are employed in those sectors. The new household incomes and electricity supply for both types of electricity are returned back to the ABM, which starts its quarterly activity again.

The two models have been wrapped as web services to use the integration framework described above. However the synchronization process is still to be decided. To synchronize the two models we have three options: (1) as at present, time step of the ABM can be set to t_{abm}=0.25 and for the CGE to t_{cge}=1. In this case the challenge is that the ABM has to use constant energy demand values for simulations at t = 0.25, t = 0.5, and t = 0.75. (2) Time step for both models can be set to ts=0.25, and in this case the challenge is to find relevant data or perform data disaggregation for every CGE model run. (3) Let t_{abm}= t_{cge}= 1, now we are making the ABM to operate on the same time step with CGE, i.e. on yearly bases. In all options integration output is sensitive to both data disaggregation process and time steps used. Besides usage of higher order integration methods in one or both of the models can improve the output. The optimal solution should be certainly decided after performing multivariable sensitivity analysis.

4.5 Conclusion

As in science in general, in modeling "uncertainty is not an accident" (Saltelli et al., 2008), it is an intrinsic part of it. We have split a simple classic predator-prey model into two components to demonstrate that when integrating them back together the output is sensitive to the time steps assigned to each of the components when they are run asynchronously. We find that using a smaller time steps in one of the components is not symmetrical. That is, sometimes we can gain in performance by allowing a larger time step in one of the component models without much loss of accuracy, but we should be selective in choosing which component will be using which time step. It does matter in which component the smaller time step is introduced.

Higher order numerical integration methods require more computational time, but introducing them in only one of the coupled models can significantly improve the accuracy of the output. However to which of the coupled models we apply higher order numeric methods is also not symmetrical and does matter.

In integrating models, if the participating models use different mathematical expressions to represent the same concept, sensitivity analysis on the two

expressions needs to be done. For example, in our case, the expression used in one of the models was dictating the pattern of the output.

In this research we have considered only sensitivity of integration output with respect to three aspects. Depending on the nature of the models to be integrated we may need to explore sensitivity to other factors, e.g. spatial resolution. Besides we have used only one-at-a -time sensitivity analysis approach (Campolongo et al., 2007), in which we vary one factor at a time and measure the variation in the output. Depending on the requirement of integration we may need to perform multivariable sensitivity analysis experiments. Trying to compensate for the model integration overhead by economizing on the accuracy within individual component models can be a risky idea and certainly deserves some careful testing before being recommended in the context of the full integrated model.

Chapter 5

Web service based approach to linking heterogeneous climate-energy-economy models for climate change mitigation analysis*

* This chapter is based on:
Belete, G.F., Voinov, A , Bulavskaya, T, Niamir, L, Dhavala, K, Arto, I, Moghayer, S, and Filatova, T. Web service based approach to linking heterogeneous climate-energy-economy models for climate change mitigation analysis. (In Review after revision, International Journal of Energy)

Abstract
Climate change mitigation and adaptation analysis requires understanding the causes of multiple socio-environmental processes and interactions to identify possible alternative scenarios and actions. We linked independently developed models that focus on climate, energy, and economy to enhance their scope and functionality in application to climate change mitigation. The models were originally developed to serve as standalone tools for their own specific purposes. They operate at various levels of complexity, and at different temporal and spatial scales, from individual to global. They were developed using quite different assumptions, modeling paradigms and tools. One of them is a Computable General Equilibrium model, the second one is an Integrated Assessment Model, and the third one is an Agent-based Model. We present the integration process and the internal details of the integration framework and tools and show how they can be applied in similar integrated modeling studies. Extensive pre-integration assessment was performed to identify the 'appropriate' models and to identify the links between them. We used the web service based approach for coupling since it enables interoperability between heterogeneous systems and provides open access to information for a wider community of users. The linked models can be applied to simulate various climate change mitigation scenarios, which could be difficult to do using model components in stand-alone mode.

5.1 Introduction

Climate change can affect the energy sector in different ways (Wachsmuth et al., 2013), for example by affecting the efficiency of power plants and the demand for energy (Ciscar and Dowling, 2014). To alleviate the effect of climate change, mitigation and adaptation actions should be taken in various spatial scales - from individual-household level to province, country, regional, and global levels. These actions depend on the specific context of the scale chosen, as well as on the particular locations and cultural, historical and political settings in those locations. These climate change mitigation actions need to be tuned for specific regions, countries and socio-economic groups and communities. There may be a number of alternatives, each coming with their possible consequences and implications. Choosing the best ones is always difficult and is associated with much uncertainty and unclear metrics to judge optimality. In many cases we need to carefully assess a variety of different hypothetical combinations of parameters and forcing functions, representing various climatic trends and mitigation or adaptation opportunities. These we call scenarios and these also need to be analyzed at different spatial and temporal scales.

Models are commonly used to understand and analyze a system and its possible future states (Voinov, 2008). There are scores of computer models

that may be of relevance for climate change mitigation and that can simulate various aspects of relevant complex socio-economic systems with various levels of detail. However, it is hardly possible to find a single model that can provide for all alternative climate mitigation scenarios at all required levels of complexity, from household to global levels. Integrated assessment models (IAMs) are very good at tracing feedbacks between global economy as a whole and climatic system but are heavily criticized for miscalculating the social costs of carbon and omitting cross-sectoral impacts, region-specific characteristics, and endogenous technological change (Gillingham et al., 2008), though progress is being currently made by various research teams to tackle the latter (Geels et al., 2016). Computational General Equilibrium (CGE) models are designed to trace cross-sectoral feedbacks of any endogenous or exogenous changes at a country or regional level and their consequences for the economic welfare but are weak in scaling up to global climatic processes. They ignore heterogeneity when using the representative agent paradigm, implying that everyone in a society is the same, makes perfectly informed choices, does not learn and does not change over time. Agent-based models (ABMs) are developed to address these drawbacks and can treat cumulative impacts of behavioral changes and policies for demand side activation. They can model endogenous emergence of technological innovation, but remain confined to local scale because of mounting complexity. The challenge is to link the existing fragmented knowledge and models, capitalizing on the strengths and capabilities of individual components instead of developing new models from scratch each time we are facing another problem. For this we need some methods of integration (Arnold, 2013) that can link the 'appropriate' component models into a system-of-systems finding complementarity in these models (Janssen et al., 2011) and compensating for limitations of one model by what other models may offer (Stoorvogel, 1995). Such integration bears the promise of better coverage and detail at multiple scales.

However model integration is not a simple process and comes with several challenges that need to be addressed. First we need to identify, access, understand and be able to run individual models that can be used to simulate the system we are investigating. This means that we need to translate our research goals into some system modeling exercises and then map this overall modeling effort into a relevant combination of subsystems and submodels that can be useful. In a way this is like cutting the big picture into a jigsaw puzzle, making some of the pieces similar to what we have in some library or repository. When doing that we also have to keep in mind that many models are poorly documented (Grimm et al., 2014), often described in diverse research papers, which may not present the full capabilities of the models. Commonly we find that much information about a model is stored in modelers' memories (Bonet et al., 2014), does not get

properly documented, and even when it is written up it tends to assume different terminology, semantics, units, which makes automated linking of models time consuming and frustrating. Besides, models may require configuration, parameterization and pre-processing of input data. These stages are very crucial parts of the modeling process but traditionally are most poorly documented. The second challenge of integration is to provide the actual interoperability between independently built models at different levels that is needed to run them in concert. Generic blueprint of interoperability for all kinds of integration is challenging since research settings are widely diverse among integration projects (Arnold, 2013). Integration process needs to consider the technical, semantic, and dataset aspects of interoperability. Synchronization of these models also depends upon the combinations of time steps and numeric integration methods used in component models, to which results can be quite sensitive (Belete and Voinov, 2016). The optimal settings depend on the specific context of the integration, which can be identified only by running extensive sensitivity analysis for integration schemes as well as for component models.

In addition, we need to figure out how to validate our results and how to present them to the users. Unlike the regular modeling processes when we have full control over the model and can make modifications if required, in case of integrated modeling we are dealing with a semi-closed system, when we may have little or no access to components. An error or an undocumented feature in one of the components can easily propagate through the whole system generating output that will be difficult to interpret and debug.

Here we are going to focus on specific aspects of model integration that pertain to the pre-integration and actual module coupling steps. The specific context of the study comes from linking climate-energy-economy models to investigate different climate change mitigation actions. The model space is taken from the EU FP7 project called COMPLEX, which is dedicated to research on alternative policy options for transition of Europe towards a low carbon society by 2050 (COMPLEX, 2016). The project consists of 17 partner institutes and universities across 11 countries in Europe. The COMPLEX model suite includes a number of models that focus on climate, hydrology, power market simulation, land use, macro-economics, house-hold energy consumption preferences, etc. The spatial coverage of the models ranges from global level to province/household level, with temporal resolutions that go from weeks to decades. One of the objectives of the project is to develop a framework (by integrating knowledge and models) that could be used to explore alternative pathways of transition to low carbon economy.

Analysis of the impact of climate change mitigation actions has been investigated by different groups using various techniques at different spatial

scales. Exploring climate change mitigation scenarios using a single model is the most common approach. For example, Vaillancourt et al. (2014) used TIMES-Canada energy model to simulate and analyze the possible trends of the Canadian energy system on a 2050 horizon. Similarly, Ruamsuke et al. (2015) used CGE model to assess the impact of climate change policy in Southeast Asian countries. They are able to simulate policy scenarios for different countries in the region. On the other hand, Lucena et al. (2016) used multi-model comparison approach for the analysis of climate policy scenarios in Brazil. They compared scenarios produced by six models with different baseline assumptions. Soft-linking of models is another type of approach used for analyzing the effect of climate policies. For example, Fortes et al. (2014) used soft-linking model integration strategy to develop an integrated technological-economic modeling platform by linking TIMES integrated system with GEM-E3 model. By connecting the two models they were able to simulate energy-climate scenarios for the analysis of economic, technological, and environmental impact of energy and climate policies for a case study in Portugal.

In this research we see model integration as an option to explore the complexity of climate change mitigation actions by combining the different knowledge represented in several models (Hamilton et al., 2015). To simulate climate change mitigation scenarios in a more detailed way we integrate models that operate at global, regional, and household level. The mitigation actions are complicated because of incomplete, contradictory, and changing requirements and information about the socio-economic system. Engaging stakeholders in the process is one essential way to address such problems. One way to remove technical and accessibility constraints and facilitate stakeholder participation is by presenting the models as web applications, making them available through standard web browsers. This led us to using web services to wrap legacy models, converting them into interoperable components that can be accessed with a regular web browser. In our framework the users do not need to install any additional software and can use their web browsers to explore modules as stand-alone components, or build chains of modules by connecting outputs from one module to inputs of other modules. Our goal is to enable non-technical stakeholders to simulate different climate change mitigation policy scenarios and improve their understanding of climate-society-environment interactions.

This paper describes the integration process that was developed, starting with the pre-integration assessment, then the framework design, and implementation. Section 5.2 discusses how the pre-integration assessment process was performed by the project members, and how the high-level conceptualization of the integrated system was developed. The interactions between models are detailed in Section 5.3. Section 5.4 presents how

interoperability is established using web services, and it also describes the architecture and implementation of the integration system. We present an example of an integration of IAM, CGE and ABM energy models by means of the online model integration platform developed. Section 5.5 discusses the lessons learned from the integration process and how it can be applied in other similar projects. The last section provides some conclusions.

5.2 The pre-integration assessment process - identifying the models and possible linkages

5.2.1 What is pre-integration assessment

The pre-integration assessment is the first phase of integration in which we identify the participating models and decide whether linking these models can provide scientifically valid output (Belete et al, 2017). Basically, the assessment process is dictated by the research objectives. So far we find it unlikely that the process can be fully automated: there is still no clear agreement on standards for model documentation, and as a result it is often hard to understand what exactly particular models do and how they are relevant to the project goals. To compensate for lack of standardized documentation we engage in discussions with stakeholders - model owners, developers, and model users who have interest in the integration output. What we can strive for is some sort of computer-aided design, when software tools can assist in informing stakeholders about the candidate models and linkage mechanisms, leaving the final decision to the actual system designers. Generally, the process of pre-integration assessment can be represented using the steps indicated in Figure 5.1. It is an iterative, highly collaborative process, which requires active participation of all members.

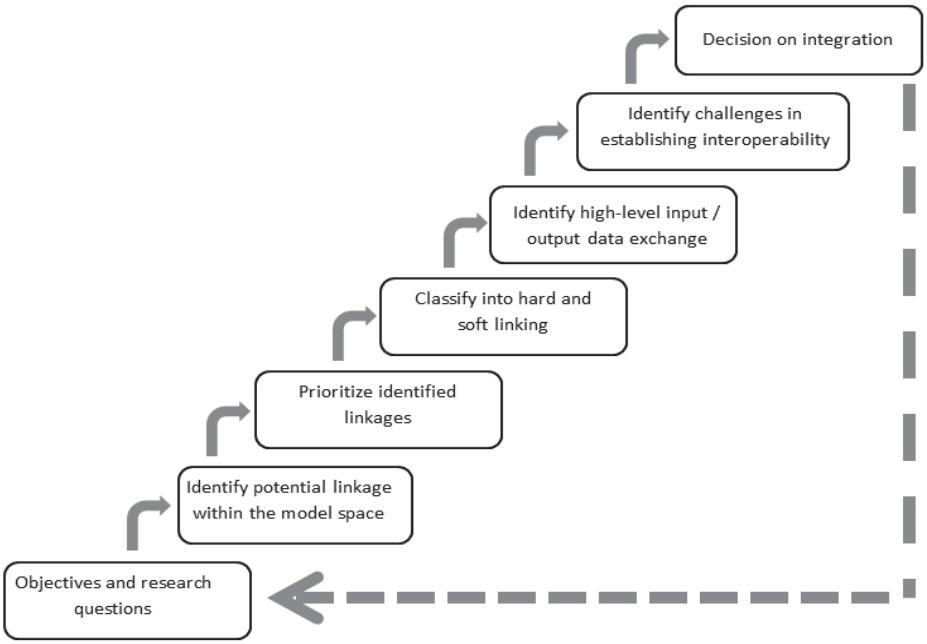


Figure 5.1: Steps of the pre-integration assessment process within a certain collection of models.

In our case the stakeholders were model developers from the COMPLEX project, and the discussion was centered on the features of the models brought by them. The goal of the pre-integration assessment was to develop a framework that supports climate change mitigation analysis at regional, country, provincial and/or individual levels.

We started the pre-integration assessment by conducting a one-day workshop that involved all model owners, with the aim of describing the models and identifying the possible linkages to other models that could lead to a system-of-systems for climate change mitigation analysis. All the models available in the project are presented in Figure 5.2. The model owners were asked to describe what their models do, define the temporal and spatial scales in which they operate, list input-output data, the tools used to develop the models, etc. From these descriptions, model owners got the opportunity to explore which output from which other models could provide input to their models. The need for 'better' input data is the driving factor for linking a model with other models. Ultimately the data exchange between models gives the chance to develop a system-of-systems, which goes beyond the functionality of individual models and rests on numerous previous expertise embedded in other existing models. Based on this, by the end of the workshop we were able to identify a number of promising possible linkages between the models in the COMPLEX repository.

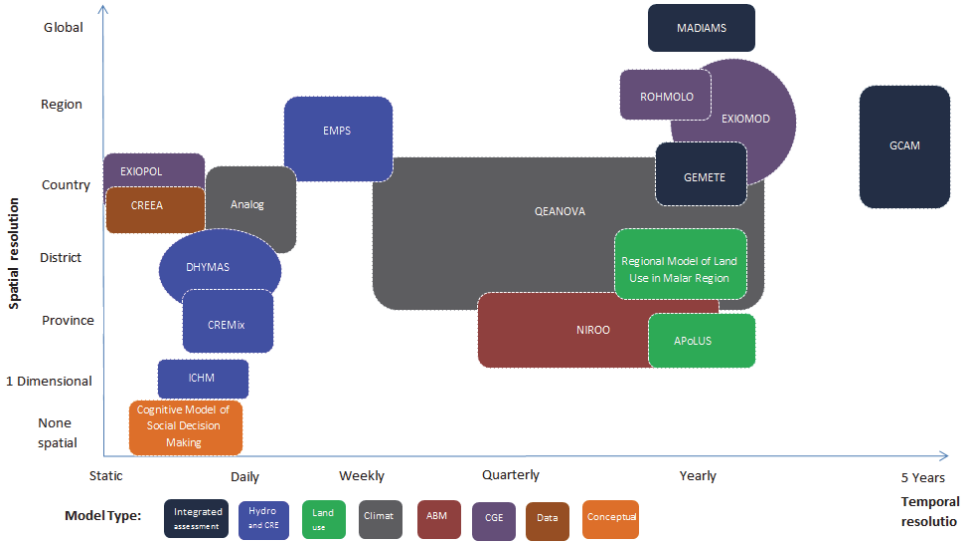


Figure 5.2: The COMPLEX project model space.

Following this workshop, based on our general objective we formulated high-level system requirements that we would like to address by using the envisaged integrated system. The purpose was 'To explore the effects of implementing United Nations Framework Convention on Climate Change - UNFCCC- policy scenarios (UNFCCC, 2014) in EU27 countries'. According to this convention the plan is to reduce emissions of greenhouse gases by 20% by the year 2020; by 40% by 2030; and by 80% by 2050. By using one of the models from our repository we can explore how such policy will affect electricity production, and by using another model we can also explore how different sectors of the economy will be affected as a result of implementing the targeted policy. However, change in electricity production can have significant consequences on different sectors of the economy. Due to this, linking the standalone systems can provide a wider picture of the effect of implementing the UNFCCC convention than when using separate models. We need the integrated system to answer such questions as: 'what will be the effect of a policy in different sectors of the economy?' or 'which sectors will be affected most?' After agreeing on this objective, a number of smaller pre-integration assessment meetings were conducted focusing on specific and already identified model linkages. The information gathered from the meetings helped us to prioritize the linkages based on the level of relevance to the system we would like to build, the feasibility, and the resources at hand. We would like to point out that the prioritization and selection of models was not clear-cut, it required much discussion and some compromise to address all the above-mentioned factors.

Among the available models both GCAM (Kim et al., 2006; Wise et al., 2009) and EXIOMOD (COMPLEX, 2016) simulate the global economy and can be used for integrated assessment. Policy scenarios that we would like to explore could be fed into GCAM as one of its simulation inputs. On the other hand, EXIOMOD can simulate how the different sectors react to economic inputs. Therefore linking GCAM and EXIOMOD can be used to analyze the effects of a policy scenario on different sectors of the economy. In addition, linking the Agent Based energy market model, NIROO (COMPLEX, 2016), with other macro models could help us to simulate the cumulative impacts of behavioral changes with respect to energy use on the demand side. NIROO complements the two macro models by explicitly modelling micro level dynamics accounting for behavioral aspects of individual decisions. For example, this may allow to trace the impacts of information policy on the residential energy demand, which are commonly difficult to account for in CGE and IAM models (Geels et al., 2016). Besides, since the two macro models operate at country/regional level, linking these models with household/province level processes presented in the NIROO could provide the functionality of analyzing the system performance at different scales. Due to this the three models presented in Table 5.1 were selected for the first phase of integration. More detailed description about these models is given in the following sections.

Table 5.1: Brief summary of the three models selected for integration.

Models	Model type	Temporal scale	Spatial scale	Examples of application	Software
GCAM	IAM	5 years	Regional level (world as a collection of 32 regions)	Global change assessment.	C++, Java, XML
EXIOMOD	CGE	1 year	Country level (43 countries + the rest of the world)	Evaluation of the impact of policies related to (energy and non-energy) resource use at the macroeconomic and sector levels.	GAMS
NIROO	ABM	Multiple temporal scale	Province scale	Studying non-marginal shifts in energy markets via simulating behavioral changes among households, and diffusion of low-carbon energy sources.	NetLogo

In addition, we also identified models, which could be hard-linked and soft-linked. Here by hard linking we mean creating automated data exchange between models while keeping them separate, and by soft-linking we mean

using file-based manual data exchange between models. Further on, we classified the hard-linked type of integration into (1) bi-directional and iterative data exchange between models, which makes intermediate output accessible at the end of each time step, and (2) one directional data exchange between models that releases output only after the final time step. Following these discussions, three models were identified for the hard-linking type of integration, but it became also clear that the final decision whether to continue integration or not required more detailed information and further analysis on establishing interoperability between the selected models.

After identifying the participating models the next step of the pre-integration assessment was the conceptualization of the integrated system and the exploration of the particular input-output data exchange pattern between the models. Spatial and temporal scales of input/output were also the main areas of discussion. We learned that establishing an automated data exchange between models requires addressing the following challenges:

1- There is a mismatch between spatial scales of models. We have two models that operate at the global level but they represent the world in a different number of regions/countries. The third model in our set operates in province/household level. We agreed that the two models that operate in global level can exchange input-output by aggregating and disaggregating data based on the corresponding spatial scales. To link the province level model to one of the global level models we agreed to modify the global level model to produce province level data for two selected case study areas.
2- There is a mismatch in the temporal synchronization of the models. One of the global level models operates with a five-year time step and releases its output only after the final time step. Due to this we concluded that only one directional data flow could be established with this model. However, the other global level model and the province level model operate on a yearly basis. These two models could exchange input-output at the end of each year.
3- The three models are developed using different programming languages, that is, C++, NetLogo, and GAMS. Establishing automated data exchange between such models requires language interoperability mechanisms between the models. We agreed to develop code on top of each models - i.e. wrappers- so that technical interoperability among them will be possible.
4- One of the models uses a proprietary database, which cannot be shared outside the developer organization. The model owners negotiated that they can present only the relevant portions of the database as binary files (also called work files or scratch files).

Despite those challenges, from the information gathered during high-level conceptualization of the integrated system, we noticed that integration of the three models can be done with the resources at hand, and we decided to continue with the integration process. More detailed description about each of these models is given below in sections 5.2.2, 5.2.3, and 5.2.4. In this paper we focused only on hard-linking type of integration, assuming that soft-linking is already better explored.

5.2.2 Global Change Assessment Model (GCAM 4.0)

GCAM is a dynamic-recursive partial equilibrium model with detailed representation of the economy, energy sector and land use that can be used to explore climate change mitigation policies including carbon taxes, carbon trading, regulations and accelerated deployment of energy technology. Regional population and labor productivity growth assumptions drive the energy and land-use systems assuming numerous technology options to produce, transform, and provide energy services as well as to produce agricultural and forest products, and to determine land use and land cover. The model includes a default scenario for the future evolution of key socioeconomic drivers (population, labor force participation, and labor productivity), which can be changed by the user. Once the exogenous socioeconomic parameters are provided and the policy target is set, the model can predict "service" demands (e.g. transportation), greenhouse gas emission by region and sector, climate indicators such as CO2 concentration and temperature, policy costs, land use patterns, prices for different markets, etc. The "service" demands are linked to energy service demand, energy price, and performance of associated alternative energy conversion technologies.

GCAM operates from 2010 to 2100, and spatially it represents the world as a collection of 32 regions. GCAM is developed using C++ and uses extensive XML based configuration files. To set inputs the user has to edit the configuration.xml file and/or other XML files associated with it. The model can be run by double clicking the executable file which will display the status of the simulation in the command based interface. The model stores its output in a Berkley XML Database format, called DBXML, and it also writes a range of model output data as a CSV file. To manipulate the model output, a Java based graphical user interface is provided. The user can query the model output by using different parameters, display as graphs, or export data.

5.2.3 EXtended Input-Output MODel (EXIOMOD)

EXIOMOD, the other model chosen during our pre-integration assessment is a Computable General Equilibrium (CGE) model that takes into account the interaction and feedbacks between supply and demand. The model considers

the global economy as 44 countries and the 'Rest of World' region, and accounts for 163 economic sectors per country. EXIOMOD captures not only the economic transactions, but also includes a representation of a wide range of emissions, materials and land use, related to the economic activities. CGE models use the idea of aggregate agents, and households, firms, and government are the main agents involved in these models (Ashley et al., 2007). Due to this a CGE model represents the behavior of the whole population group as a single household, or of the whole industrial sector as a single firm. These agents interact through the provision of goods, services, and taxes, and as a result of these interactions the emergent behavior of agents determines the price of goods and services. The model equations are based on the following assumptions:

- cost-minimizing behavior of producers - producers attempt to produce output at least cost,
- average cost pricing - businesses firms set the unit price of a product relatively close to the average cost needed to produce it, and
- household demands are based on optimizing behavior - they try to maximize utility while considering constraints imposed by their income and market prices.

EXIOMOD is developed using GAMS programming language, and uses an underlying database, called EXIOBASE, to define simulation inputs for countries/regions, currencies, types of industries, categories of products, and other lookup values. The model writes its output into its database and into some text files. Currently EXIOMOD operates from year 2007 to 2050 with yearly time steps.

5.2.4 Agent-based Energy Market Model (NIROO)

The third model selected for the integration is an Agent-based Model (ABM) - Nonlinearities in the Residential lOw-carbOn economy transition, NIROO. The ABM is designed to study the cumulative impacts of individual behavioral changes with respect to energy use and impacts of various policies on demand side activation. The model is designed in a disaggregated manner to trace potential energy market discontinuities driven endogenously from within the economic system or triggered by changes in the environment (Niamir and Filatova, 2015). NIROO simulates a retail energy market by disaggregating both residential demand and energy supply sides of the market. NIROO is mostly focused on modelling households' energy use and their potential behavioral changes by considering 3 types of actions: investments, switching to green(er) energy producers, or change in the energy use habits (Niamir and Filatova, 2016a). The decision making process of a household is modelled as a multi-stage process based on behavioral science theories. It offers opportunities to explore demand side activation not

only through price mechanisms but also through information policies that affect values and awareness to explore under which circumstances shifts in residential demand may lead to potential structural changes in energy markets.

Households consume energy in many different ways: electricity, heating, transport, and food. A household can get energy either from conventional fossil fuel based sources or from alternative low-carbon ones, such as hydro, solar, wind, biomass, and nuclear sources. NIROO zooms specifically into the electricity and heating energy consumption choices that households make, including switching between low-carbon (LCE) and fossil fuel (FF) based energies (Moghayer, et al, 2015). The main agents in NIROO are households, which are differentiated in terms of socio-demographic characteristics, environment and climate change awareness, and preferences. The model explicitly accounts for social networks, which can impact behavior of individual households over time and space. Supply side of NIROO is represented in a simplistic way, and the changes on supply side come through the integration of NIROO and the other two, CGE and IAM, models. External factors such as changes in income and saving, technology diffusion, energy consumption, energy prices and some internal factors such as psychological and sociological characteristics – all may influence household choices with respect to energy use (Niamir, L. and Filatova, T. 2016b). NIROO is a spatially explicit model, thus it takes input from csv files and GIS databases exogenous to the integrated modeling suit. After the simulation the model produces an output with respect to households behavioral change such as the share of low-carbon energy consumption, and new energy prices differentiated by their source of production (low-carbon vs. fossil fuels) to be imported into the CGE model. NIROO is developed using NetLogo 5.2 with the GIS extension. We use open source applications, such as PostgreSQL and R, for the spatio-temporal and statistical analyses.

5.3 Formulating detailed data exchange patterns between the models

Providing appropriate emphasis on the design of a modeling system can prevent design errors, improve quality, reduce costs, optimize performance, and produce reliable system behaviors (Garrido et al., 2009). Based on this we followed a top-down approach of system design – we start from the highest level and then we go through the details. This includes: deciding the direction of data flow from one model to the next model, defining how to manage differences in spatial scales between the models, the time steps in which the models exchange data, and how the data mediation should be performed. This process has to be done regardless of the technology we are going to use for the data exchange, which is discussed in the next section.

Formulating the data exchange at conceptual level helped us not to be constrained by implementation issues, for example, how to make these models interoperable, how to build reusable data mediation modules. This approach gave us the chance to evaluate design options without framing it within a specific implementation method.

5.3.1 Linking GCAM to EXIOMOD

From the pre-integration assessment we identified that by linking GCAM and EXIOMOD we can simulate how the climate mitigation policy scenarios affect different sectors of the economy. We also identified that the communication between the two models can only be one directional. To decide on the direction of data flow we considered the following information about the models.

- For GCAM, population and GDP are exogenous variables, which are to be provided from external data sources, such as the Shared Socioeconomic Pathways (SSP2) scenarios database of IIASA (SSP, 2012). Note that the SSP2 storyline is also called Middle of the Road approach since it lies in between the Low challenging scenarios of SSP1 and High challenging scenarios of SSP3 and also between Mitigation challenges dominated SSP5 and Adaptation challenge dominated SSP4 (O'Neill et al., 2014). It is based on the assumption that trends of the recent decades will continue, with some progress towards achieving development goals, reducing resource and energy intensity at historic rates, and slowly decreasing fossil fuel dependency. In our case, even though EXIOMOD can provide GDP predictions we decided that, in order to align the simulations with the IPCC scenarios, it would be convenient to follow the SSP2 approach. Accordingly we decided to use the assumptions of the SSP2 scenario in both models (Capellán-Pérez et al, 2014). Due to the implementation of SSP2 scenarios, the data flow from EXIOMOD to GCAM is not a priority to explore.

- For EXIOMOD, electricity generation mix is an exogenous variable. For the base year this variable is set using data from the EXIOBASE database. To run scenarios in EXIOMOD we should make assumptions about how this variable can change, depending on management decisions or policies that we model. In practice, if a target value for electricity mix is set at some point in the future, we would need to interpolate between the current EXIOBASE value and the target value. The interpolation is required because EXIOMOD runs on annual basis and needs electricity mix inputs for each run. The interpolation is usually based either on the outcomes of a more specialized energy model, or on the consensus opinion of energy policy experts. On the other hand, GCAM is a

specialized energy model, which represents energy technologies and markets very well. GCAM computes energy mix using the capacity and lifetime of technologies, among other parameters. So using the energy mix from GCAM is less ad-hoc and takes technical restrictions into account. Also, GCAM will produce different electricity generation mixes for different scenarios, which is not done when we use EXIOMOD as a stand-alone system. Besides, due to detailed data on costs of electricity generation technologies, GCAM can make a better prediction of changes in electricity prices when we shift to more expensive technologies (renewables).

Based on this, we identified that the data flow between the two models should be from GCAM to EXIOMOD as shown in Figure 5.3. Apart from electricity prices, we have identified some other variables (output) that can be used in EXIOMOD, such as carbon emissions, final energy, primary energy, refinery inputs, floor space, km of passenger transportation and price for food demand. However for now we have decided to focus only on prices and electricity mix.

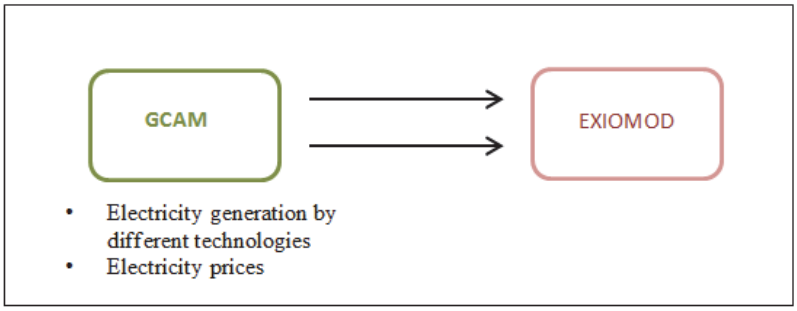


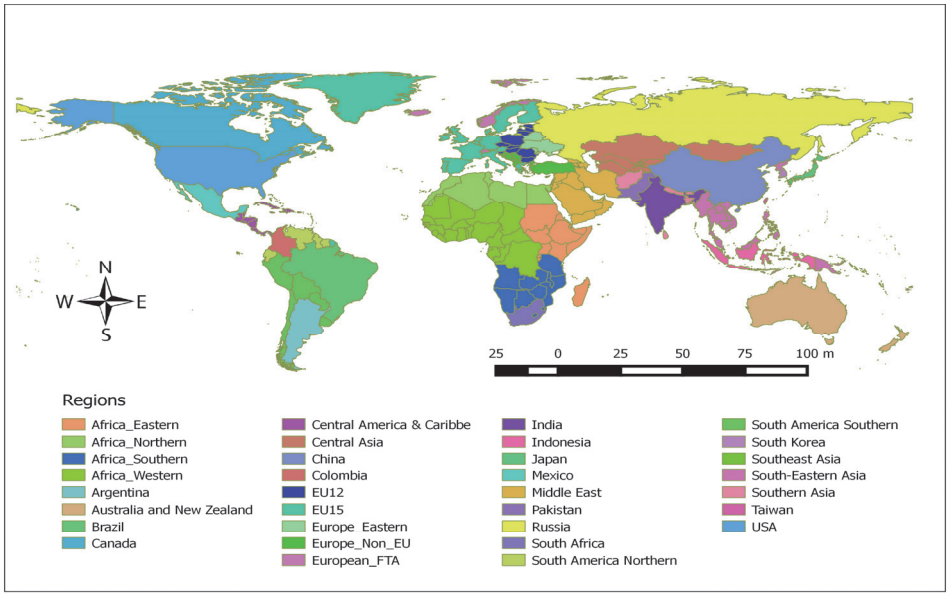
Figure 5.3: Linking GCAM and EXIOMOD.

After identifying the variables involved in the data exchange between the two models we also found out that the two models represent the data differently. The units used by the two models are shown in Table 5.2. The data produced by GCAM should be mediated before they are passed to EXIOMOD, which requires conversion both in units and spatial scales.

Table 5.2: Comparison of units for data mediation between GCAM and EXIOMOD.

	Variable	Quantities involved in data exchange	Unit used by GCAM	Unit used by EXIOMOD
1	E_t	Electricity generation by different technologies	EJ (Exa Joule)	Non-dimensional shares, sum up to 1
2	E_p	Electricity prices	1975 USD/GJ (GJ – Giga Joule)	Relative price to the base scenario

As mentioned earlier, GCAM uses 32 regions/countries to represent the world while EXIOMOD represent the world as 44 regions/countries. With this setting the data mediation requires direct matching, aggregation, and disaggregation of data between regions/countries of the two models. We found that the data disaggregation process requires additional data, which was not available at hand. To avoid the need for additional data, the regionalization of EXIOMOD was modified from 44 regions to 20 regions (Figure 5.4). The regionalization has been chosen in such a way that the resulting 20 regions of EXIOMOD are either equal or more aggregated than the regions in GCAM. The process of regional aggregation of EXIOMOD is relatively straightforward. Firstly, the data on economic transactions and environmental extensions can be just added up, based on the concordance map between the 44 and 20 regions, cancelling out the trade relationships between the regions that get merged. Secondly, the parameters of the models are recalibrated using the standard EXIOMOD formulas based on the aggregated database. This helped us to restrict the spatial aspect of data mediation task to direct matching and aggregation.



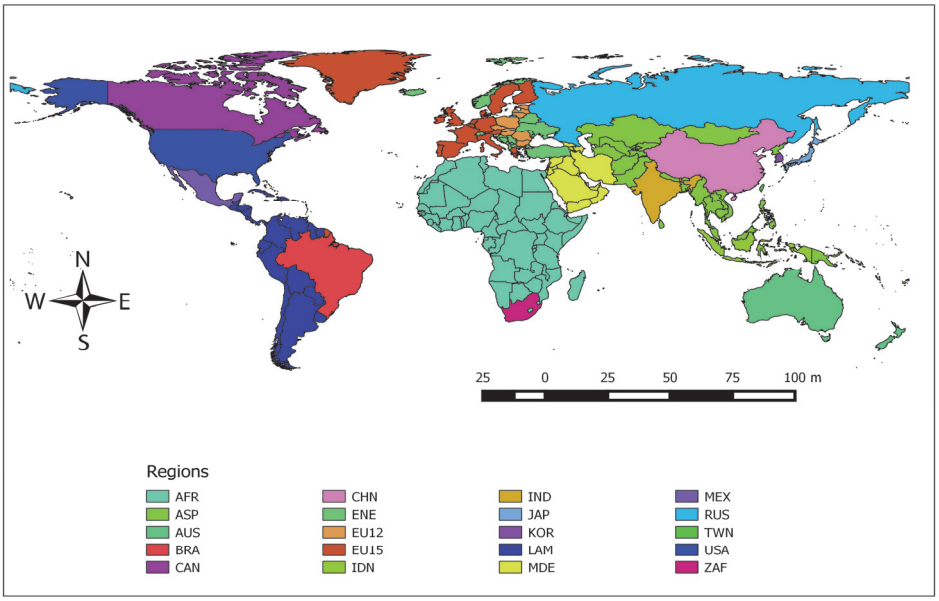


Figure 5.4: Classification of regions in (a) GCAM, (b) EXIOMOD.

The next step was to define how the data for the two selected variables would be mediated. For E_t the data conversion begins with mapping of data between regions of the two models. For aggregated regions data aggregation was done by summing of values. Moreover, E_t data include electricity generated using various technologies, such as biomass, wind, coal, gas,

hydro, oil, geothermal, refined liquids, and solar. GCAM expresses this data in terms of Exa Joules, while EXIOMOD represents it in terms of percentage of each category. Due to this, we compute the share of electricity produced by a certain category as a quotient of amount of energy produced by that category divided by the total amount of energy produced from all categories:

$$\text{Share of electricity generation of energy category x} = \frac{E_t \text{ of category x}}{\sum_{t=1}^{n} E_{tx}} \quad \text{--- (Eq. 1)}$$

Where

x = one type of energy technology category,
E_t = Electricity generation by different technologies,
n = the number of energy technology categories.

With regards to E_p, GCAM expresses it in 1975 USD/GJ, and in EXIOMOD it is expressed in relative prices, with the price in the base year taken as 1. The data mediation of E_p data from GCAM to EXIOMOD is done using the following steps. For regions that exist in both models the first step is to match the data directly. For region names that do not match directly we do aggregation of data, and to aggregate electricity price data we use weighted average, where electricity generation in EJ is used as weights. At this stage we will have E_p values for all regions of EXIOMOD in 1975 USD/GJ. Then, the next step is finding the relative price values for each region. The relative price value of a given region is computed by dividing the regional price value for that specific year with the price value of the base year (note that relative price value for the base year is 1).

5.3.2 Linking EXIOMOD to NIROO

The aim of the NIROO and EXIOMOD integration is to provide direct feedbacks between potential behavioral change with consequent changes in market shares of LCE vs. FF and impacts of these on other sectors of economy (ABM=>CGE), and as well as accounting for non-residential electricity demand and changes in households incomes as economy evolves (CGE=>ABM) (Niamir and Filatova, 2015).

As the first step of the integration, NIROO can get data such as income, saving and energy consumption of households and share of LCE vs. FF energy production at initialization from EXIOMOD. As the second step, EXIOMOD will be updated with the new shares and prices (LCE vs. FF) after the market clears and price expectations are updated in NIROO. When EXIOMOD runs and produces new sectoral impacts, the information on the growth of income, saving and energy consumption of households will be sent to NIROO. Therefore the linking of EXIOMOD to NIROO assumes a two-way data exchange (Figure 5.5).

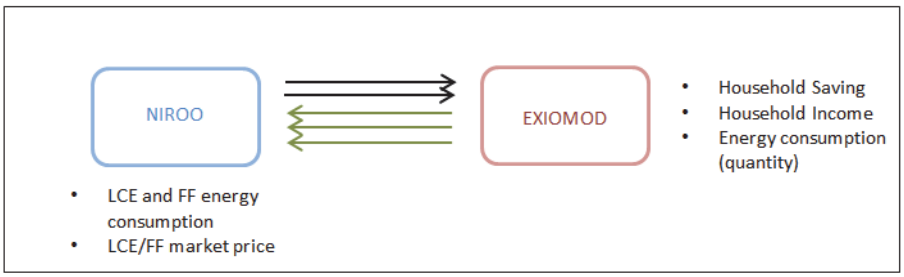


Figure 5.5: Conceptual linkage of EXIOMOD to NIROO.

Currently both models operate on yearly time steps. Spatially, NIROO is designed to function at the province level: Overijssel province in Netherlands and Navarra region in Spain were chosen. In EXIOMOD these two regions are treated as part of the corresponding hosting countries. To link the two models we made the following assumptions:

- Household income, saving, and energy consumption data produced by EXIOMOD at national level, in five quantile groups, should be disaggregated to province level so that it can be used as input by NIROO. To do this, we assume that income distribution (share of income received by five income quintile groups), saving rate (share of income that goes to savings) and share of income that households in each income quintile spend on electricity are the same in Overijssel as in the whole Netherlands. Due to the small size and relatively high homogeneity of incomes in the country, we believe that in the case of the Netherlands this assumption is quite realistic. Subject to data availability, one could also use the actual distribution of income classes in the future, especially when considering the cases of more heterogeneous regions.

- The NIROO calculates the share of low-carbon and fossil fuel energy consumption and new LCE/FF prices data at provincial level, which we have to up-scale to the country level data so that it will be used by EXIOMOD. Here also we make the assumption that the share of low-carbon and fossil fuel energy consumption at national level is the same as at the province level.

However, if we have household income, saving, and energy consumption data at province level, then data disaggregation should be done to mediate country level data to province level instead of making assumptions. Besides, these assumption may not work for countries of larger size and more heterogeneity than Netherlands.

5.3.3 Simulating Climate-Energy-Economy

Model integration is an iterative (Holzworth and Huth, 2011) and incremental process (Holzworth et al., 2014). As a first iteration we implemented the one-way communication between GCAM and EXIOMOD to explore the effect of emission reduction policy scenario on electricity production and the cascading effect of changes in electricity production and price on the different sectors of the economy. This cannot be done using these models as standalone components. Moreover, the bi-directional data exchange between EXIOMOD and NIROO (Figure 5.6) can help us to explore if such policy has any effect on household consumption of green and grey electricity. In addition, the linking of these models helps to minimize ad-hoc input-data producing processes, and to model the climate-energy-economy system across different spatial scales.

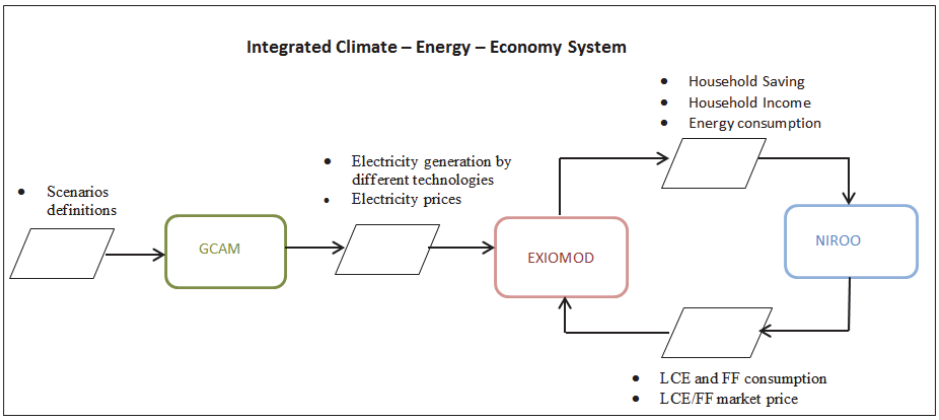


Figure 5.6: Conceptual design of the integrated climate-energy-economy system.

5.4 Web services to handle the interaction of heterogeneous models

Once we have decided which models we want to integrate we need to come up with the technique and technology to couple them. In our case, the models selected for integration were developed using very different tools and programming languages. Also very different programming paradigms were employed - C++, a general purpose programming language, NetLogo - a special programming language for Agent-based modeling available as part of special modeling interface, and GAMS - a language for mathematical programming and optimization. We need an interoperability mechanism that enables these heterogeneous models to talk to each other.

Models developed using different programming tools can interoperate if additional code is provided to set their inputs, run the model, and expose

outputs (Peckham et al., 2013). A wrapper is a thin layer of code that enables such passing of input parameters, running the model, and reading its output. Wrappers can be developed using different programming languages. We need to set clear requirements for wrappers, which we are to develop, keeping in mind that:

- Developing wrappers will be more convenient if, depending on the specific context of the model, we are free to choose among different programing languages. For example, if we can develop a wrapper for one of the models using Java and we are not forced to use Java to develop wrapper for the second model and we can choose either Python or C#, then we are providing flexible wrapping options.
- The wrapper we build for a model should be reusable for linking the model with different models, not just one.
- A wrapper that enables remote access to models (to keep models on different locations) will have a better chance of including more models and data sources into the integrated system. Such wrapper can improve the accessibility and reusability of models by a wider community.
- If the wrapper can make available metadata of model in machine-readable format then online search for models, semantic mediation, and data conversion can be more facilitated.

One of the possible solutions, which suited most of the requirements above, was to look at web services. These are modular and dynamic applications that can be described, published, located, or invoked over the network independent of the underlying hardware and software platforms (Erl, 2008a). Web services can be used to develop wrappers. This gives us the opportunity to establish automated data exchange between heterogeneous systems, which could be located anywhere on the Internet. Besides, metadata information of models can be made available as service descriptions of web services, which are commonly accessible in search engines. This can improve the discovery and reuse of models significantly.

In the modeling domain web services have been used to make environmental models interoperable (Granell et al., 2010), to integrate ecological forecasting models (Dubois et al., 2013), to link hydrological models with climate models (Goodall et al., 2013), to build large scale multi-agent simulations (Tsai et al., 2006), etc. This shows that web services have a wide range of applications.

After some deliberations we decided to use the web services approach and develop wrapper web services for the models considered above. The wrapper for GCAM was developed on top of the DLL file of the model as a C#-based web service that manipulates GCAM. The wrapper has also functions that can

update input XML files and that fetch output data from the database. Similarly, the wrapper for EXIOMOD was built on top of GAMS.NET API. The API runs GAMS-based models (i.e. GAMSJob), and can customize GAMS settings (i.e. GAMSOptions). The wrapper web service consists of functions to set simulation inputs, run model, and query model output. The wrapper web service for NIROO is developed using Java. This is because NetLogo provides a Java-based library file (NetLogo.jar), and using this library the wrapper web service can manipulate the underlying model. In general these wrappers can be used to run the models as stand-alone web services-based models and also to establish automated data exchange between them.

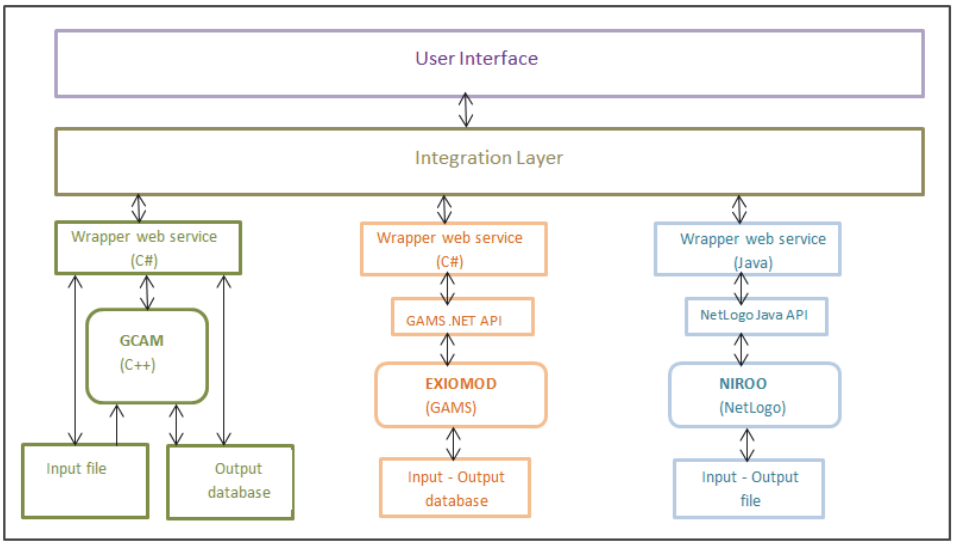


Figure 5.7: Architecture of the integrated climate-energy-economy system. Models are wrapped with web services and transformed to web-enabled components. The integration layer handles the communication between participating component models.

On top of wrapper web services, an integration layer (Figure 5.7) was developed, which manipulates web service-based models. The semantic mediation and dataset conversion functionalities are also built as part of this integration layer. Users can interact with the system using web-based user interface, which can be accessed from the COMPLEX project website. The three wrapped models and the integrated system can be accessed from any computer connected to the Internet using a standard browser without the need for installing the underlying modeling platforms.

As we can see in Figure 5.8, the user can select the simulation end year (between 2008 and 2050), the type of scenario, and then run the simulation. During simulation the communication between models is orchestrated as

follows. GCAM runs the simulation with a time-step of five years until the year 2050. It produces output every five years, i.e. for 2010, 2015, etc. EXIOMOD reads input data both from its database and from output of GCAM, and it produces simulation output for every year, i.e. starting from year 2008 until 2050. Since the communication between EXIOMOD and NIROO is bi-directional, at the end of each simulation period the models exchange data. Some simulation outputs from the models are displayed in the GUI and the user can also download it. The complete model output can be accessed from the output files of the corresponding models.

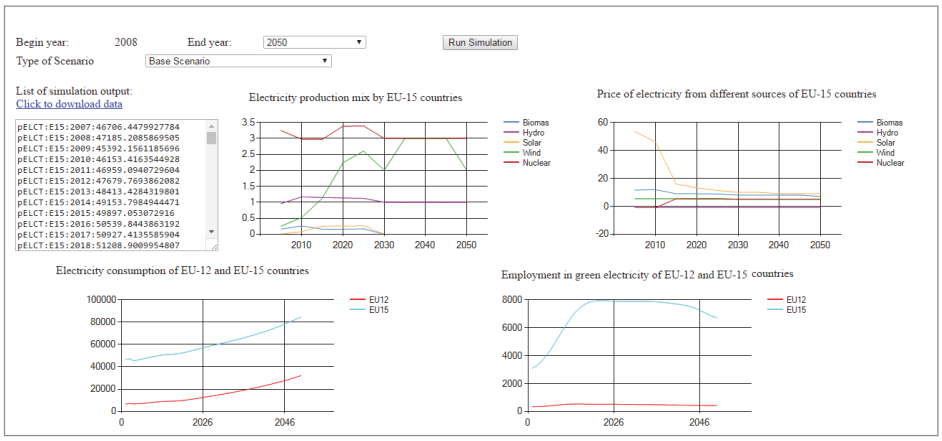


Figure 5.8: An example of web-based user interface of the integrated system.

5.5 Results and Discussion

As a proof of concept we presented the results of an implementation of two global level scenarios for GCAM and simulated them using the integrated system. The first scenario is business as usual, or base scenario, in which GDP and population values are set using the SSP2 scenario and no specific climate policy is enforced in the system. The second case is when the emission reduction policy shown in Table 5.3 is implemented in the system. The targets here are based on the Intended Nationally Determined Contributions (UNFCCC, 2014). Based on this we computed targeted emission amounts from historical emission data and the targeted emission reduction percentages to drive the system. Here we will present the results for EU12 countries (Romania, Bulgaria, Cyprus, Czech Republic, Estonia, Hungary, Lithuania, Latvia, Malta, Poland, Slovakia, Slovenia) and EU15 countries (Austria, Belgium, Germany, Denmark, Spain, Finland, France, United Kingdom of Great Britain and Northern Ireland, Greece, Ireland, Italy, Luxembourg, Netherlands, Portugal, Sweden).

Table 5.3: Targeted emission reduction policy scenario for different regions and countries to be simulated using the integrated system.

Name of country (region)	**Targeted indicator**
Australia	Base year 2000: 25% reduction by 2020
New Zealand	Base year 2005: 30% reduction by 2030.
Canada	Base year 2005: 17% reduction by 2020; 30% by 2030.
China	Peaks CO2 emissions in year 2030
European Free Trade Association: Iceland, Switzerland and Norway	Base year 1990: 30% reduction by 2020; 35-40% by 2025; 50% by 2030
EU-27	Base year 1990: 20% reduction by 2020; 40% by 2030; 80% by 2050
Japan	Base year 1990: 25% reduction by 2020. Base year 2005: 25.4% by 2030
Mexico	Base year 2000: 50% by 2050
Russia	Base year 1990: 15-25% by 2020 ; 70-75% by 2030
South Korea	Base year Business As Usual: 37% by 2030
USA	Base year 2005: 17% reduction by 2020; 42% by 2030; and 83% by 2050

The first spot to track the effect of the implemented scenario is at the interface between models when data is passed from one model to the next one. E_t and E_p are the two variables, which are passed from GCAM to EXIOMOD. Consider Renewable Electricity Production (which is a subset of Et) and Ep data of EU12 and EU15 countries for the base and policy scenarios shown in Figure 5.9 (a) and (b). With the targeted policy scenarios both renewable electricity production and Ep will increase. The increase in prices could be associated with the higher cost of production of green electricity. In fact, renewable energy systems are one of those sectors, which will be strongly affected by climate change (Wachsmuth et al., 2013).

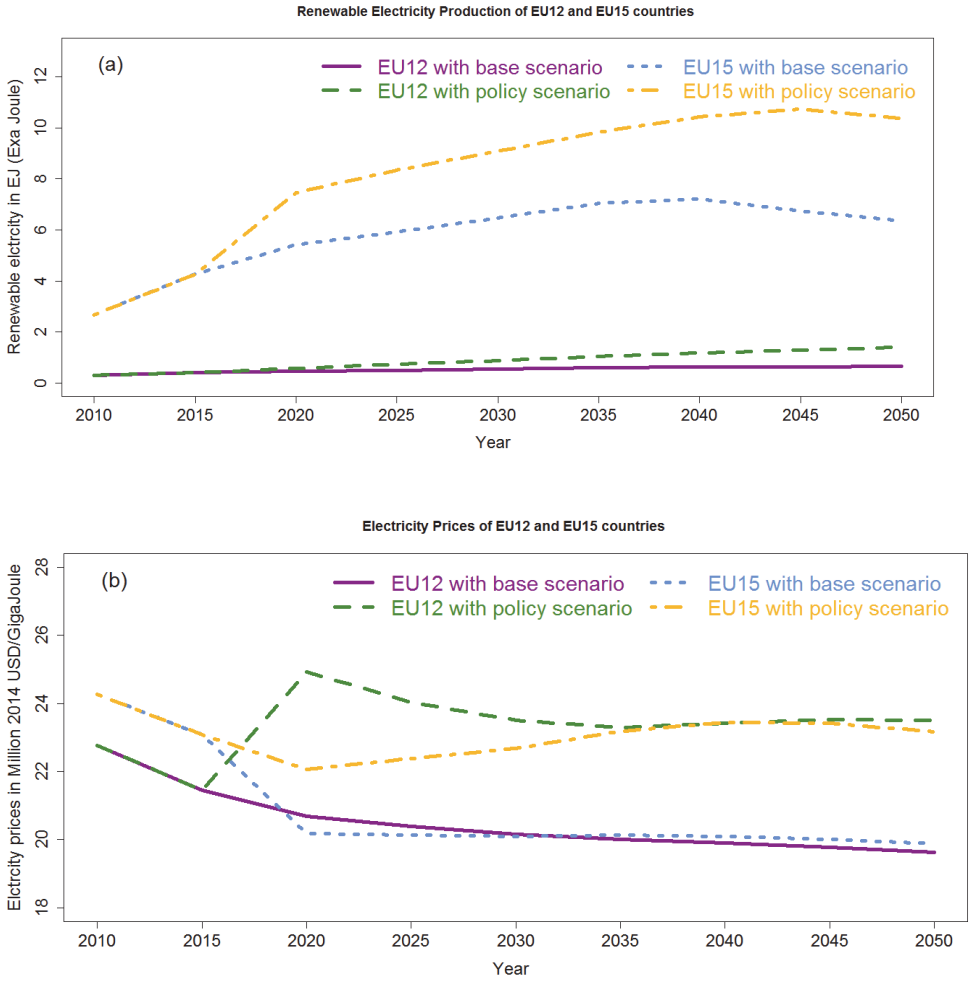


Figure 5.9: (a) Renewable electricity production of EU12 and EU15 countries, (b) Electricity prices of EU12 and EU15 countries for both base scenarios and targeted policy scenarios. Here the model starts updating from 2015 (previous periods are given by default) and we introduced the policy (i.e. 20% reduction by 2020 compare to 2005 level) in 2020. Since the model runs with 5-year time steps, there is a sudden jump from 2015 to 2020.

At the end of the simulation the integrated system produces data for hundreds of variables (which represent different sectors). To demonstrate the effect of the simulated emission reduction policies we selected three variables from EXIOMOD: Household consumption of electricity, Employment in green electricity sector, Import of mining products (including oil and gas). Besides, for comparison we did the simulations for three different cases:
A - when EXIOMOD did not receive data from GCAM,

B - when the integrated system runs with reference or business as usual scenario, and
C - the integrated system runs with targeted policy scenario. The results of the simulations for the selected variables are presented below.

The trend in Household electricity consumption is shown in Figures 5.10 (a) and (b). Each figure consists of three graphs: (1) Electricity consumption predicted using the standalone system, i.e. using system A; (2) Electricity consumption predicted using system B; (3) Electricity consumption predicted using system C. The graphs show that implementing those policy scenarios will result in reduction of household electricity consumption, which is associated with the increase in electricity prices shown in Figure 5.9(b).

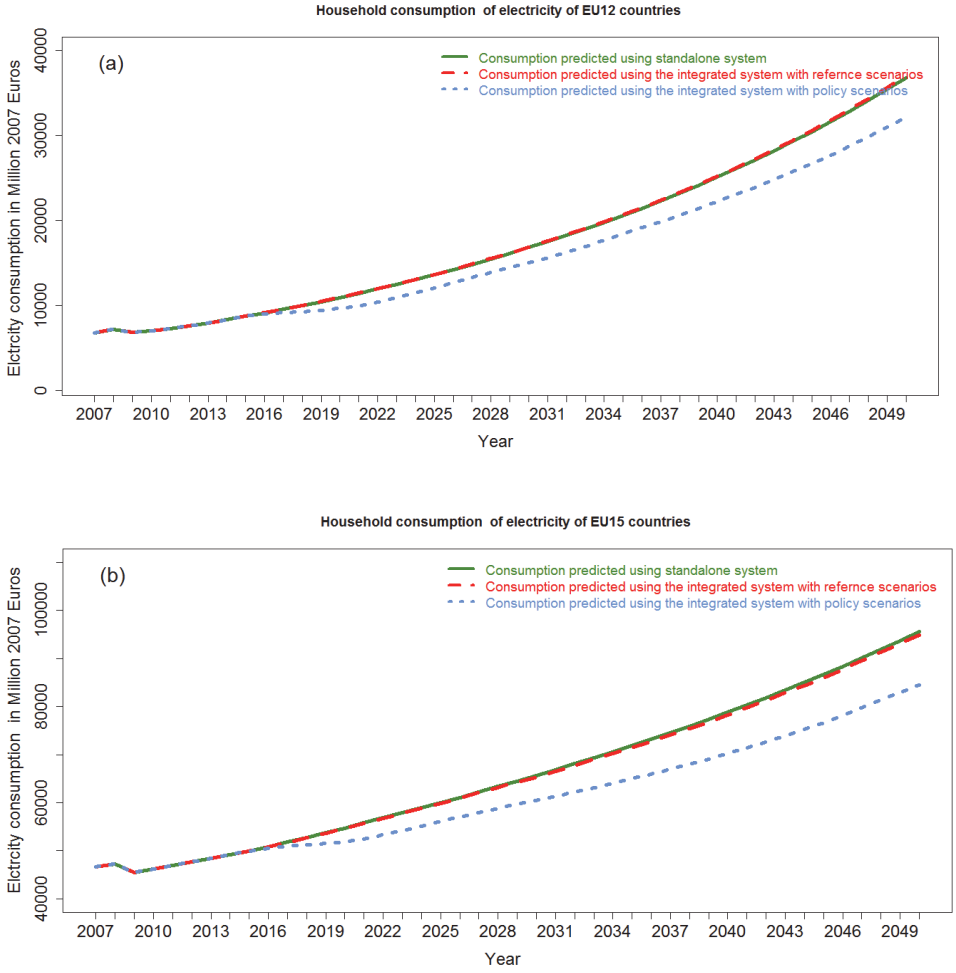


Figure 5.10: Household electricity consumption prediction: using system A, B, and C for (a) EU12 countries, and (b) EU15 countries.

Employment in green electricity sector is the second variable we selected to demonstrate the benefit of building the system-of-systems. As shown in Figure 5.11 implementing those emission reduction scenarios will increase the employment in the green electricity sector for both EU12 and EU15 countries, which is associated with an increase in renewable electricity production (Figure 5.9(a)).

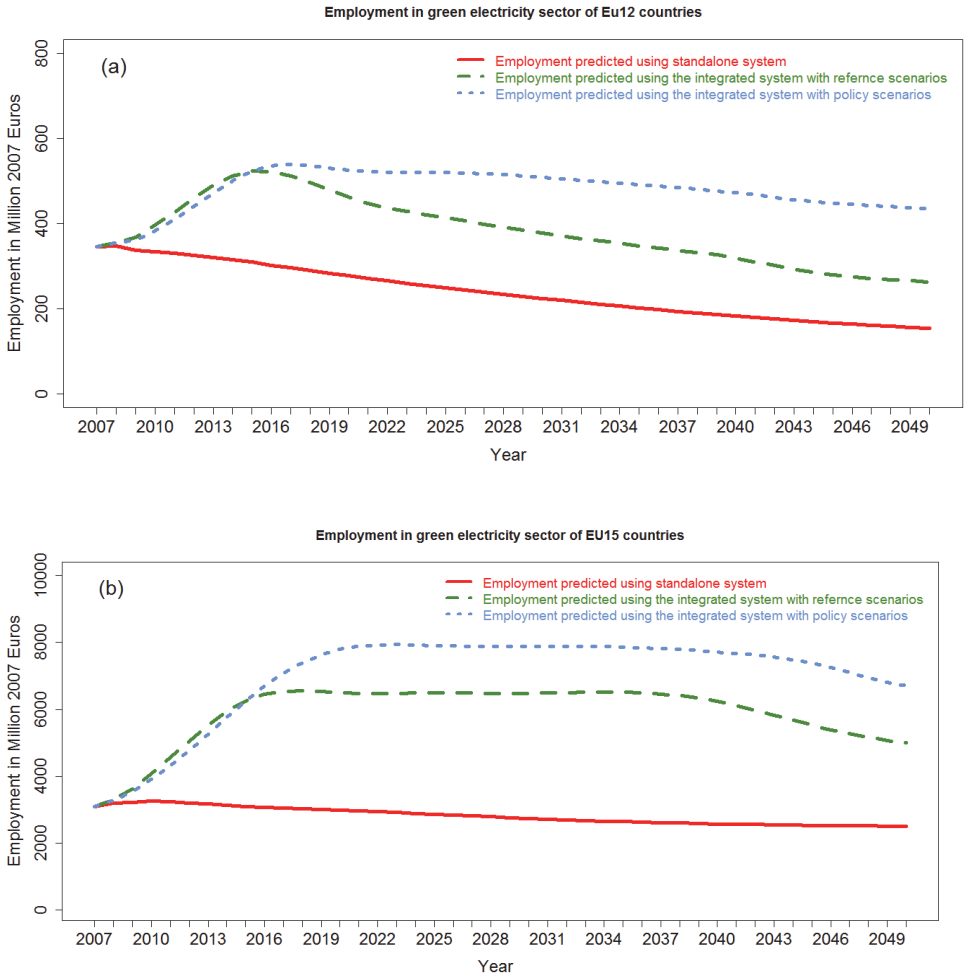


Figure 5.11: Employment in green electricity of (a) EU12 countries, and (b) EU15 countries.

Not all variables show so distinct differences in the three systems analyzed. For example, if we look at Import of mining products, which includes coal, oil, gas, metals and non-metallic mineral products, in all three systems the results are quite similar (Figure 5.12). This is because EXIOMOD do not use separate variable to represent import of coal, which is one of the main

resources used for electricity production and which could be affected by an increase in green electricity production. Considering coal together with other mining products will have its effect on the model output. The reduction in the imports of coal could be compensated by an increase in the imports of other mining product such as oil and gas (as energy sources used for direct heating and transportation) and metals and non-metallic minerals. In case of policy analysis where coal import is of particular interest, the choice of model with sufficient level of detail therefore becomes crucial.

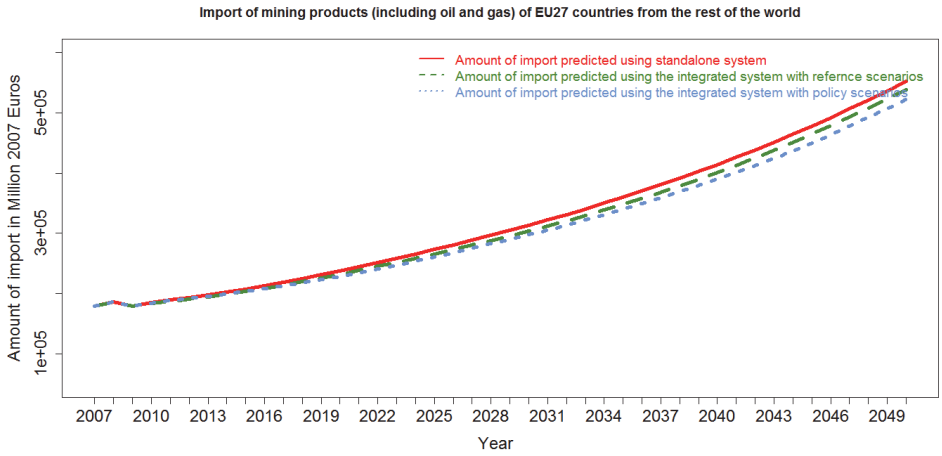


Figure 5.12: Import of mining products in EU27 countries.

In general, linking these standalone models has given us a chance to streamline the ad-hoc input data producing process (i.e. electricity generation mix for EXIOMOD). It also provides the flexibility in implementing targeted scenarios. Impact assessment of climate change policies for specific sectors can be handled using a standalone system, for example, we can use the CGE model to analyze impacts of climate change policy on energy and economics (Ruamsuke et al., 2015), or on the transportation sector (Solaymani et al., 2015). With integration we can explore more complex situations, such as the effect of implemented scenarios across sectors presented in individual standalone models. Each individual model can be set to a number of different system states by setting the various variables to a range of values, and linking such models with other models (which themselves have several system states) will significantly increase the number of system states. For example, in our case the two variables from GCAM (Et and Ep) can possibly affect all the 163 variables (sectors) of EXIOMOD. Stoorvogel (1995) pointed out that by linking standalone models they could evaluate an 'infinite number of scenarios', which we can consider as an important 'by-product' of integration.

The integration methodology we described in this paper can be applied in interdisciplinary studies. The steps we followed in the pre-integration assessment and in conceptualizing the integrated system can be replicated by other similar projects, including for soft-linking of models (Deane et al., 2012) which is popular in linking climate and energy models. We observed that identifying the appropriate models and formulating the interaction between them is difficult and time consuming. This is mainly because model owners know more about their models, and creating common understanding about participating models requires several iterations and much discussion. We saw that, besides having clearly defined integration objectives, a list of scenarios together with the corresponding indicator variables (to be tested by the integrated system) could help to keep pre-integration assessment discussions focused and productive.

We also found that the technical interoperability aspect of integration could be standardized and reused. In our specific case wrapping models with web services was the standard for technical interoperability. The web service based wrappers developed to link models can be reused in other cases. The wrappers we developed can be accessed from any platform, and they can also be reused with non-web service based integration frameworks. Besides, wrapper classes developed to encapsulate NetLogo and GAMS based models can be easily customized in wrapping new models. Additional NetLogo or GAMS based models can be incorporated into our system without reinventing the wrappers. It requires only modifying the model name, input-output variable names, and the path in which the base models are residing.

However, our case study demonstrated that semantic mediation and data conversion tasks might require case specific functionalities. For example, if we link GCAM to yet another model then data processing may be different because of different combinations of input-output variables to be exchanged with the new model, different units used in those variables, and the different spatial and temporal scales assumed for them. This is certainly challenging for building generic semantic mediation and data conversion tools. On the other hand, we did see that some generic functionality to handle some parts of semantic mediation and dataset conversion can be built. For example, standardized unit conversion functionality for SI and derived units can be provided using available units ontologies. Similarly, automatic regridding functionalities in CSDMS (Syvitski et al., 2004), or ESMF (DeLuca et al., 2012), are good examples of advances toward standardized data mediation.

There are several advantages of web-based modeling, such as: ease of use and collaboration, simple licensing, deployment, reuse, access control, versioning, customization and maintenance (Byrne et al., 2010). On top of this with the web service based approach we can incorporate new models,

modify existing models, or remove a member model without disturbing the existing integrated system. This is because in web service based integration a member component has no knowledge about the existence of the other components. Communication between component models is through the integration layer. This gives us the opportunity to create an extendable model integration platform, which can evolve through time. The downside of web service based approach is that it requires high-speed Internet access and can be quite slow when large amounts of data are exchanged repeatedly. However, the greater the processing time of the participating models, the less the communication over-head (Wainer et al., 2008).

Any modeling is accompanied with uncertainty due to approximations used, ignored or misrepresented processes in conceptualization of the model, uncertainties in data, or inability to accurately quantify the input parameters of a model (Shrestha, 2009). Uncertainty is inevitable (Hall, 2003) due to the nature of the task. Of course, as more models are coupled and the higher the complexity of the overall system of systems we build, the more parameters of the system we have and the higher the overall uncertainty will be. Integration of models is likely to propagate uncertainty throughout the model chain (Dubois et al., 2013). There are certain new parameters that come just from the coupling algorithms used. Belete and Voinov (2016) have explored the sensitivity of the model integration to various combinations of time-stepping chosen in component models, as well as to the differences in the functions and the numeric methods assumed. The acceptance of model output, especially in complex systems such as the one we described above will be always contingent upon proper identifying and quantifying of the associated uncertainty. In our case this means quantifying uncertainty in the three component models as well as tracking how it will be propagating through the integrated system. However, this requires a separate extensive treatment, which at this point this goes beyond the scope of the paper.

5.6 Conclusions

The emphasis of this paper was on integration methodology and practices, and not on the analysis of particular climate mitigation policy scenarios. We demonstrated how models which are functioning at different levels, from global, to household, can be linked together to function as a system-of-systems. We described a pretty lengthy pre-integration process that led to some joint decisions about the synergies that various models can generate by feeding information from one model to another and back. The integration technique that was developed is quite general and can accommodate all sorts of data passing between models. If other variables are chosen for data exchange (say, feeding end-use demand into EXIOMOD), it will require the same type of steps as the ones we describe to make this happen (data

mediation, unit conversion, etc.) and not much (if any) additional programming. By linking the three models we were able to create a framework that could help to investigate the energy and economy aspects of climate change, which could be hardly accomplished using stand-alone model components. Despite some clear promises of future reuse of the components we have developed in other system-of-systems designs and other integration schemes, we saw that there are still big concerns about 'automating' and 'streamlining' model integration (Voinov, Shugart, 2013). It remains very unlikely that productive integration can be achieved without the more qualitative pre-integration assessment process, which requires very active involvement of stakeholders/modelers for scoping the objective of integration, formulating scenarios, reviewing and identifying the candidate models, and conceptualizing the envisaged system. The pre-integration assessment phase can take significant percentage of the project life-time with a number of iterations. Moreover, failing to identify mismatches between models at this stage could result in a big waste of resources and delays for the project. We also demonstrated that web service based approach can be used to link heterogeneous models which are developed using quite different modeling languages and paradigms. The web service wrappers we developed can be reused by other integration frameworks, also in non-service based integration frameworks. Our approach can be applied in other similar projects, which require to link models, which operate at different temporal and spatial scales.

Chapter 6

Synthesis

6.1 Introduction

Most socio-environmental problems are wicked problems since there is always more than one solution to the problems and there is no idealized end state to arrive at. Engaging stakeholders is one essential step in tackling such problems. Modeling has been also recognized as an important tool for analyzing such problems. The challenge then is to make potentially quite complex and sophisticated models accessible for stakeholders, while at the same time providing means for stakeholders to have a say in what models do and how they do it. We see model integration as a way to explore the different aspects of the system at stake and improve our understanding of it. The assumption is that it makes more sense to use existing well developed and tested models as building blocks rather than build the whole system model each time from scratch. This requires transparency and flexibility of the model integration process, allowing stakeholders to play a more significant role in deciding what modules are to be linked, how they should be treated, and what scenarios should be analyzed. By switching various modules on and off they can then also test the overall system sensitivity, both parametrically and structurally. We find that one way to remove technical and accessibility constraints and facilitate stakeholder participation is by presenting the models as web applications, making them available through standard web browsers. We performed this research with the objective to develop a methodology and propose software design that transforms independently developed models into web-based interoperable components and further links them into integrated models that can represent complex socio-environmental systems.

This research was conducted in five blocks, and in this chapter we discuss how our findings contribute to the global effort of improving integrated modeling for analyzing complex socio-environmental problems. Section 6.2 presents a brief summary of the main findings of the research in accordance to the specific objectives and section 6.3 provides answers to research questions. Section 6.4 discusses the main contributions of the research described in this dissertation and Section 6.5 provides a list of recommendations for further work.

6.2 Summary of results and their inter-relationships

(1) Methodology to facilitate the model integration process from requirements identification to system testing.

In Chapter 2 we did a literature review on integration of models and integration frameworks. The objective of the review was to improve our

understanding of the model integration process and to come up with strategies and best practices that can improve the different stages of the integration process. From our review of literature and also from our experience we found that integration of models requires higher level of software engineering and semantic processing than the conventional modeling process. Integration of models cannot be fully handled by using only the modeling or software development life cycle. Due to this we classified the model integration processes into five phases: pre-integration assessment, preparing models for integration, orchestration of participating models during simulation, data interoperation, and testing. The relationship between these development phases is iterative and incremental (Figure 2.1). We highlighted key strategies, features, standards, and practices that can be employed during each of these phases. Besides, we suggested techniques that can improve discovery, reusability, and ease of use of models in general. We also applied the model integration methodology in a case study (discussed on Chapter 5) and we confirmed that following the integration phases as well as some of best practices identified can facilitate the model integration process. Based on the proposed integration methodology, first, we defined scenarios to be simulated by the envisaged integrated system. Then, we applied the pre-integration assessment questions (of the methodology) in selecting models and in conceptualizing the integrated system. This helped us to keep the pre-integration assessment meetings focused and to minimize the back and forth iterations with participants, which commonly require several meetings and delay the process.

(2) Design a web based model integration framework that links independently developed multidisciplinary models into an integrated system.

Simulating complex socio-environmental systems may require linking models from different disciplines and developed using different assumptions, semantics, and tools. The models can also be implemented remotely and can be hosted on different hardware and software platforms. In Chapter 3, we used these requirements as design criteria for a model integration framework and we developed the architecture (Fig 3.1) and prototype of the Distributed Model Integration Framework (DMIF) based on the principles of distributed computing and service oriented software development approaches. We used web services to wrap legacy models so that they can be accessed remotely and converted into interoperable components. By using the web service based approach we transformed several stand-alone models developed using such programming languages as GAMS, NetLogo, R, and C++ into interoperable components. We used C# and Java to develop wrappers for these models. The C# based wrapper web services are hosted on Microsoft Internet Information Service web server and the Java based services are

hosted on Oracle Glassfish web server, while the integrated system appears to users as a single coherent system.

We also investigated runtime integration of models (Fig 3.5, 3.6), i.e. an integration method in which users can access and integrate models 'on the fly' using a graphical user interface. As part of this research, we developed a prototype of a generic interface for runtime access and integration of web service based models. Our first step was to develop an interface that can execute web service based models in runtime. The interface requires the URL of the Web Service Description (WSDL) as an input, and then it dynamically extracts the available functions and input-output details about the service by using the information provided. With the interface, the user can also provide inputs that are required for the model and then execute it. The second step was to upgrade this utility so that we can link web service based models in runtime. An Integration Builder utility was developed to provide a GUI for users to define the sequence of execution of participating models and the data exchange pattern between them. These kinds of generic interfaces can improve stakeholder engagement by minimizing the need of writing custom code to link various models. Besides, it also resolves issues of access rights to model source code and the need for technical knowledge and skills to deploy models on end user computers.

(3) Method to reconcile implicit semantic relations in integration of models.

Integration of models requires establishing data exchange between participating component models. Semantic mediation is a mechanism to ensure the consistency of the data to be exchanged by checking that the contextual meaning is understood, correctly mapped and, if necessary, translated. In Chapter 3, we developed semantic matching algorithms for input-output text data and attribute names used in models and also a dynamic unit conversion technique between participating models. We used a freely available lexical database called WordNet and an ontology called QUDT as a backend for some of the semantic mediation functions. The inclusion of the lexical database enabled the semantic matching algorithm to consider hierarchical semantic relationships between such concepts as synonym, hypernym, and hyponym. Similarly, the ontology served to provide dynamic unit conversion between SI, Derived, SI-Derived and None-SI units.

As a proof of concept for sematic mediation techniques developed we used the integration framework for a case study of matching text-based data of freely available web service based models. We selected an Airports information web service that provides a list of airports in a country and a Global Weather service that provides weather information of a given city. The objective of the linking was to generate an enhanced list of airports in a given

country, which includes current weather information such as visibility, sky conditions, pressure, temperature, relative humidity, and dew point. The two web services were effectively linked by applying the semantic mediation techniques. The semantic matching algorithms were also used for searching of components within the Community of Surface Dynamics Modelers (CSDMS) model repository. The results (Fig 3.2 and Table 3.2) indicated that the algorithms could effectively automate some of the semantic mediation tasks in integrating models. However, users can interactively improve results by setting the margins for 'valid' matching outputs. The results (Table 3.3) reassure that standardization of metadata of models will minimize the need for semantic processing in linking models, and we believe that standardization is the way forward for automating semantic mediation.

(4) Sensitivity analysis to investigate how integration results are affected by time steps, numeric integration methods, and functional responses assumed in the component models.

Models that we are considering for integration may operate at different time steps and they may also represent similar processes using different mathematical expressions or functions. Executing a model for a given time step requires computing the model output at specified time steps, which is done using numeric integration methods. However, different numeric integration methods have different precision and our choice of the numeric integration method may affect the integration output. In general, the precision of the integration output depends on the arrangements chosen for these parameters.

In chapter 4, to understand how integration parameters can impact the overall results we used a demo case study where we split the classical predator-prey model into two separate component models, one for prey, another for the predator populations. Then, we linked the two components using the integration framework and we conducted multiple runs for different time steps, numeric integration methods, and mathematical expressions used in components. The results (Fig 4.3, 4.4, and Table 4.1) indicated that integration of models with different time steps can be (1) highly sensitive to the size of the time steps chosen; (2) quite sensitive to the choice of the component where the bigger time step is assumed, and (3) relatively less sensitive to the difference between the time steps in component models. We then used different combinations of Euler and Runge-Kutta numeric integration methods in components. As expected, usage of Runge-Kutta method in even one of the coupled models can improve overall accuracy, but usage of different methods in component models is not symmetrical (Table 4.2). We also found that the results are asymmetrical to the type of trophic functions assumed in component models. Overall, sensitivity analysis is very

essential to understand the overall system performance, and model integration opens new dimensions for sensitivity analysis, since additional important parameters are brought into the process.

(5) A case study of integrated modeling of complex scenarios, such as climate change mitigation actions.

The literature review in Chapter 2 and the integration framework described in Chapter 3 were foundational inputs for this case study, which was performed in the context of the EU FP 7 project named COMPLEX and aimed at developing advanced modeling tools for low-carbon scenario analysis. In this project we had a 'model space' that consisted of more than a dozen of models (Fig 5.1), representing a wide spectrum of modeling types and paradigms. We used pre-integration assessment methodology (Fig 5.2) and we set a high-level requirement for the system as a tool to explore the effects of implementing United Nations Framework Convention on Climate Change (UNFCCC) policy scenarios in EU27 countries. Through the pre-integration assessment process we identified that an integrated climate-energy-economy system can be constructed by linking three models from the COMPLEX project model repository. The Global Change Assessment Model - GCAM, a Computable General Equilibrium economic model – EXIOMOD, and an agent-based energy market model – NIROO were chosen for integration. The models were developed using C++, GAMS, and NetLogo programming, respectively. We converted them into interoperable components by wrapping them with web services. Input-output data exchanges between the models were identified (Fig 5.3, 5.5, and Table 5.1) and the required data conversion functions were implemented. Finally, an integrated climate-energy-economy system based on the three models was developed.

By using the integrated system we simulated two selected climate change related scenarios. The first one was the business as usual scenario that simulates the situation when there are no climate change policy interventions, and the second one is a policy-based scenario in which UNFCCC policy targets set by different regions and countries are used as inputs for the integrated system. As a result (Fig 5.10, 5.11, 5.12) by linking the three models we were able to simulate the full climate-energy-economy system, which could be hardly accomplished using stand-alone model components. The case study helped us to experiment how models functioning at different levels, from global to household, can be linked together in an integrated system. Our approach can be applied in other similar research, which requires linking models operating on various platforms and at different spatial and temporal scales.

6.3 Answers to research questions and main conclusions

Based on the summary of the results we provided answers to the five main research questions specified in Chapter1:

(1) How to develop a methodology that facilitates the model integration process from requirements identification to system testing?

Model integration process can be facilitated if organized into five phases: pre-integration assessment, preparing models for integration, orchestration, data interoperation, and testing. It is an iterative and incremental process, which starts with some general qualitative considerations, while more complex issues and features are incorporated later through iterations. At the end of each phase it requires validating the output against the objective and requirements and going back to incorporate missing requirements. The various strategies and best practices provided in Chapter 2 can be applied to facilitate the different phases of integration of models.

(2) How to design a web based model integration framework that links independently developed multidisciplinary models into an integrated system?

To meet the current needs of model integration frameworks the design should be organized as a layered structure with separate components for specific tasks of integration. The design should support models developed in any programming language and located anywhere on the Internet. This can be realized by using the principle of distributed systems and service oriented architectural approach of software development. Existing models can be transformed into interoperable components and incorporated into the integrated system by developing wrappers on top of them. Some standardized methods for model wrapping have been proposed to streamline the process. Web services are a good option for developing wrappers since they are language interoperable, platform independent, and they can be accessed over the web. The communication between models can be handled using a centralized technical integration module and then the system gives a sense of a single integrated model. In this design approach addition or removal of a model component does not affect other existing model components and this enables the framework to evolve easily through time. This approach allows keeping computationally intensive simulations on remote servers, and as a result it gives the opportunity to manipulate complex simulations from web interfaces. In addition, it promises the functionality needed to provide light -weight user interfaces that can manipulate models from mobile devices.

(3) How to reconcile implicit semantic relations in integration of models?

Semantic mediation is done based on the contextual information of participating models, and contextual information of models is found from metadata of models. The availability and also clarity of the model metadata will affect the semantic mediation process. Semantic matching algorithms can be used to identify relations between different concepts represented by models. Model metadata provided as annotations (comment or an explanation) for model interfaces can help to automate semantic matching. Applying lexical databases and ontologies in semantic matching process will help to find implicit semantic relations between various concepts. The challenge is that existing lexical databases and ontologies cover only a limited percentage of concepts used by various models. However, lexical databases and ontologies can be applied to automate some of the semantic mediation tasks, such as matching of text data and attribute names of models. The accuracy of semantic matching results can be improved by presenting semantic matching results to users so that they will make final decisions on matching output. This will help to safeguard us from unintended interpretation of concepts and misuse of models.

(4) How can integration result be affected by time steps, numeric integration methods, and functional responses assumed in the component models?

There are no generic recommendations for optimal component synchronization but we recommend that sensitivity analysis should be included as an essential part of integrated modeling. This is because integration output is sensitive to the time steps assigned to each of the components and the results that we get by varying the time steps in components is not symmetrical. As a result we should be selective in choosing which component will be using which time step. The same applies to our choice of numeric integration methods: in which of the coupled models we apply higher order numeric methods is also not symmetrical and does matter. Similarly, if the participating models use different mathematical expressions to represent the same concept, sensitivity analysis needs to be done.

(5) How can we apply integrated modeling approach to an actual case study of complex socio-environmental scenarios, such as climate change mitigation actions?

At first pre-integration assessment should be done to identify the participating models and to conceptualize the envisaged integrated system.

The pre-integration assessment should begin by setting a clearly defined objective, integration scenarios, and indicator variables to be simulated using the envisaged integrated system. This keeps the process focused and minimizes the number of iterations in conceptualizing the integrated system. By the end of the pre-integration assessment the workflow, the input-output data exchange between component models, and the mechanisms for dataset conversions should be formulated. Then, the next step is technical implementation of the conceptualized system. Models developed using different programming tools can be converted into interoperable components by wrappers built on top of them. The user interface should present the integrated system in a self-explanatory and flexible way. The simulation output should include appropriate indicator variables, which could be possibly affected as a result of integration.

6.4 Main contributions

The main contributions of the research described in this dissertation are:

- A Methodology that can facilitate the model integration process is provided. Key strategies and best practices for different phases of integration are pointed out. Besides, strategies to improve discoverability, accessibility, and ease of use of models are discussed. The model integration methodology, together with identified key strategies and best practices, can be applied by developers to increase accessibility, reuse, and interoperability of model components and integrated models.
- State of the art design of distributed model integration framework is developed. As a proof of concept the prototype of the distributed integration framework is implemented. Besides, a prototype of generic interfaces for runtime access and integration of web service based models is developed.
- Semantic mediation techniques using a freely available lexical database and ontology are proposed. Algorithms for semantic matching of text-based data and attribute names of models are provided. These semantic mediation techniques can be used in other integration frameworks.
- The different time steps, numeric integration methods, and different formalizations used by component models have an effect on the integration output. This dissertation indicated that sensitivity analysis is important to understand how the integrated system behaves as a result of different combinations of those parameters.
- A case study on linking independently developed models to create an integrated system that could help to investigate the energy and economy aspects of climate change is presented. The step-by-step procedures tested in the case study can be applied in similar model integration endeavors.

6.5 Recommendations for future work

During the realization of this research we encountered a number of issues, which lead to recommendations for further studies.

(1) Artificial intelligent data conversion agents. An artificially intelligent data conversion agent that can perceive the context of the participating component models and then convert data automatically minimizing the need for custom developed code for data conversion. Identification of the context of participating models can be done based on the information from metadata tags of the models. These tags can include attribute names, spatial and temporal scales, units, etc. The semantic matching algorithms together with the application of lexical databases and ontologies described in this dissertation can be explored further for identification and matching of contexts of participating models. The result from this research will contribute towards the provision of generic model integration framework.

(2) Improving the ease of use of the user interface of integrated modeling frameworks. The user interface is the only means in which users interact with modeling tools. The user interface plays an important role in engaging stakeholders since it affects how users act on the system, and also the way information is presented and perceived. User interface has some learning curve. The improvement in the user interface can be achieved by considering the user cognition processes and mental models of users and modelers in designing the interface. The design should focus more on the users and their tasks than on the underlying implementation technology. In fact, the user interface has to handle complexity from both users and underlying applications. User interface of models for mobile devices is another important issue to be considered in the future. This requires considering the processing power, memory, and various screen sizes of mobile devices on the one hand, and flexibility, completeness, and usability of the user interface on the other.

(3) Identifying and quantifying propagation of uncertainty in integration of models. Commonly models participating in the integration process have associated uncertainties. In linking models, when the output of one model goes as input for the next model, the uncertainty produced by one model will also propagate to the next model. Besides, in many cases data conversion is done based on some assumptions, which could also raise uncertainty. To quantify the level of uncertainty on the integration output requires identifying uncertainty in each component and then computing the combined uncertainty due to the model component itself and utilizing uncertain input data. The computation of the propagation of

uncertainty may depend on the number of component models involved, the type of interaction between them (e.g. one directional vs. two directional and iterative), the variables involved in the data exchange, and the type of mathematical operations in which the models use the input data they received. This will be very important to present the integration output together with the overall level of uncertainty, which will have a significant effect in the acceptance of the integration results.

(4) Further development of the generic interfaces for runtime integration of models. Improvement of the generic interface can be done in various ways. For example, Integration Builder is part of the generic interface that is used to define the workflow of the integration. The workflow definition includes the sequence of execution of participating models, the respective member functions, input-output variables, and data exchange pattern between them. The user can define the workflow at the GUI level and then the system will generate code for the workflow. Currently the code of the workflow is generated using in-house developed syntax (discussed in Chapter 3). However, the workflow encoding can be enriched by adopting existing workflow specification standards. At first this will help to have workflow specification dedicated to linking various types of models. Second, it will enable to reproduce results, share workflows with other systems, and compare the outputs with other systems.

Bibliography

Achananuparp, P., Hu, X., Shen, X., 2008. The evaluation of sentence similarity measures, Data warehousing and knowledge discovery. Springer, pp. 305-316.

Alonso, G., Casati, F., Kuno, H., Machiraju, V., 2004. Web services: Concepts, Architectures and Applications. Springer.

Ammann, P., Offutt, J., 2008. Introduction to software testing. Cambridge University Press.

Arditi, Roger, & Ginzburg, Lev R., 1989. Coupling in predator-prey dynamics: ratio-dependence. Journal of Theoretical Biology, 139(3), 311-326.

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

Armstrong, R., Gannon, D., Geist, A., Keahey, K., Kohn, S., McInnes, L., Parker, S., Smolinski, B., 1999. Toward a common component architecture for high-performance scientific computing, High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on. IEEE, pp. 115-124.

Arnold, T.R., 2013. Procedural knowledge for integrated modelling: towards the modelling playground. Environmental modelling & software 39, 135-148.

Ashley, R., Garvin, S., Pasche, E., Vassilopoulos, A., Zevenbergen, C., 2007. Advances in urban flood management. CRC Press.

Athanasiadis, I.N., Janssen, S., 2008. Semantic mediation for environmental model components integration. Information Technologies in Environmental Engineering 1.

Belete, G.F., Voinov, A., 2014. Integration of Models for Low Carbon Economy, 7th International Congress on Environmental Modelling and Software (iEMSs), San Diego, California, USA.

Belete, G.F., Voinov, A., 2016. Exploring temporal and functional synchronization in integrating models: A sensitivity analysis. Computers & Geosciences 90, 162-171.

Belete, G.F., Voinov, A , Bulavskaya, T, Niamir, L, Dhavala, K, Arto, I, Moghayer, S, and Filatova, T, (in revision). Web service based approach to linking heterogeneous climate-energy-economy models for climate change mitigation analysis. International Journal of Energy. Manuscript in revision.

Belete, G.F, Voinov, A., Holst, N., 2014. An architecture for integration of multidisciplinary models, 7th International Congress on Environmental Modelling and Software (iEMSs). San Diego, California, USA.

Belete, G.F., Voinov, A., Laniak, G.F, 2017. An overview of the model integration process: from pre-integration assessment to testing. Environmental Modelling and Software 87, 49-63.

Bennett, Neil D, Croke, Barry FW, Guariso, Giorgio, Guillaume, Joseph HA, Hamilton, Serena H, Jakeman, Anthony J, . . . Perrin, Charles., 2013. Characterising performance of environmental models. Environmental Modelling & Software, 40, 1-20.

Berner, S., Weber, R., Keller, R.K., 2005. Observations and lessons learned from automated testing, Proceedings of the 27th international conference on Software engineering. ACM, pp. 571-579.

Bertolino, A., 2007. Software testing research: Achievements, challenges, dreams, 2007 Future of Software Engineering. IEEE Computer Society, pp. 85-103.

Bonet, F.J., Pérez-Pérez, R., Benito, B.M., De Albuquerque, F.S., Zamora, R., 2014. Documenting, storing, and executing models in Ecology: A conceptual framework and real implementation in a global change monitoring program. Environmental Modelling & Software 52, 192-199.

Booth, N.L., Everman, E.J., Kuo, I.L., Sprague, L., Murphy, L., 2011. A Web-Based Decision Support System for Assessing Regional Water-Quality Conditions and Management Actions1. JAWRA Journal of the American Water Resources Association 47, 1136-1150.

Brooking, C., Hunter, J., 2013. Providing online access to hydrological model simulations through interactive geospatial animations. Environmental Modelling & Software 43, 163-168.

Bruggeman, J., Bolding, K., 2014. A general framework for aquatic biogeochemical models. Environmental Modelling & Software 61, 249-265.

Buccella, A., Cechich, A., Fillottrani, P., 2009. Ontology-driven geographic information integration: A survey of current approaches. Computers & Geosciences 35, 710-723.

Butterfield, M.L., Pearlman, J.S., Vickroy, S.C., 2008. A system-of-systems engineering GEOSS: Architectural approach. Systems Journal, IEEE 2, 321-332.

Byrne, J., Heavey, C., Byrne, P.J., 2010. A review of Web-based simulation and supporting tools. Simulation modelling practice and theory 18, 253-276.

Campolongo, Francesca, Cariboni, Jessica, & Saltelli, Andrea., 2007. An effective screening design for sensitivity analysis of large models. Environmental modelling & software, 22(10), 1509-1518.

Canhasi, E., 2013. Measuring the sentence level similarity.

Capellán-Pérez, I., González-Eguino, M., Arto, I., Ansuategi, A., Dhavala, K., Patel, P., Markandya, A., 2014. New climate scenario framework implementation in the GCAM integrated assessment model.

Castronova, A.M., Goodall, J.L., Elag, M.M., 2013. Models as web services using the open geospatial consortium (ogc) web processing service (wps) standard. Environmental Modelling & Software 41, 72-83.

Cellier, François E, & Kofman, Ernesto., 2006. Continuous system simulation: Springer.

Ciscar, J.-C., Dowling, P., 2014. Integrated assessment of climate impacts and adaptation in the energy sector. Energy Economics 46, 531-538.

Clemens, S., Dominik, G., Stephan, M., 1998. Component software: beyond object-oriented programming. Addison-Wesley New York.

Cohen, W., Ravikumar, P., Fienberg, S., 2003. A comparison of string metrics for matching names and records, Kdd workshop on data cleaning and object consolidation, pp. 73-78.

COMPLEX, 2016. COMPLEX project. Retrieved from http://owsgip.itc.utwente.nl/projects/complex

Crnkovic, I., Chaudron, M., Larsson, S., 2006. Component-based development process and component lifecycle, Software Engineering Advances, International Conference on. IEEE, pp. 44-44.

da Silva, A., DeLuca, C., Balaji, V., Hill, C., Anderson, J., Boville, B., Collins, N., Craig, T., Cruz, C., Flanigan, D., 2003. The Earth system modeling framework, 3rd NASA Earth Science Technology Conference.

David, O., Ascough, J., Lloyd, W., Green, T., Rojas, K., Leavesley, G., Ahuja, L., 2013. A software engineering perspective on environmental modeling framework design: The Object Modeling System. Environmental Modelling & Software 39, 201-213.

Deane, J., Chiodi, A., Gargiulo, M., Gallachóir, B.P.Ó., 2012. Soft-linking of a power systems model to an energy systems model. Energy 42, 303-312.

Debayan, B., 2011. Component Based Development-Application In Software Engineering. Indian Statistical Institute.

DeLuca, C., Theurich, G., Balaji, V., 2012. The Earth System Modeling Framework, Earth System Modelling-Volume 3. Springer, pp. 43-54.

De Nicola, A., Missikoff, M., Navigli, R., 2009. A software engineering approach to ontology building. Information systems 34, 258-275.

Dubois, G., Schulz, M., Skøien, J., Bastin, L., Peedell, S., 2013. eHabitat, a multi-purpose Web Processing Service for ecological modeling. Environmental Modelling & Software 41, 123-133.

Edmonds, J., Wise, M., Pitcher, H., Richels, R., Wigley, T., Maccracken, C., 1997. An integrated assessment of climate change and the accelerated introduction of advanced energy technologies-an application of MiniCAM 1.0. Mitigation and adaptation strategies for global change 1, 311-339.

Erl, T., 2008a. SOA design patterns. Pearson Education.

Erl, T., 2008b. Soa: principles of service design. Prentice Hall Upper Saddle River.

Erl, T., Karmarkar, A., Walmsley, P., Haas, H., Yalcinalp, L.U., Liu, K., Orchard, D., Tost, A., Pasley, J., 2009. Web service contract design and versioning for SOA. Prentice Hall.

Filatova, Tatiana, Moghayer, Saeed, Arto, Inaki, Belete, Getachew F., Dhavala, Kishore, Hasselmann, Klaus, Kovalevsky, Dmitry V., Niamir, Leila, Bulavskaya, Tatyana, Voinov, Alexey, 2014. Dynamics of climate-energy-economy systems: development of a methodological framework for an integrated system of models, Available at: http://owsgip.itc.utwente.nl/projects/complex/complex_files/COMPLEX-Report-D5%20-Final.pdf (Accessed: 3rd July 2015).

Fortes, P., Pereira, R., Pereira, A., Seixas, J., 2014. Integrated technological-economic modeling platform for energy and climate policy analysis. Energy 73, 716-730.

Fowler, M., 2003. Who needs an Architect. IEEE SOFTWARE

Fowler, M., 2010. Domain-specific languages. Pearson Education.

Frakes, W.B., Kang, K., 2005. Software reuse research: Status and future. IEEE transactions on Software Engineering, 529-536.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1994. Design patterns: elements of reusable object-oriented software. Pearson Education.

Garrido, J., Zafra, Á., Vázquez, F., 2009. Object oriented modelling and simulation of hydropower plants with run-of-river scheme: a new simulation tool. Simulation Modelling Practice and Theory 17, 1748-1767.

Gavrilovska, A., Kumar, S., Raj, H., Schwan, K., Gupta, V., Nathuji, R., Niranjan, R., Ranadive, A., Saraiya, P., 2007. High-performance hypervisor architectures: Virtualization in hpc systems, Workshop on System-level Virtualization for HPC (HPCVirt). Citeseer.

Geels, F.W., Berkhout, F., van Vuuren, D.P., 2016. Bridging analytical approaches for low-carbon transitions. Nature Climate Change.

Geller, G.N., Melton, F., 2008. Looking forward: Applying an ecological model web to assess impacts of climate change. Biodiversity 9, 79-83.

Geller, G.N., Turner, W., 2007. The model web: a concept for ecological forecasting, Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International. IEEE, pp. 2469-2472.

Gijsbers, P., Gregersen, J., 2005. OpenMI: a glue for model integration, MODSIM 2005 International Congress on Modelling and Simulation, pp. 648-654.

Gillingham, K., Newell, R.G., Pizer, W.A., 2008. Modeling endogenous technological change for climate policy analysis. Energy Economics 30, 2734-2753.

Goecks, J., Nekrutenko, A., Taylor, J., 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome biology 11, 1.

Goff, S.A., Vaughn, M., McKay, S., Lyons, E., Stapleton, A.E., Gessler, D., Matasci, N., Wang, L., Hanlon, M., Lenards, A., 2011. The iPlant collaborative: cyberinfrastructure for plant biology. Frontiers in plant science 2, 34.

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

Goodall, J.L., Saint, K.D., Ercan, M.B., Briley, L.J., Murphy, S., You, H., DeLuca, C., Rood, R.B., 2013. Coupling climate and hydrological models: Interoperability through Web Services. Environmental Modelling & Software 46, 250-259.

Granell, C., Díaz, L., Gould, M., 2010. Service-oriented applications for environmental models: Reusable geospatial services. Environmental Modelling & Software 25, 182-198.

Gregersen, J., Gijsbers, P., Westen, S., 2007. OpenMI: Open modelling interface. Journal of Hydroinformatics 9, 175-191.

Grimm, V., Augusiak, J., Focks, A., Frank, B.M., Gabsi, F., Johnston, A.S., Liu, C., Martin, B.T., Meli, M., Radchuk, V., 2014. Towards better modelling and decision support: documenting model development, testing, and analysis using TRACE. Ecological Modelling 280, 129-139.

Hall, J.W., 2003. Handling uncertainty in the hydroinformatic process. Journal of Hydroinformatics 5, 215-232.

Hamby, DM., 1995. A comparison of sensitivity analysis techniques. Health Physics, 68(2), 195-204.

Hamilton, S.H., ElSawah, S., Guillaume, J.H., Jakeman, A.J., Pierce, S.A., 2015. Integrated assessment and modelling: overview and synthesis of salient dimensions. Environmental Modelling & Software 64, 215-229.

Harris, G., 2002. Integrated assessment and modelling: an essential way of doing science. Environmental Modelling & Software 17, 201-207.

Hill, C., DeLuca, C., Suarez, M., Da Silva, A., 2004. The architecture of the earth system modeling framework. Computing in Science & Engineering 6, 18-28.

Hillyer, C., Bolte, J., van Evert, F., Lamaker, A., 2003. The ModCom modular simulation system. European Journal of Agronomy 18, 333-343.

Hohpe, G., Woolf, B., 2004. Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional.

Holst, N., 2013. A universal simulator for ecological models. Ecological Informatics 13, 70-76.

Holst, N., Belete, G.F., 2015. Domain-specific languages for ecological modelling. Ecological Informatics 27, 26-38.

Holzworth, D.P., Huth, N.I., 2011. Simple software processes and tests improve the reliability and usefulness of a model. Environmental Modelling & Software 26, 510-516.

Holzworth, D.P., Huth, N.I., Zurcher, E.J., Herrmann, N.I., McLean, G., Chenu, K., van Oosterom, E.J., Snow, V., Murphy, C., Moore, A.D., 2014. APSIM–evolution towards a new generation of agricultural systems simulation. Environmental Modelling & Software 62, 327-350.

Holzworth, D.P., Snow, V., Janssen, S., Athanasiadis, I.N., Donatelli, M., Hoogenboom, G., White, J.W., Thorburn, P., 2015. Agricultural production systems modelling and software: current status and future prospects. Environmental Modelling & Software 72, 276-286.

Huhns, M.N., Singh, M.P., 2005. Service-oriented computing: Key concepts and principles. Internet Computing, IEEE 9, 75-81.

ISO/IEC (International Organization for Standardization/International Electrotechnical Commission), 2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. (IEC))ISO/IEC 25010:2011.

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

Janssen, S.J., 2009. Managing the Hydra in integration: developing an integrated assessment tool for agricultural systems. Wageningen Universiteit (Wageningen University).

Janssen, S., Athanasiadis, I.N., Bezlepkina, I., Knapen, R., Li, H., Domínguez, I.P., Rizzoli, A.E., van Ittersum, M.K., 2011. Linking models for assessing agricultural land use change. Computers and Electronics in Agriculture 76, 148-160.

Janssen, S., Ewert, F., Li, H., Athanasiadis, I.N., Wien, J., Thérond, O., Knapen, M., Bezlepkina, I., Alkan-Olsson, J., Rizzoli, A.E., 2009. Defining assessment projects and scenarios for policy support: use of ontology in integrated assessment and modelling. Environmental Modelling & Software 24, 1491-1500.

Keating, B.A., Carberry, P.S., Hammer, G.L., Probert, M.E., Robertson, M.J., Holzworth, D., Huth, N.I., Hargreaves, J.N., Meinke, H., Hochman, Z., 2003. An overview of APSIM, a model designed for farming systems simulation. European Journal of Agronomy 18, 267-288.

Kim, S.H., J. Edmonds, J. Lurz, S. J. Smith, and M. Wise., 2006. "The ObjECTS Framework for Integrated Assessment: Hybrid Modeling of Transportation " Energy Journal (Special Issue #2) pp 51-80.

Knapen, R., Janssen, S., Roosenschoon, O., Verweij, P., De Winter, W., Uiterwijk, M., Wien, J.-E., 2013. Evaluating OpenMI as a model integration platform across disciplines. Environmental Modelling & Software 39, 274-282.

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

Laplante, P.A., 2007. What every engineer should know about software engineering. CRC Press.

Leimbach, M., Jaeger, C., 2005. A modular approach to integrated assessment modeling. Environmental Modeling & Assessment 9, 207-220.

Li, W., Yang, C., Nebert, D., Raskin, R., Houser, P., Wu, H., Li, Z., 2011. Semantic-based web service discovery and chaining for building an Arctic spatial data infrastructure. Computers & Geosciences 37, 1752-1762.

Lotka, A. J., 1956. Elements of Mathematical Biology (p. 465). New York: Dover.

Lucena, A.F., Clarke, L., Schaeffer, R., Szklo, A., Rochedo, P.R., Nogueira, L.P., Daenzer, K., Gurgel, A., Kitous, A., Kober, T., 2016. Climate policy scenarios in Brazil: A multi-model comparison for energy. Energy Economics 56, 564-574.

Luo, L., 2001. Software testing techniques. Institute for software research international Carnegie mellon university Pittsburgh, PA 15232, 19.

Madni, A.M., Sievers, M., 2014. Systems integration: Key perspectives, experiences, and challenges. Systems Engineering 17, 37-51.

Moghayer, S., P. Wissink., T. Filatova, I. Arto, D. V. Kovalevsky, and L. Niamir., 2016. Reporting on the development of the model database of COMPLEX Climate-Energy-Economy System of Models. EU FP7 COMPLEX. Report D5.5, January: 45 p.

Moore, A., Holzworth, D., Herrmann, N., Huth, N., Robertson, M., 2007. The Common Modelling Protocol: A hierarchical framework for simulation of agricultural and environmental systems. Agricultural Systems 95, 37-48.

Moore, R.V., Tindall, C.I., 2005. An overview of the open modelling interface and environment (the OpenMI). Environmental Science & Policy 8, 279-286.

Myers, G.J., Sandler, C., Badgett, T., 2011. The art of software testing. John Wiley & Sons.

Nativi, S., Mazzetti, P., Geller, G.N., 2013. Environmental model access and interoperability: The GEO Model Web initiative. Environmental Modelling & Software 39, 214-228.

Niamir, Leila., Filatova, Tatiana., 2015. Linking Agent-based Energy Market with Computable General Equilibrium Model: an Integrated Approach to Climate-Economy-Energy System. In the 20th WEHIA Conference, Sophia Antipolis, France, 21-23 May. DOI: 10.13140/RG.2.1.3242.0961

Niamir, L.; Filatova, T. 2016a. From Climate Change Awareness to Energy Efficient Behaviour. International Environmental Modelling and Software Society (iEMSs), Toulouse, France.

Niamir, L. and Filatova T. 2016b. Transition to Low-Carbon Economy: Simulating Nonlinearities in the Electiricity Maket, Navarre Region, Spain. Advances in Social Simulation 2015, Advances in Intelligent Systems and Computing 528, DOI 10.0017/9783-319-47253-9_28

Niwattanakul, S., Singthongchai, J., Naenudorn, E., Wanapu, S., 2013. Using of Jaccard coefficient for keywords similarity, Proceedings of the International MultiConference of Engineers and Computer Scientists, p. 6.

O'Neill, B.C., Kriegler, E., Riahi, K., Ebi, K.L., Hallegatte, S., Carter, T.R., Mathur, R., van Vuuren, D.P., 2014. A new scenario framework for climate change research: the concept of shared socioeconomic pathways. Climatic Change 122, 387-400.

Papazoglou, M., 2008. Web services & SOA: principles and technology. Pearson Education.

Peckham, S., 2014. The CSDMS Standard Names: Cross-Domain Naming Conventions for Describing Process Models, Data Sets and Their Associated Variables. International Environmental Modelling and Software Society (iEMSs), San Diego, California, USA.

Peckham, S.D., Goodall, J.L., 2013. Driving plug-and-play models with data from web services: A demonstration of interoperability between CSDMS and CUAHSI-HIS. Computers & Geosciences 53, 154-161.

Peckham, S.D., Hutton, E.W., Norris, B., 2013. A component-based approach to integrated modeling in the geosciences: The design of CSDMS. Computers & Geosciences 53, 3-12.

Pessoa, R.M., Silva, E., van Sinderen, M., Quartel, D.A., Pires, L.F., 2008. Enterprise interoperability with SOA: a survey of service composition approaches, Enterprise Distributed Object Computing Conference Workshops, 2008 12th. IEEE, pp. 238-251.

Rahman, J., Seaton, S., Perraud, J., Hotham, H., Verrelli, D., Coleman, J., 2003. It's TIME for a new environmental modelling framework, MODSIM 2003 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand Inc. Townsville, pp. 1727-1732.

Ramamoorthy, C., Chandra, C., Kim, H., Shim, Y., Vij, V., 1992. Systems integration: problems and approaches, Systems Integration, 1992. ICSI'92., Proceedings of the Second International Conference on. IEEE, pp. 522-529.

Regueiro, M.A., Viqueira, J.R., Taboada, J.A., Cotos, J.M., 2015. Virtual integration of sensor observation data. Computers & Geosciences 81, 12-19.

Rekaby, A., Osama, A., 2012. INTRODUCING INTEGRATED COMPONENT-BASED DEVELOPMENT (ICBD) LIFECYCLE AND MODEL. International Journal of Software Engineering & Applications 3, 87.

Renner, S., 2001. A community of interest approach to data interoperability, Federal database colloquium, p. 2.

Rizzoli, A.E., Donatelli, M., Athanasiadis, I.N., Villa, F., Huber, D., 2008. Semantic links in integrated modelling frameworks. Mathematics and Computers in Simulation 78, 412-423.

Roman, D., Schade, S., Berre, A., Bodsberg, N.R., Langlois, J., 2009. Model as a service (MaaS), AGILE Workshop: Grid Technologies for Geospatial Applications, Hannover, Germany.

Rosen, M., Lublinsky, B., Smith, K.T., Balcer, M.J., 2008. Applied SOA: service-oriented architecture and design strategies. John Wiley & Sons.

Ruamsuke, K., Dhakal, S., Marpaung, C.O., 2015. Energy and economic impacts of the global climate change policy on Southeast Asian countries: a general equilibrium analysis. Energy 81, 446-461.

Saltelli, Andrea, Ratto, Marco, Andres, Terry, Campolongo, Francesca, Cariboni, Jessica, Gatelli, Debora, Tarantola, Stefano., 2008. Global sensitivity analysis: the primer: John Wiley & Sons.

Saltelli, Andrea, Tarantola, Stefano, Campolongo, Francesca, & Ratto, Marco., 2004. Sensitivity analysis in practice: a guide to assessing scientific models: John Wiley & Sons.

Sarawagi, S., Kirpal, A., 2004. Efficient set joins on similarity predicates, Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ACM, pp. 743-754.

Schmolke, A., Thorbek, P., DeAngelis, D.L., Grimm, V., 2010. Ecological models supporting environmental decision making: a strategy for the future. Trends in ecology & evolution 25, 479-486.

Schut, P., Whiteside, A., 2007. OpenGIS web processing service. OGC project document.

Shrestha, D., 2009. Uncertainty analysis in rainfall-runoff modelling: application of machine learning techniques. TU Delft, Delft University of Technology.

Solaymani, S., Kardooni, R., Yusoff, S.B., Kari, F., 2015. The impacts of climate change policies on the transportation sector. Energy 81, 719-728.

SSP, 2012. SSP Database (Shared Socioeconomic Pathways) - Version 1.0. Retrieved from
https://secure.iiasa.ac.at/web-apps/ene/SspDb/dsd?Action=htmlpage&page=about

Stoorvogel, J.J., 1995. Integration of computer-based models and tools to evaluate alternative land-use scenarios as part of an agricultural systems analysis. Agricultural Systems 49, 353-367.

Sutton, M.A., Reis, S., Bahl, K.B., 2009. Reactive nitrogen in agroecosystems: Integration with greenhouse gas interactions. Agriculture, Ecosystems & Environment 133, 135-138.

Svirezhev Yu , M. and Logofet , D.O., 1983. Stability of Biological Communities . Mir Publishers.

Syvitski, J., DeLuca, C., David, O., Peckham, S., Hutton, E., Gooding, J., 2011. Cyber-infrastructure and community environmental modeling. Handbook in Environmental Fluid Dynamics. Taylor and Francis Group, Boca Raton, FL.

Syvitski, J., Paola, C., Slingerland, R., Furbish, D., Wiberg, P., Tucker, G., 2004. Building a community surface dynamics modeling system: rationale and strategy. A Report to the National Science Foundation, Penn State University, State College, 41.

Tanenbaum, A.S., Van Steen, M., 2007. Distributed systems. Prentice-Hall.

Tsai, W.-T., Fan, C., Chen, Y., Paul, R., 2006. A service-oriented modeling and simulation framework for rapid development of distributed applications. Simulation Modelling Practice and Theory 14, 725-739.

UNFCC, 2014. Appendix I - Quantified economy-wide emissions targets for 2020 United Nations Framework Convention on Climate Change. Retrieved from
http://unfccc.int/meetings/copenhagen_dec_2009/items/5264.php

Uschold, M., 2003. Where are the semantics in the semantic web? Ai Magazine 24, 25.

Vaillancourt, K., Alcocer, Y., Bahn, O., Fertel, C., Frenette, E., Garbouj, H., Kanudia, A., Labriet, M., Loulou, R., Marcy, M., 2014. A Canadian 2050 energy outlook: Analysis with the multi-regional model TIMES-Canada. Applied Energy 132, 56-65.

Van Ittersum, M.K., Ewert, F., Heckelei, T., Wery, J., Olsson, J.A., Andersen, E., Bezlepkina, I., Brouwer, F., Donatelli, M., Flichman, G., 2008. Integrated assessment of agricultural systems–A component-based framework for the European Union (SEAMLESS). Agricultural systems 96, 150-165.

Verburg, P.H., Eickhout, B., van Meijl, H., 2008. A multi-scale, multi-model approach for analyzing the future dynamics of European land use. The Annals of Regional Science 42, 57-77.

Verweij, P., Knapen, M., De Winter, W., Wien, J., Te Roller, J., Sieber, S., Jansen, J., 2010. An IT perspective on integrated environmental modelling: The SIAT case. Ecological Modelling 221, 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 & Software 24, 577-587.

Vitolo, C., Elkhatib, Y., Reusser, D., Macleod, C.J., Buytaert, W., 2015. Web technologies for environmental Big Data. Environmental Modelling & Software 63, 185-198.

Voinov, A., 2008. Systems science and modeling for ecological economics: Academic Press.

Voinov, A., Cerco, C., 2010. Model integration and the role of data. Environmental Modelling & Software 25, 965-969.

Voinov, A., Kolagani, N., McCall, M.K., Glynn, P.D., Kragt, M.E., Ostermann, F.O., Pierce, S.A., Ramu, P., 2016. Modelling with stakeholders–next generation. Environmental Modelling & Software 77, 196-220.

Voinov, A., and Shugart, Herman H., 2013. 'Integronsters', Integral and Integrated Modeling. Environmental Modelling & Software 39 (January): 149–158. doi:10.1016/j.envsoft.2012.05.014.

Volterra, V., 1926. Fluctuations in the abundance of a species considered mathematically. Nature, 188, 558–560.

Wainer, G.A., Madhoun, R., Al-Zoubi, K., 2008. Distributed simulation of DEVS and Cell-DEVS models in CD++ using Web-Services. Simulation Modelling Practice and Theory 16, 1266-1292.

Wainwright, H. M., S. Finsterle, Y. Jung, Q. Zhou and J. T. Birkholzer., 2014. Making sense of global sensitivity analyses. Computers & Geosciences 65: 84-94.

Wang, X., Chan, C.W., Hamilton, H.J., 2002. Design of knowledge-based systems with the ontology-domain-system approach, Proceedings of the 14th international conference on Software engineering and knowledge engineering. ACM, pp. 233-236.

Wachsmuth, J., Blohm, A., Gößling-Reisemann, S., Eickemeier, T., Ruth, M., Gasper, R., Stührmann, S., 2013. How will renewable power generation be affected by climate change? The case of a Metropolitan Region in Northwest Germany. Energy 58, 192-201.

Whelan, G., Kim, K., Pelton, M.A., Castleton, K.J., Laniak, G.F., Wolfe, K., Parmar, R., Babendreier, J., Galvin, M., 2014. Design of a component-based integrated environmental modeling framework. Environmental Modelling & Software 55, 1-24.

Wise, M., Calvin, K., Thomson, A., Clarke, L., Sands, R., Smith, S., Janetos, A., Edmonds, J., 2009. The implications of limiting CO2 concentrations for agriculture, land-use change emissions. and bioenergy. technical report.[PNNL-17943].

Wolstencroft, K., Owen, S., Krebs, O., Nguyen, Q., Stanford, N.J., Golebiewski, M., Weidemann, A., Bittkowski, M., An, L., Shockley, D., 2015. SEEK: a systems biology data and model management platform. BMC systems biology 9, 33.

Woodside, M., Franks, G., Petriu, D.C., 2007. The future of software performance engineering, Future of Software Engineering, 2007. FOSE'07. IEEE, pp. 171-187.

Yang, C., Chen, N., Di, L., 2012. RESTFul based heterogeneous Geoprocessing workflow interoperation for Sensor Web Service. Computers & Geosciences 47, 102-110.

Yu, L., 2011. A developer's guide to the semantic Web. Springer Science & Business Media.

Zhao, P., Di, L., Yu, G., 2012. Building asynchronous geospatial processing workflows with web services. Computers & Geosciences 39, 34-41.

Summary

Most socio-environmental problems are wicked problems that are difficult to solve because of incomplete, contradictory, and changing requirements and information about the systems. Engaging stakeholders in the problem solving process is one essential way to address such problems. Modeling has been also recognized as an important tool for such problems. The challenge then is to make potentially quite complex and sophisticated models accessible for stakeholders. We see model integration as an option to explore the complexity continuum and improve our understanding. The assumption is that it makes more sense to use existing well developed and tested models as building blocks rather than build the whole system each time from scratch. This requires transparency and flexibility of the model integration process, allowing stakeholders to play a more significant role in deciding what modules are to be linked, how they should be treated, and what scenarios should be analyzed. By switching various modules on and off they can then also test the overall system sensitivity, both parametrically and structurally. We find that one way to remove technical and accessibility constraints and facilitate stakeholder participation is by presenting the models as web applications, making them available through standard web browsers. We performed this research with the objective to develop a methodology and software design that enables to transform independently developed models into web-based interoperable components and further link them into integrated models that can represent complex socio-environmental systems. The research was conducted in five blocks that are described as follows:

(1) Develop a methodology that facilitates the model integration process from requirements identification to system testing.

We did a literature review to improve our understanding of the model integration process, and to design better strategies of integrated modeling. Integration of models is an iterative and incremental process that requires higher level of software engineering and semantic processing than conventional modeling. We classified the model integration processes into five phases: pre-integration assessment, preparation of models for integration, orchestration of models during simulation, data interoperability, and testing. We highlighted key strategies, features, standards, and practices that can be employed during integration of models across disciplines, and suggested techniques that can improve discovery, reusability and ease of use of integrated modeling frameworks.

(2) Design a web based model integration framework that links independently developed multidisciplinary models into an integrated system.

Simulating complex socio-environmental systems may require linking models from different disciplines and developed using different assumptions, semantics, and tools. The models can also be located remotely and can be hosted on different platforms. We developed the design and prototype of the Distributed Model Integration Framework (DMIF) that can address these requirements by using distributed computing and service-oriented software development approaches. We used web services to wrap legacy models so that they can be accessed with a regular web browser and to convert them into interoperable components. In this approach the users do not need to install any additional software and can explore modules as stand-alone components, or build chains of modules by connecting outputs from one module to inputs of other modules. Data sets and user scenarios can be also packaged as web services and made available for integration. We also investigated runtime integration of models, i.e. an integration method in which users can access and integrate models 'on the fly' using a graphical user interface. Prototypes of generic interfaces for runtime access and integration of web service based models are developed.

(3) Method to reconcile implicit semantic relations in integration of models.

Integration of models requires establishing data exchange between participating component models. Semantic mediation is a mechanism to ensure the consistency of exchanged data by checking that the contextual meaning is understood, correctly mapped and, if necessary, translated. In this research we developed semantic matching algorithms for input-output text data and attribute names of models and also a dynamic unit conversion feature. We used freely available lexical database called WordNet and ontology called QUDT as a backend for some of the semantic mediation functions. The inclusion of the lexical database enabled the semantic matching algorithm to consider hierarchical semantic relationships between such concepts as synonym, hypernym, and hyponym. Similarly, the ontology provided dynamic unit conversion between SI, Derived, SI-Derived and None-SI units. The semantic matching algorithms were used in a case study of matching text-based data and also for searching of components within the Community of Surface Dynamics Modelers (CSDMS) model repository. The results indicated that the algorithms can effectively automate some of the semantic mediation tasks in integrating models. However, users can interactively improve results by setting the margins for 'valid' matching outputs.

(4) Sensitivity analysis to investigate how integration results are affected by time steps, numeric integration methods, and functional responses assumed in the component models.

To understand how integration arrangements can impact the overall results we used a demo case study where we split the classical predator-prey model into two separate component models. Then, we integrated the two components and conducted multiple runs for different time steps, numeric integration methods, and mathematical expressions used in components. The results indicated that integration of models with different time steps can be (1) highly sensitive to the size of the time steps chosen; (2) quite sensitive to the choice of the component where the bigger time step is assumed, and (3) relatively less sensitive to the difference between the time steps in component models. We then used different combinations of Euler and Runge-Kutta numeric integration methods in components. As expected, usage of Runge-Kutta method in even one of the coupled models can improve overall accuracy, but usage of different methods in component models is not symmetrical. We also found that the results are asymmetrical to the type of trophic function assumed in component models. Overall, sensitivity analysis is clearly very essential for our understanding of the overall system performance.

(5) A case study of integrated modeling of complex scenarios, such as climate change mitigation actions.

The case study was performed in the context of the EU FP7 project COMPLEX. The high-level system requirement for integrated modeling was to simulate the effect of United Nations Framework Convention on Climate Change (UNFCCC) policy scenarios in different sectors of the economy. We did pre-integration assessment and we identified that an integrated climate-energy-economy system can be constructed by linking three models from the COMPLEX project model repository: the Global Change Assessment Model - GCAM, a Computable General Equilibrium economic model – EXIOMOD, and an agent-based energy market model - NIROO. The models were developed using C++, GAMS, and NetLogo programming. We converted them into interoperable components by wrapping them with web services. By using the integrated system we simulated two selected scenarios. The first one is the business as usual scenario that simulates the situation when there are no climate change policy interventions, and the second one is a policy-based scenario in which UNFCCC policy targets set by different regions and countries are fed into the integrated system. The results indicate that by linking the three models we were able to simulate the scenarios in a more detailed way, which could be hardly accomplished using stand-alone model components.

Samenvatting

De meeste sociaal-ecologische problemen zijn lastige problemen die moeilijk op te lossen zijn als gevolg van onvolledige, tegenstrijdige en veranderende eisen en informatie over de systemen. Het betrekken van stakeholders bij het probleemoplossend proces is een essentiële manier om deze problemen aan te pakken. Ook modelleren wordt gezien als een belangrijk instrument voor dergelijke problemen. Het is daarbij een uitdaging om, in potentie vrij complexe en geavanceerde modellen, toegankelijk te maken voor de betrokken partijen. We zien model-integratie als een optie om het complexiteitscontinuüm te verkennen en beter te begrijpen. Verondersteld wordt dat het zinvoller is om de bestaande, goed ontwikkelde en geteste modellen te gebruiken als bouwstenen, in plaats van het hele systeem telkens vanaf nul op te bouwen. Dit vereist transparantie en flexibiliteit van het model-integratieproces, waardoor stakeholders een grotere rol kunnen spelen bij het bepalen welke modules moeten worden gekoppeld, hoe ze moeten worden behandeld en welke scenario's moeten worden geanalyseerd. Door verschillende modules aan en uit te zetten kunnen ze ook de gevoeligheid van het gehele systeem testen, zowel parametrisch als structureel. We zien het aanbieden van modellen als webapplicatie, beschikbaar gemaakt via standaard webbrowsers, als een belangrijke manier om technische- en toegankelijkheidsbeperkingen te minimaliseren en deelname van betrokken partijen te faciliteren. We hebben dit onderzoek uitgevoerd met als doel een methodologie en software-ontwerp te ontwikkelen, dat in staat is om zelfstandig ontwikkelde modellen om te zetten in web-based inter-operationele componenten en ze nog beter te laten aansluiten op geïntegreerde modellen die complexe sociaal-ecologische systemen vertegenwoordigen. Het onderzoek werd uitgevoerd in vijf blokken die als volgt worden omschreven:

(1) Ontwikkel een methodiek, die het model-integratieproces faciliteert vanaf het identificeren van de vereisten tot en met het testen van het systeem.

We hebben een literatuurstudie gedaan om het model-integratieproces beter te kunnen begrijpen, en om betere strategieën te ontwikkelen van geïntegreerde modellen. Integratie van modellen is een iteratief en incrementeel proces, dat een hoger niveau van software engineering en semantische verwerking vereist dan conventionele modellen. We hebben de model-integratieprocessen ingedeeld in vijf fasen: pre-integratie evaluatie, klaarmaken van de modellen voor integratie, bijstellen van de modellen tijdens de simulatie, data interoperabiliteit en testen. We hebben de belangrijkste strategieën, kenmerken, normen en praktijken die worden toegepast tijdens de integratie van modellen in verschillende disciplines uitgelicht, en we hebben technieken voorgesteld die de ontdekking, het

hergebruik en het gebruiksgemak van een geïntegreerd modelraamwerk kunnen verbeteren.

(2) Ontwerp een web-based model-integratie raamwerk dat onafhankelijk ontwikkelde multidisciplinaire modellen verbindt tot een geïntegreerd systeem.

Voor het simuleren van complexe sociaal-ecologische systemen is het misschien vereist dat modellen uit verschillende disciplines en ontwikkeld met behulp van verschillende aannames, semantiek, en gereedschappen worden gekoppeld. De modellen kunnen ook op afstand worden geplaatst en kunnen worden gehost vanaf verschillende platformen. We hebben het ontwerp en het prototype van de Distributed Model Integration Framework (DMIF) ontwikkeld, dat deze vereisten door het gebruik van "distributed computing" en een service-georiënteerde software-ontwikkelingsbenadering kunnen aanpakken. We hebben web services gebruikt om bestaande modellen in te pakken zodat ze kunnen worden benaderd met een gewone webbrowser en om ze om te zetten in compatibele componenten. Bij deze benadering hoeven gebruikers geen extra software te installeren en kunnen ze modules verkennen als stand-alone componenten of ze kunnen ketens van modules bouwen door uitgangen van de ene module met ingangen van andere modules te verbinden. Datasets en gebruikersscenario's kunnen ook worden verpakt als webservices en beschikbaar worden gesteld voor integratie. We hebben ook de runtime integratie van modellen onderzocht, d.w.z. een integratiemethode waarbij gebruikers toegang hebben tot modellen en de modellen 'on the fly' kunnen integreren door het gebruik van een grafische gebruikersinterface. Er zijn prototypes van generieke interfaces voor runtime-toegang en integratie van webservice gebaseerde modellen ontwikkeld.

(3) Methode om impliciete semantische relaties in de integratie van modellen met elkaar in overeenstemming te brengen.

Integratie van modellen vereist dat het uitwisselen van gegevens tussen de deelnemende component-modellen wordt vastgesteld. Semantische mediatie is een mechanisme om de consistentie van de uitgewisselde gegevens te waarborgen door te controleren of de contextuele betekenis wordt begrepen, op de juiste wijze in kaart is gebracht en zo nodig wordt vertaald. In dit onderzoek hebben we semantische matching algoritmes voor input-output tekstdata ontwikkeld en attribuutnamen van modellen en ook een dynamische eenheidsconversiefunctie toegekend. We hebben een vrij beschikbare lexicale database genaamd WordNet en ontologie genaamd QUD gebruikt als sluitstuk voor sommige van de semantische mediatiefuncties. Het inpassen van de lexicale databank maakte het voor het semantische

matching-algoritme mogelijk om hiërarchisch semantische relaties tussen concepten als synoniem, hyperniem en hyponiem mee te nemen in de beschouwing. Op dezelfde manier voorziet ontologie in conversie van een dynamische eenheid tussen SI, Derived, SI-derived en None SI-eenheden. De semantische matching algoritmen werden gebruikt in een case studie om tekstdata te matchen en ook voor het zoeken naar onderdelen binnen de Community van Surface Dynamics Modelers (CSDMS) model repository. De resultaten hebben aangetoond dat algoritmen enkele van de semantische mediatietaken effectief kunnen automatiseren bij het integreren van modellen. Gebruikers kunnen echter interactief de resultaten verbeteren door het instellen van marges voor 'geldige' matching resultaten.

(4) Gevoeligheidsanalyse om te onderzoeken hoe integratieresultaten worden beïnvloed door tijdstappen, numerieke integratie en functionele reacties verondersteld in de componentmodellen.

Om te begrijpen hoe integratieclassificaties de algehele resultaten kunnen beïnvloeden hebben we een demo casestudie gebruikt, waarin we het klassieke roofdier-prooi model in twee afzonderlijke componenten hebben opgedeeld. Daarna hebben we de twee componenten geïntegreerd en hebben we meerdere runs uitgevoerd voor verschillende tijdstappen, numerieke integratie methoden en wiskundige tekens gebruikt in componenten. De resultaten hebben aangetoond dat de integratie van modellen met verschillende stappen in tijd (1) zeer gevoelig kunnen zijn voor de grootte van de gekozen tijdstappen; (2) vrij gevoelig voor de keuze van de component waarbij wordt uitgegaan van de grotere tijdstap; en (3) relatief minder gevoelig voor het verschil tussen de tijdstappen in componentmodellen. Vervolgens hebben we verschillende combinaties van Euler en Runge-Kutta numerieke integratiemethoden gebruikt in de componenten. Zoals verwacht, kan het gebruik van de Runge-Kutta methode in slechts één van de gekoppelde modellen de algehele nauwkeurigheid verbeteren, maar het gebruik van verschillende methoden in componentmodellen is niet symmetrisch. We hebben ook vastgesteld dat de resultaten asymmetrisch zijn aan het trofische functietype, verondersteld in componentmodellen. Algemeen kan worden gesteld dat een gevoeligheidsanalyse duidelijk zeer essentieel is om de algehele prestatie van het systeem te kunnen begrijpen.

(5) Een case studie van geïntegreerde modellering van complexe scenario's, zoals acties voor het tegengaan van klimaatverandering.

De case studie werd uitgevoerd in het kader van het EU FP7 project COMPLEX. De vereiste voor een high-level systeem voor geïntegreerde modellering was om het effect van de beleidsscenario’s van de United Nations

Framework Convention on Climate Change (UNFCCC) op de verschillende sectoren in de economie te simuleren. We hebben een pre-integratie evaluatie gedaan en vastgesteld dat een geïntegreerd klimaat-energie-economie-systeem kan worden geconstrueerd door het koppelen van drie modellen uit de Complex project model repository: de Global Change Assessment Model (GCAM), een Computable General Equilibrium economisch model (EXIOMOD), en een agent-based energiemarktmodel (NIROO). De modellen werden ontwikkeld met behulp van C ++, GAMS, en NetLogo programmering. We hebben ze omgezet in compatibele componenten door ze in te pakken met web services. Door het geïntegreerde systeem te gebruiken hebben we twee geselecteerde scenario's gesimuleerd. De eerste is het gebruikelijke scenario, waarbij een situatie wordt gesimuleerd zonder klimaatbeleidsinterventies. De tweede is een scenario dat is gebaseerd op UNFCCC beleid waarbij doelen gesteld door verschillende regio's en landen zijn opgenomen in het geïntegreerde systeem. De resultaten laten zien dat, door het koppelen van de drie modellen, we de scenario's gedetailleerder hebben kunnen simuleren, hetgeen nauwelijks mogelijk zou zijn geweest bij het gebruik van op zichzelf staande modelcomponenten.

Biography

Getachew Feleke Belete was born on 4 July 1976 in Bahir Dar, Ethiopia. He attended primary education in King Sertse Dengel elementary school and junior high school in Fasilo junior secondary school. Then, he attended high school in Tana Haike Comprehensive High School and he received Ethiopian School Leaving Certificate Examination (ESLCE) in 1993. From 1993 – 1997 he studied his bachelor's degree at Addis Ababa University Faculty of Science and he graduated with BSc degree in Physics. In 2002 he joined HiLCoE School of Computer Science and Technology and in 2004 he received Postgraduate Diploma in Computer Science. From 2006-2008 he did his masters study at Addis Ababa University Department of Computer Science and he graduated with MSc degree in Computer Science.

Getachew started his career by teaching Physics in high school and then in a college. In 1999 while he was teaching Physics in Defence Engineering College he took some courses on computer application programs, which latter resulted in changing his profession to computing. He has worked as software developer for Ministry of Defence, Addis Ababa University, and United Bank. In 2012 he got short term research opportunity at Aarhus University in Denmark and he conducted research on Domain-Specific Programming Languages. In March 2013, he joined University of Twente, Faculty of ITC to pursue his PhD. His research was financially supported by European Union FP7 project called COMPLEX and the focus of the research was on integration of various simulation models. During his PhD, he has presented the research findings in various international conferences and in project workshops. He also published his research findings in highly reputable journals. Some of his recent publications are listed as follows.

List of Publications

ISI Journal Articles

Belete, G.F., Voinov, A., Laniak, G.F. 2017. An overview of model integration process: from pre-integration assessment to testing. Environmental Modelling and Software 87, 49-63.

Belete, G.F., Voinov, A., 2016. Exploring temporal and functional synchronization in integrating models: A sensitivity analysis. Computers & Geosciences 90, 162-171.

Belete, G.F., Voinov, A., Bulavskaya, T, Niamir, L, Dhavala, K, Arto, I, Moghayer, S, and Filatova, T. Web service based approach to linking

heterogeneous climate-energy-economy models for climate change mitigation analysis. International journal of Energy. In review (after revision).
Belete, G.F., Voinov, A., Morales, J. Designing the Distributed Model Integration Framework – DMIF. Environmental Modelling and Software. In review.

Conference Proceedings (Full paper)
Belete, Getachew, Voinov, Alexey, & Holst, Niels. 2014. An architecture for integration of multidisciplinary models. Paper presented at the 7th International Congress on Environmental Modelling and Software (iEMSs), San Diego, California, USA.
Belete, Getachew, Voinov, Alexey. 2014. Integration of models for low carbon economy. Paper presented at the 7th International Congress on Environmental Modelling and Software (iEMSs), San Diego, California, USA.

ITC Dissertation List

https://www.itc.nl/Pub/research_programme/Research-review-and-output/PhD-Graduates