Institut für Informatik und Computational Science
Service and Software Engineering Group

Semantics-based automatic geospatial service composition

Author:
Samih Al-Areqi

Supervisor:
Prof. Dr.-Ing. Tiziana Margaria-Steffen

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy (Dr.-Ing.) in Service and Software Engineering

Faculty of Mathematics and Natural Sciences
University of Potsdam

Published online at the
Institutional Repository of the University of Potsdam:
URN urn:nbn:de:kobv:517-opus4-402616
http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-402616

I would like to dedicate my thesis work to my loving mother and father. To my wife Amal and my daughters Ranim and Rital and to my little boy Ibrahim. To my sisters Shima, Rima and Aya. To my brothers Hitham and Abdulkafi. To my uncles and aunts, especially to my uncle Abdulrakeeb. To my cousin Abdulhakeem.

Acknowledgements

This PhD would not have been possible to do without the support and guidance that I received from many people. I am deeply grateful to all of them and I would like to thank some of these people in particular.

First, I wish to thank my parents for their love and encouragement. Without you I would never have success in many opportunities. I can never thank enough my wife Amal for her love and patience. Your endless attempts to motivate me are the main enablers of my success. Thank you for accommodating my weird mood swings and for the weekends that I spent working on my thesis.

I would like to express my sincere gratitude to my supervisor Prof.Tiziana Margaria-Steffen that gave me the opportunity to carry out this PhD project. Thank you for your support, guidance and encouragement throughout this duration of this work.

I consider myself very lucky to work with Dr. Anna-Lena Lamprecht. She was very patient, understanding, and tolerant enough to watch the development of a young researcher like me with all the ups and downs. From her I learnt a lot, apart from the interesting research discussions that we had, it was a pleasure for me to discuss with her on many other issues. A special word of thanks is due to you, you have continuously provided me with the professional and personal support required to finalize my PhD project.

Special thanks to Prof. Hartmut Asche for his valuable feedbacks and being a part of my committee. You taught me how to do scholarly research, and helped me think creatively and independently.Your guidance and support are greatly appreciated.

I am very grateful to Steffen Kriewald, Dominik Reusser, and Markus Wrobel from the climate change and development group at the Potsdam Institute for Climate Impact Research (PIK), who provided me with the tools and data sets required for the research.

Throughout my years in the chair of service and software engineering at the University of Potsdam, I have made very special friends who have made my stay very enjoyable. I thank them dearly, especially Alexander Wickert. I would also like to thank Dr. Henning Bordihn for his support and friendship.

For reviewing the final thesis, I am very grateful to Prof. Barry who accept to be a reviewer for my thesis and thanks for your advises and feedbacks.

Last but not least, I gratefully acknowledge the funding received towards my PhD from the German Academic Exchange Service (DAAD) PhD scholarship. Special Thanks to Andrea Gerecke for the best DAAD coordination service. Receiving this scholarship motivated me to maintain and complete my PhD successfully. I thank the DAAD for their confidence and willingness to help me achieve my goals.

Abstract

Although it has become common practice to build applications based on the reuse of existing components or services, technical complexity and semantic challenges constitute barriers to ensuring a successful and wide reuse of components and services. In the geospatial application domain, the barriers are self-evident due to heterogeneous geographic data, a lack of interoperability and complex analysis processes.

Constructing workflows manually and discovering proper services and data that match user intents and preferences is difficult and time-consuming especially for users who are not trained in software development. Furthermore, considering the multi-objective nature of environmental modeling for the assessment of climate change impacts and the various types of geospatial data (e.g., formats, scales, and georeferencing systems) increases the complexity challenges.

Automatic service composition approaches that provide semantics-based assistance in the process of workflow design have proven to be a solution to overcome these challenges and have become a frequent demand especially by end users who are not IT experts. In this light, the major contributions of this thesis are:

(i) Simplification of service reuse and workflow design of applications for climate impact analysis by following the eXtreme Model-Driven Development (XMDD) paradigm.

(ii) Design of a semantic domain model for climate impact analysis applications that comprises specifically designed services, ontologies that provide domain-specific vocabulary for referring to types and services, and the input/output annotation of the services using the terms defined in the ontologies.

(iii) Application of a constraint-driven method for the automatic composition of workflows for analyzing the impacts of sea-level rise. The application scenario demonstrates the impact of domain modeling decisions on the results and the performance of the synthesis algorithm.

Zusammenfassung

Obwohl es gängige Praxis geworden ist, Anwendungen basierend auf der Wiederverwendung von existierenden Komponenten oder Diensten zu bauen, stellen technische Komplexität und semantische Herausforderungen Hindernisse beim Sicherstellen einer erfolgreichen und breiten Wiederverwendungen von Komponenten und Diensten. In der geowissenschaftlichen Anwendungsdomäne sind die Hindernisse durch heterogene geografische Daten, fehlende Interoperabilität und komplexe Analyseprozessen besonders offensichtlich. Workflows manuell zu konstruieren und passende Dienste und Daten zu finden, welche die Nutzerabsichten und -präferenzen abdecken, ist schwierig und zeitaufwändig besonders für Nutzer, die nicht in der Softwareentwicklung ausgebildet sind. Zudem erhöhen die verschiedenen Zielrichtungen der Umweltmodellierung für die Bewertung der Auswirkungen von Klimaänderungen und die unterschiedlichen Typen geografischer Daten (z.B. Formate, Skalierungen, und Georeferenzsysteme) die Komplexität.

Automatische Dienstkompositionsansätze, die Semantik-basierte Unterstützung im Prozess des Workflowdesigns zur Verfügung stellen, haben bewiesen eine Lösung zur Bewältigung dieser Herausforderungen zu sein und sind besonders von Endnutzern, die keine IT-Experten sind, eine häufige Forderung geworden. Unter diesem Gesichtspunkt sind die Hauptbeiträge dieser Doktorarbeit:

(i) Vereinfachung der Wiederverwendung von Diensten und des Workflowdesigns von Klimafolgenanalysen durch Anwendung des Paradigma des eXtreme Model-Driven Development (XMDD).

(ii) Design eines semantischen Domänenmodells für Anwendungen der Klimafolgenanalysen, welches speziell entwickelte Dienste, Ontologien (die domänenspezifisches Vokabular zur Verfügung stellen, um Typen und Dienste zu beschreiben), und Eingabe-/Ausgabe-Annotationen der Dienste (unter Verwendung von Begriffen, die in den Ontologien definiert sind) enthält.

(iii) Anwendungen einer Constraint-getriebenen Methode für die automatische Komposition von Workflows zum Analysieren der Auswirkungen des Meeresspiegelanstiegs. Das Anwendungsszenario demonstriert die Auswirkung

von Domänenmodellierungsentscheidungen auf die Ergebnisse und die Laufzeit des Synthesealgorithmus.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software reuse ranges from simple functions to complete applications, and is often considered the most effective means of improving productivity and maintainability in software development projects. The emergence of paradigms such as component-based software engineering (CBSE) and service-oriented software engineering (SOSE) has leveraged the development of applications based on the reuse of existing components and services. These paradigms significantly increased the possibilities of building systems and applications from reusable components [4]. Although it has become a common practice to build applications based on the reuse of existing components or services, technical complexity and semantic challenges constitute barriers to ensuring a successful and widespread reuse of components and services. In the geospatial application domain, the barriers are self-evident due to the large volumes of geographic data, the lack of interoperability, and the complex analysis processes. Thus, service-oriented architecture (SOA) principles and web service technology have been embraced by the geospatial community, and many works quickly followed the trend of building geospatial applications by reusing components and services [5; 6; 7].

In principle, embracing a service orientation in the geospatial domain makes geospatial data and processes increasingly accessible remotely in a distributed fashion through standardized geospatial web services [6]. Hence, scientific communities have become more aware of the benefits of sharing their data and reusing computational services. Hence, they are contributing to the development of distributed

data and services [2; 8; 9].

Scientific workflow technologies have emerged to simplify the service composition process. However, most existing workflow management systems suffer from a complexity of design. They aim to address the composition at a low level, and not at the level where the users will handle the composition and execution tasks. In order to hide the complexity of using the underlying technologies, agile methods have been introduced to open up the software development work to customers and users. Utilizing agile methods facilitates the service composition and enables large numbers of users to develop their own applications by carrying out the workflow design process in an easy and flexible manner. Therefore, the demand for semantically described services has increased with regard to identifying suitable services and enabling automatic workflow design [10; 11; 12; 13].

Discovering the appropriate service functionality is a crucial task in building an application based on service composition, which includes the matching of service input with its adjacent service output syntactically and semantically. This task becomes more complicated with geospatial services due to the many general-purpose geospatial services (such as data access services, portrayal services, data transformation services, and location-based services) and specific-purpose services designed to address particular geospatial applications [8; 14]. Even with a few services available, anywhere from dozens to hundreds of service composition solutions could potentially exist. In such cases, users face two significant challenges related to discovering a service that satisfies their needs, as well as exploring how to reuse and compose components correctly in order to develop their own software applications. Ontology-based approaches have been proposed to address the semantic challenges of service discovery and composition.

The remainder of this chapter is organized as follows: Section 1.1 presents a motivation example of the climate impact analysis to clarify the problem and the possible solution. Section 1.2 describes the research problem in detail. Before presenting the contributions of the thesis in Section 1.4, Section 1.3 poses the research questions.

1.1 Motivation Example

This is from the perspective of using the distributed infrastructure to develop the user's own software applications. Making the processes of climate impact analysis available as services and enabling non-IT professional end-users to compose these services in a workflow form tailored to specific needs are sure to increase productivity and support the flexibility of analysis processes. However, users will face the challenges of handling a semantic service discovery and an adequate workflow design for the multi-scale and multi-objective nature of environmental modelling to assess climate change impacts.

The scenario in Figure 1.1 is depicted as an illustrative example, where an arbitrary user wants to analyse the impacts of an SLR by defining a magnitude value of SLR for a particular region. Often, users have some information about the initial data and processes (in this case, the magnitude of the SLR of 2.5m and the region in question) and what they want to see in the end (e.g. a map showing the flooded areas for SLR impacts), but have no idea about the required steps or services and the adequate workflows that would realize the objectives and preferences of data input or output formats, scales, and geo-referencing systems.

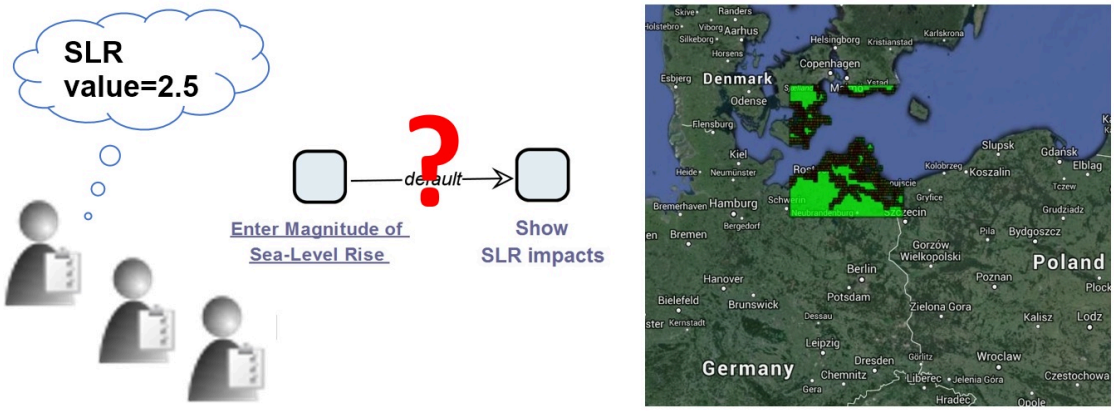


Figure 1.1: Example scenario: Analysing the impacts of SLR.

A possible approach to handle this situation involves providing services with a semantic description in a machine-understandable fashion. Ontologies are a widely accepted and state-of-the-art form of knowledge representation. They are employed to enrich semantics in both service description and composition [15; 16;

17; 18]. Using semantic descriptions, users can employ (semi-)automatic workflow design techniques [19; 20]. The most interesting such techniques for automatic workflow design are those that match users' specific requirements and preferences—for instance, regarding services that should or should not be used, by adding additional constraints to the workflow specifications.

1.2 Research Problem

Geospatial services, such as data access, portrayal, and processing services, are designed to perform a general task. They are applicable to data from multiple sources or may be coupled with a specific data source. Therefore, enabling users with diverse perspectives to compose these services in order to develop their own applications in a flexible and easy manner will maximize reuse of the service. However, the current solutions do not provide training on how to use new technology, such as for service composition, especially with the interoperability challenges of the geospatial application domain. This diversity of services and the various data sources increase the demand for semantic service discovery. In particular, finding the correct match between services to form a composed service (workflow) corresponding to users' requirements is the biggest semantic challenge.

Alongside attribute data, geospatial data includes geo-data that is larger in size and has a complex structure model. Geo-data is available in a range of different formats, scales, and data quality. It depends on different coordinate-referencing systems. Thus, it is important to take into account the data exchange between services to discover the correct service chain . In practice, however, this process is a nontrivial task, mostly involving geographical data, which is very different from other types of data [21]. For instance, different service interfaces are required to manage different data models (e.g. chaining a service returning image files (PNG, JPEG) with a service taking vector files) [8]. In addition to the enormous heterogeneity of geospatial data, the diverse sources of geospatial data are a major problem for semantic data compatibility between services [22].

Owing to the special characteristics of geo-processing operations, it is difficult to describe geospatial services and then ensure a semantic reuse for these services. For instance, operations calculating the distance between two points have the same

signature. However, they refer to the different meanings of distances (distance in the plane, on the surface, or across the earth) [10], and are based on different units and geo-referencing systems. Even more challenging is the complexity and the time taken to identify adequate (combinations of) services manually.

These challenges are critical for several reasons: First, turning spatial data and processes into loosely coupled components and interoperable geospatial services suffers from the technical complexity of using standards. Thus, users face a great challenge when it comes to *servification* [1]. Second, there is a lack of a framework for facilitating a geospatial workflow design and handling a lightweight approach of semantic-based automatic geospatial service composition. Third, no available ontological models can be used to address the semantic service composition for a greater range of user requirements, constraints, and input or output preferences.

1.3 Research Questions

Several requirements should be available to enable the end-user to reuse geospatial services and then find the adequate workflow design. In this section, we pose questions and define requirements regarding the handling of a user-level geospatial service composition and an automatic workflow design.

- How do agile workflows handle the complexity, flexibility, and reusability of geospatial service composition?
  - **RQ1 (Complexity handling)**: Which software technologies can provide the means to simplify geospatial service reuse and workflow design? Often, end-users working in scientific domains are scientists who have significant knowledge in their domain. They typically develop software applications tailored to their needs. However, 90% are primarily self-taught; therefore, they lack exposure to basic software development practices, such as developing systems from reusable components [23]. Therefore, hiding the technical complexity from end-users who are not

[1] With the term servification we refer to the process of turning arbitrary software components into proper services that are adequate, for example, for (re-) use in workflow management systems.

IT experts and providing them with technology that is reusable and that allows them to compose geospatial services in an easy way are key requirements for carrying out a manual or an automatic workflow design.

- **RQ2 (Flexibility handling)**: What system features can enable users to reuse existing services in an agile style in order to design several variations of workflows tailored to their objectives and preferences of input or output data? Geospatial applications are often used to support decision-makers; hence, users may need an exploitative approach to show experiment results and simulations. To increase the productivity and result exploration of workflows, we believe the system should provide users with advanced flexibility features. For instance, enabling users with a workflow-tracking mechanism and supporting parameter exploration during the workflow design are important in the geospatial domain because users have some information about data resolutions and ortho-rectification parameters.
- **RQ3 (Reusability handling)**: Geospatial services have certain characteristics such as complex processes and big datasets that hamper service reuse. Therefore, how can hiding the complexity from users and enabling a flexible workflow design increase geospatial service reuse? Moreover, how can the remote execution of geospatial processes remotely facilitate service reuse?

- What characteristics of our domain model play substantial roles to address semantic geospatial service composition? Moreover, how can the different sorts of semantics be achieved?

  - **RQ4 (Semantic handling of geospatial service reuse)**: How can geospatial services be described to ensure a semantic service reuse in a large number of workflow applications ? For instance, how can they be used to ensure consistent service reuse for general-purpose services, such as data manipulation services? Furthermore, how can users ensure the reliable reuse of complex geospatial processes, such as buffering and

overlay functions that are based on different data types (point, line, and polygon) and data models (raster and vector)?

- **RQ5 (Semantic handling of user intents)**: To cater to users in the geospatial domain on a large scale, considering different perspectives, how can the services be described to ensure a correct service composition that matches the user's goals and preferences? How can they provide the right service composition to the right user groups?
- **RQ6 (Semantic handling of geospatial data)**: How can geospatial services be described to ensure a semantic service composition with a wide range of formats, scales, and geo-referencing systems of geospatial data?
- **RQ7 (Quality of service (QoS) -awareness handling of geospatial workflow design)**: How can an adequate semantic domain model help to improve the performance of workflow synthesis?

1.4 Research Contributions

The contributions of this thesis are presented and performed through the following steps:

1. First, the research questions RQ1, RQ2, and RQ3 are addressed by developing agile workflows of climate impact applications. This is achieved by:
   (a) Applying the concepts of service engineering by making the functionality underlying the ci:grasp climate impact information platform [1] available as reusable services [24].
   (b) Using the eXtreme Model-Driven Design (XMDD)- oriented workflow development style [25; 26] supported by the jABC (jAVA Application Building Center) process modelling and execution framework [27], and enabling the end-user to easily compose geospatial services at the model level.

[1] http://www.cigrasp.org

(c) Leveraging the capabilities provided by the jABC to support a flexible workflow design. Thus, the goal is to design and adapt workflows of climate impact analysis according to the user's specific preferences and constraints [28].

(d) Evaluating the impact of using the XMDD paradigm over the service reuse of climate impact analysis processes [29].

2. Second, the research questions RQ4, RQ5, RQ6, and RQ7 are addressed by performing an automatic semantics-based workflow composition. This is achieved by designing a semantic domain model for specific applications of SLR impact analysis. Thus, to enable the end-user to easily integrate their knowledge in terms of the domain model and then to achieve a semantic service composition, the PROPHETS plugin [30; 31] of the jABC framework is used. This domain model comprises the following:

   (a) Semantics interface description: A higher level of semantics interface description is defined for geospatial services by providing symbolic names for services and their input or output data. The aim is to enable users to find services that match their needs and reflect their domain language [32].

   (b) Novel ontology: A new domain-specific service and type taxonomies for SLR applications are defined. By considering the bigger picture of the geospatial domain, the required taxonomies are designed by us. They have not been derived from any existing ontology models.

   (c) Constraints: These are used to filter the possible solutions of the workflow and to address several sorts of semantics of the geospatial service composition [33]. This helps to ensure and return adequate solutions of workflows and to improve the performance of workflow synthesis.

1.5 Research Organization

The remaining chapters are organized as follows:

- Chapter 2 demonstrates the related concepts of software technology trends that are used throughout this thesis. The focus is on the emergence of technologies, concepts, and systems that played important roles in software reuse, workflow design, and reducing the semantic gap between programmers and domain application experts. The technical frameworks, such as jABC, jETI (Java Electronic Tool Integration), and PROPHETS (Process Realization and Optimization Platform using a Human-readable Expression of Temporal-logic Synthesis) , are described in detail in this chapter.

- Chapter 3 discuses and evaluates the state of the art for developing geospatial applications based on service reuse. This includes investigations into current technologies, approaches, and systems that are used to address the challenges of geospatial service composition. Automatic geospatial service composition approaches, along with their limitations and challenges, are also discussed.

- Chapter 4 presents the use of XMDD technologies to enable users to flexibly define and perform multi-objective workflows tailored to their specific needs. To this end, variations of workflow examples for real-world scenarios of climate impact analysis are developed. These include the realization and experiences of the agile workflow design of climate impact analysis.

- Chapter 5 presents the semantic domain model that is designed for SLR applications. This includes the service interface description, ontology models, and domain constraints.

- Chapter 6 demonstrates how a semantics-based automatic workflow design is achieved based on the semantic domain model. It also shows how, through refinements in defining the constraints, several sorts of semantic service composition are performed.

- Chapter 7 discuses and evaluates the results of using XMDD and the semantic domain model to answer the questions mentioned above. This includes the discussion and comparison with related work.

- Chapter 8 concludes the thesis with a summary discussion and ideas for future research.

Chapter 2

Foundation

In this chapter, we introduce the key aspects of component-based software development (CBSD) through concepts such as software or service reuse, service composition, and workflow design. As these concepts are developed, we discuss the contributions of agile technology and its impact on reusable software. We also introduce methods that address the semantics of service composition. We discuss how component-based technology can overcome the challenges of ordinary systems to improve software reuse. Service engineering concepts, such as service-oriented architecture (SOA), are discussed to illustrate the impact of software reuse in CBSD. Finally, from the perspective of service engineering, we discuss how agile methodologies address application development and how semantic service composition can be effectively created.

2.1 Component-based Software Development

CBSD is an emerging discipline in software engineering that aims to encourage the reuse of software applications. Systems here are built by assembling components that have already been developed and prepared for integration. The concept of software reuse follows the DRY ('Don't repeat yourself') principle [34] [35]. It is largely considered the next evolutionary step to object-oriented programming (OOP) for building large-scale software applications by integrating existing software components [36]. The CBSD approach can potentially decrease develop-

ment and maintenance costs by increasing software reuse, flexibility, productivity, and maintainability [37]. However, several issues should be considered, such as abstraction (the more generic, the more reuse), interface design (better design), interoperability (components that are able to work together easily), and development strategies (how to manage the development process) to improve the reuse of software components [38] [39].

While studies have found that the success of software component reuse is higher with higher levels of abstraction, they also found that there is an inherent conflict. To be widely reusable, a component must be sufficiently general, scalable, and adaptable. However, these characteristics lead to increased complexity and, thus, are more complicated to use. Moreover, this complexity leads to more demanding computing resources and, as a result, to more expense. Hence, some researchers suggest that effective reusability may require a development approach that focuses on, for example, building at a level of abstraction that allows less flexibility and fine-tuning, but achieves greater simplicity [4].

This conflict between complex abstraction and simplicity in implementation provides a basis interface design in reusability with guidelines that focus on the ease of understanding. Research [40] suggests that a portfolio of reuse characteristics should include the following:

- Ease of understanding
- Functional completeness
- Reliability
- Good error and exception handling
- High cohesion and low coupling
- Portability
- Modularity

Another important facet of software reuse is interoperability. Interoperability is defined as 'the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or

no knowledge of the unique characteristics of those units' [41]. This issue has been addressed via the emergence of middleware technologies, such as the Common Object Request Broker Architecture (CORBA), Sun's JavaBeans, and Microsoft's Component Object Model (COM).

The use of software components offers interesting characteristics for the development of complex applications using distributed and enterprise platforms. However, the process of designing and engineering software systems based on component reuse can be difficult and more complex than those experienced in centralized computing systems. Some critical factors that affect component reuse include an assessment of the applicability or usefulness of a particular component to the design, the amount of work needed to reuse a component, and the assessment of the quality of a reusable component [40].

Although CBSD has proven to be successful for software reuse and maintainability, software developers still face numerous complexities, such as the increasing variety of platforms, varying and inconsistent protocols, and the proliferation of devices [42]. For instance, in the case of commercial off-the-shelf (COTS) software, one can customize each component until it matches the customers' needs. However, several problems may occur when the underlying software base is updated and does not fit with the customization. Moreover, in addition to the technical complexity of component-consuming frameworks, it is a critical task to define the level of granularity that should be applied for successful component reuse. More challenges of CBSD are identified for each category of stakeholder in [43].

2.2 Service Engineering

Service engineering—a discipline that realizes the design of services as a process and often in the context of web services—is a natural development of the software component. The term 'services' can be defined as loosely coupled reusable software components, which are often delivered in a distributed network using internet-based protocols that encapsulate discrete functionality [44]. In this section, we discuss the characteristics of service engineering, services, and their relationship to the concepts of reusable components.

Service engineering has much in common with component engineering and,

according to Sommerville, is the process of developing services for reuse in service-oriented applications [45]. Based on this definition, Sommerville presents three stages of a service-engineering process:

1. Identifying services that are independent and reusable. Service engineers have to guarantee that the service represents a reusable abstraction that could be useful in various systems.

2. Designing a service interface by defining service operations, as well as their inputs and outputs. This involves the structural design of messages that are sent and received by the service.

3. Implementing and deploying services. This involves making services available for use, often on a web server.

For the first stage, several issues must be addressed beforehand (see pages 519 and 520 in [45]). Furthermore, to identify services, it is helpful to use a service classification scheme, such as the one suggested by Erl [46].

In contrast to traditional software engineering principles, most service engineering tasks are performed on the fly at runtime in a collaborative manner [47]. This may occur because it is often the case that complete information about the system requirements is available, and because of the success of different methodologies and agile-based approaches in developing software systems based on services.

We believe the paradigm of service orientation introduces a new approach in building distributed applications, one where the application development process is conducted by discovering and composing services, rather than by traditional design and coding. Thus, a service-orientation approach utilizes services as fundamental elements to support agile, low-cost, and easily distributed system development based on service reuse [48]. By quickly assembling new business processes from existing services, a developer is able to effectively address the changing needs of a volatile business market [49].

Service-oriented Architecture

As noted earlier, a service-oriented approach aligns with our concepts of CBSD and reusability. In this section, we discuss the concept of SOA, as well as describe

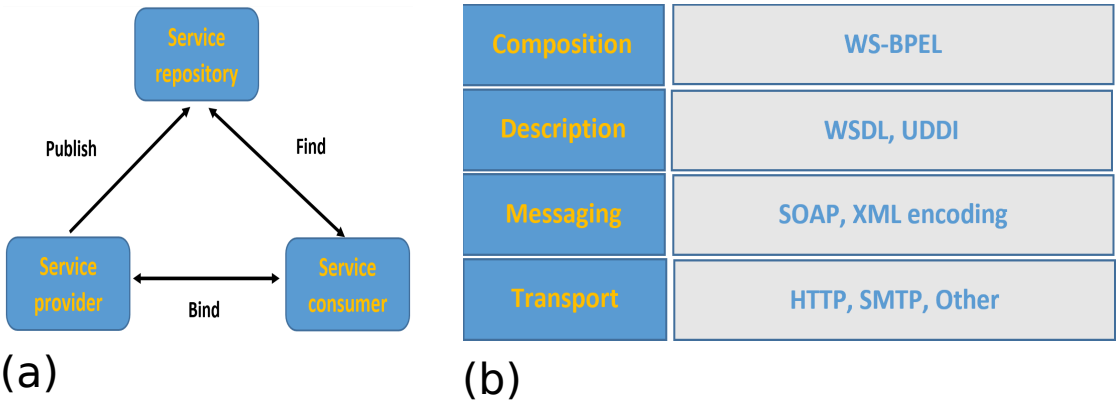


Figure 2.1: Service-oriented architecture. (a) SOA interaction structure. (b) Web service standards.

its approach and standards, and comment on issues like the complexity of its use.

SOA is an approach to software development that overcomes the challenges of heterogeneity and interoperability in CBSD by defining standard interfaces and protocols for developing internet-based services (hereafter called web services). These distributed systems are built on various platforms and technologies [42; 50], and they offer extensive functionality. To enable interoperability across applications over different platforms, SOA defines XML-based standards to represent computational or information resources that can be used by end-user applications and by other services distributed over the network.

Figure 2.1 shows interactions between three independent but collaborative entities that are involved in an SOA development process: service providers, service requesters, and a service repository. In addition to creating services and designing its interfaces, a service provider publishes its services in an accessible repository, thus giving service requesters the ability to discover services in the repository. After discovery, service requesters can bind their application to the discovered services by using standard service protocols.

Figure 2.1b shows key SOA standards, including: (1) Simple Object Access Protocol (SOAP) to describe communication between services, (2) Web Services Description Language (WSDL) to describe the service interface, and (3) Web Services Business Process Execution Language (WS-BPEL), a workflow language that uses WSDL, to orchestrate SOA applications. In addition, Universal Description,

Discovery, and Integration (UDDI) is an XML-based standard for describing, publishing, and discovering web services. Owing to improvements in search engine technology, however, using a standard search engine for discovering external services is now the preferred approach [45].

The process of using SOAP- and WSDL-based web service standards requires considerable effort in creating, transmuting, and interpreting XML messages. To simplify this process, the concept of RESTful web services has been developed, which presents resources as services. These services are identified by their URL and they communicate with service requesters using the HTML protocol [45; 51]. In principle, RESTful web services are not considered to be a standard but they represent an architectural style of web service design [52].

Service Composition

An important consideration in the process of developing software by using services is the concept of service composition. Service composition is a process of aggregating elementary services into one service, thus offering more functionality, and then recursively repeating this process to create new functionalities from composite services. Therefore, the process of composing services can be described as a sequence of individual steps to achieve a larger business goal. In principle, a composite service is similar to a workflow [53], where workflow refers to a progression of tasks to achieve an organizational or a professional goal. In sequential workflows, each step is dependent on the previous step. In parallel workflows, workflow tasks may be accomplished in parallel.

To have these workflows operate effectively, one must manage the transition from task to task. Thus, current research on service composition is focused on enhancing workflow systems to support the process of workflow composition [54].

Orchestration in the context of service engineering refers to the execution of a single business process, and describes the interactions between services at the message level. In orchestration, the interactions among services are controlled from a one-party perspective. This process control technique can be contrasted with choreography, which allows multiple business process endpoints (parties) to describe their part and interact globally by exchanging messages. Both techniques

may be involved in reuse; however, it can be seen that service composition supports service reuse and rapid application development. Towards this end, several approaches—such as workflow systems (Taverna), frameworks (jABC), and languages (BPEL)—have emerged to handle service composition. In principle, no standard composition approach currently satisfies a dependable and correct service composition. Nevertheless, to achieve a successful service composition, several requirements should be considered:

- Powerful process model: A powerful process model should be capable of (1) expressively modelling interactions between services into a hierarchical level of composition, (2) modelling control-flow structures, such as iteration and concurrency control, (3) representing data-flow among services, and (4) supporting exception handling.

- Validation and verification of the process model: It is important to ensure the correctness of the design of a service composition. Verifying static properties, such as syntax and type checks, may prevent errors before the actual service composition is executed. It is also feasible and important to simulate the behaviour of a workflow in order to detect errors in service composition before the execution time.

- Semantic-based service composition: In addition to the syntactic service description of services, which aims to describe the interfaces of services for invoking the service purpose, a semantic service description is an important issue. It describes the service's functionality, input, and output to ensure a common understanding between services and to ensure a correct service composition. An approach that is used to compose services should support: (1) semantic service discovery—that is, for a given workflow, a semantic description of service input and output should be available to find services that match existing services in the workflow; (2) automatic service composition—that is, a service-composition approach should have the ability to model the workflow of services automatically, based not only on input or output descriptions but also on higher-level specifications and non-functional attributes.

- Execution support: During the execution of service composition, a service

composition system should support: (1) dynamic service binding or replacing by other services, (2) runtime monitoring of service composition, and (3) robustness service composition.

Unlike CBSD, in service composition, service requestors and providers have access only to WSDL's functional descriptions of services [55]. Thus, various XML-based languages have emerged to handle the description of service compositions. For instance, WS-BPEL [1]is a well-known standard for web service orchestration, Web Services Choreography Description Language (WS-CDL)[2] is a language that describes the cooperating services of participants by defining their observable behaviour, in which services act as peers, and Business Process Modeling Language (BPML) [3] is a language that describes business processes that can execute BPEL systems and it incorporates many concepts of a web service choreography interface. According to a recent survey in [56], Web Ontology Language for Services (OWL-S)[4] seems to be the most popular format for web service description. Its models have been used to describe the composition and execution of web services.

The dramatic increase in the number of available services, the demand to update services on the fly, and the heterogeneity of concept models describing services all raise the complexity of service composition. Furthermore, developing and managing service composition has surpassed human capability to deal with the entire process manually. Therefore, there is a need for a semi-automatic or an automatic approach. To meet this need, several approaches have been proposed. In particular, most of these approaches use methods from the realm of workflow technologies and AI planning [57; 58]. Nevertheless, all these approaches depend on the availability of suitable semantic characterizations of services and data types in the application domain. Accordingly, in the context of service composition, a domain model that typically comprises services, data, their descriptions, and domain-specific vocabularies (usually in the form of an ontology) is a critical requirement to describe an application domain and, thus, to ensure a dependable service composition. We discuss these issues next.

[1] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[2] https://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/
[3] http://www.ebpml.org/bpml.htm
[4] https://www.w3.org/Submission/OWL-S/

2.3 Ontologies and Semantic Web Services

Ontologies are understood as dictionaries, taxonomies, categorization schemata, or modelling languages in the research area of artificial intelligence and computer science. According to Gruber [59], an ontology consists of terms, their definitions, and axioms of the relationships among them. In this approach, terms are normally organized into taxonomies.

Ontologies can be classified into three levels: top-level ontologies, domain ontologies, and application ontologies [60]. Top-level ontologies are concerned with describing the general concepts, independent of any domain. Domain ontologies describe the concepts in a specified sphere of activity or knowledge, such as in geospatial knowledge. Application ontologies are created for a specific use or application focus. Their scope is specified through testable use cases, such as those that would exist in an SLR application. Despite the commonality that ontologies possess in categorizing the same world, they are very different in their intent and focus. They also differ in their treatment of the basic parts of the ontology, including things, processes, and relations [61]. Some approaches are also incomplete in providing the necessary functionalities.

Within the semantic web [62], ontologies have been used to enable the machine learning of a large proportion of shared data on the web. Although the XML Schema approach enables common data interoperability, this approach reveals nothing about the semantic specification of data. Other approaches, such as the Resource Description Framework (RDF), overcome this problem. RDF is a framework built on top of XML and it provides data with semantics. To complete the approach, Web Ontology Language (OWL) [63] is added on top of RDF, supporting a reasoning service of the semantic web.

Building further, OWL-S [64] is an extension of OWL and provides web services with semantics. It comprises three essential types of knowledge about a service: a service profile, which is for advertising and discovering service capabilities; a process model, which provides answers to the question of how the service is used; and a service grounding, which provides information about transport protocols and the necessary details on how to interact with a service via messages. Therefore, OWL-S is a prominent solution supporting semantic service discovery and

composition.

Finally, Web Service Modeling Ontology (WSMO) provides web services with semantic descriptions, enabling the automation of service discovery, composition, and execution [17]. In order to define semantic web services, WSMO comprises four elements as the main concepts that have to be described: (1) ontologies that formalize the relevant aspects of the domains of the discourse; (2) web services that represent computational entities enabling access to services; (3) goals that describe a user's desires; and (4) mediators that handle interoperability and resolve heterogeneities. In contrast to OWL-S, which requires other languages to formulate service conditions and effects, WSMO has its own formal language, Web Service Modeling Language (WSML), which can formulate WSMO elements [65].

Despite the great impact of using ontologies and the semantic web to handle automatic service composition, different application domains may have very different constraints, features, and preferences. Thus, it is important to consider the application domain and its specific requirements for achieving a dependable service composition [66]. Furthermore, domain experts that have an insufficient technical background face complexity challenges in describing services and in using service-composition languages. One approach for resolving these complexities involves the use of model-driven design.

2.4 eXtreme Model-Driven Development: XMDD

Agile methods in the spirit of [67] have become increasingly popular in software development. Their core principle is to open up software development to customers and users in order to improve productivity, quality, and stakeholder collaboration and satisfaction.

As noted earlier, SOA supports agility by using services as the building blocks for applications. This approach enhances the involvement of inexperienced stakeholders in application development by providing feedback, such as which model is right and which model is better [68]. However, the semantic gaps between the involved, yet inexperienced, stakeholders and traditional software developers of orchestration and the management of evolution remain a problem [69]

With the trend of CBSD, SOAs, and aspect orientations as approaches to tackle

the software development complexity problem, model-driven development (MDD) emerged. MDD raises the abstraction level of software development through a development paradigm that focuses on and produces models rather than code skeletons and fragments. Rather than focus on the underlying implementation technology, in MDD, models are expressed using concepts that are closer to the problem domain. This approach makes the modelling activity easier for non-software professionals to specify and understand [70]. However, according to [34], the full benefits of MDD can be attained only when automatically generating complete programs from models and verifying the models on a computer as they are performed.

The MDD approach is known for having introduced into the field of SOA the ability to support the automation of service composition development [71] and to gain insight into the process of service composition. Nevertheless, MDD still remains mostly in the IT realm; hence, non-IT professionals are unable to gain insight into the process of service orchestration and evaluation. This lack of inclusion of non-IT professionals provided an impetus to reorient the approach, making the user the centre of software development, specifically when developing complex and cross-organizational systems that must adapt to rapidly changing market requirements. This approach is discussed in the next section.

The XMDD paradigm [26; 72; 73] represents a rigorous method of MDD that supports a very agile and cooperative development of SOAs by turning system development into a user-centric orchestration of intuitive service functionality. In principle, this paradigm combines and inherits the power of the ideas of MDD, service orientation, extreme programming, and aspect orientation [73].

XMDD promotes models to be the central and primary development tool for artefacts, thus constructing, controlling, and evaluating the overall development process at a model level. Rather than focus on software components, which XMDD uses only for specification, elementary building blocks should be established at a higher level of analysis and these components should be incorporated into a working, executable model [50].

In sum, software development using XMDD is performed based on a model library that houses integrated and combined to construct the large artefact—the global system model that specifies the system. In principle, as shown in Figure

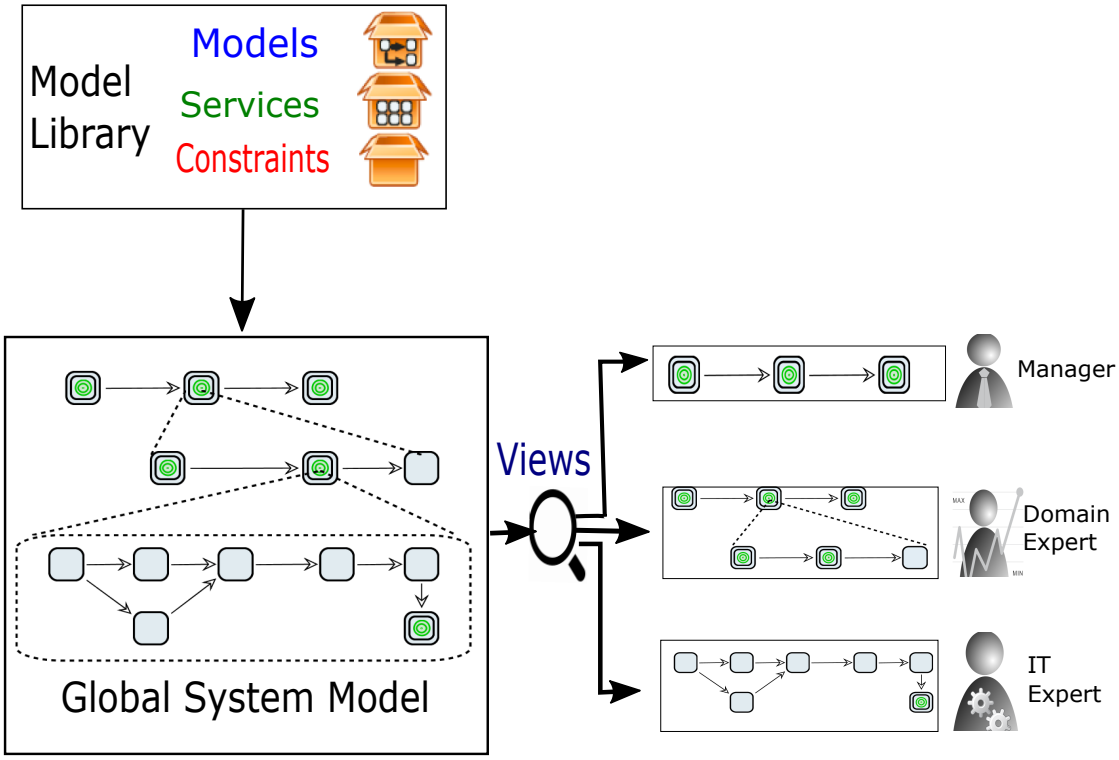


Figure 2.2: The Extreme Model-Driven Development (XMDD) approach, adopted from [1].

2.2, software development is performed based on a model library that consists of the following:

- Atomic services are elementary building blocks that represent the real functionality of the system and reflect models at the hierarchical level.
- Models are exploited from hierarchical modelling. Therefore, 'once created, a model can be seen as a ready-made aspect or feature, which becomes a part of the model library and thus can be reused in other application scenarios'.
- Constraints are used to ensure a consistent interaction of all contained models and services of the global system model. While local constraints concern the contained atomic services, global constraints concern the whole model that comprises all contained models and services.

As a core aim of XMDD, software development involves business experts continuously in the project control and evolution at the process-modelling level. A global system model is used to support a collaborative environment for designing the actual system. This approach provides a shared language for bridging the semantic gaps between developers and business experts. Thus, at the modelling level, the abstract nature of the models supports the model hierarchy, which hides the technical details and exposes higher levels of abstraction from the business perspective of the final product. This approach also enables a cooperative creation of models between developers and stakeholders—for example, between the customer and the management.

Like extreme programming, XMDD supports an agile mechanism dealing with immediate feedback for changing requirements. Furthermore, system changes can be made at the modelling level by re-synthesis of the global system model, which means that the global system model can be immediately and automatically translated into code for a desired target platform [1].

In the XMDD setting, the notion of a product is defined in terms of the features and services. Therefore, given adequate knowledge about the available services with an abstract description of a business process, one is able to generate adequate executable orchestrations [72]. These orchestrations exist in a user-centric, real, and immediately executable environment. Accordingly, this approach enables

rapid prototyping, debugging, and pragmatic evaluation of the modelled system, even at very early stages of development [1]. We now present an example of this technique using jABC.

2.4.1 jABC

The multi-purpose process-modelling and execution framework jABC [50; 74] realizes and implements the principles of XMDD. jABC has a comprehensive and intuitive graphical user interface that facilitates application development [27]. In essence, jABC enhances other modelling practices, like the UML-based Rational Unified Process (RUP), and by leveraging plug-in technology, supports most activities needed along the development lifecycle, including animation, rapid prototyping, formal verification, debugging, code generation, and evolution [72]. jABC's way of handling the collaborative design of complex software systems has proven to be effective and adequate for the cooperation of non-programmers and technical personnel [75]. Five main features characterize jABC: agility, customizability, consistency, verification, and a service orientation [72].

In jABC, the term 'Service' is used to denote a functional building block (called Service-Independent Building Block [SIB] in jABC), which is viewed as independent from its location, the program entity, and the hardware platform that provides services (SIBs) [76]. These SIBs are very close to an intuitive understanding of services that are required to be ubiquitously accessible (location-agnostic) and mechanically configurable [76]. Thus, an SIB can refer to any programmatically accessible piece of software functionality, such as APIs, REST services, and command-line programs [77]. SIBs are orchestrated with their behavioural semantics in mind, by means of lightweight process coordination [78], rather than by structured properties or a composition methodology focusing on operational aspects. Once each SIB is activated, it executes its logic and, upon termination, triggers subsequent SIBs according to the outcome of this execution.

jABC users can easily develop workflow applications by composing reusable building blocks (SIBs) into hierarchical (flow-)graph structures (called Service Logic Graphs [SLGs] in jABC) that are executable models of the application. The SLGs are hierarchical and support the reuse of sub-models. SLGs—being

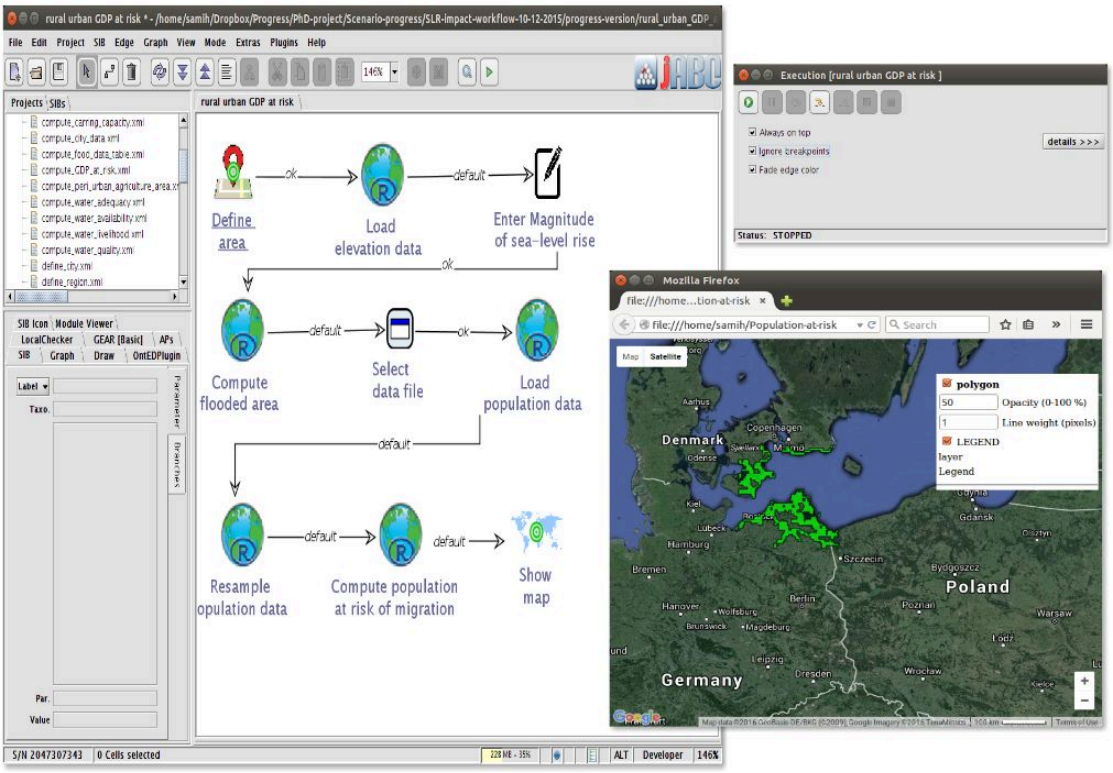


Figure 2.3: Graphical user interface of the jABC framework.

control-flow models of the developed applications—also allow an end-user to include essential control structures, such as conditional branching and loops, into the workflows.

Parallel execution is also supported by a fork/join mechanism that distributes the execution flow into different threads. Furthermore, the SLGs can be analysed, verified, executed, and compiled directly in the jABC environment. We now discuss the jABC framework.

Figure 2.3 gives an impression of the graphical user interface of jABC in action: The SLG on the canvas has been created using SIBs from the library (displayed in the upper left of the window) in a drag-and-drop fashion, and connecting them with labelled branches representing the flow of control. After the parameters of the SIBs have been configured (in the SIB inspector in the lower left), the workflow is ready for execution. The small window in the upper-left corner of the figure is the control panel of the Tracer plugin that steers the execution of the model. This

Tracer plugin indicates that it is currently executing an SIB. The green branches of the model on the canvas visualize where the execution has currently arrived. The third window in the figure shows an intermediate result from the workflow execution and has been opened by the currently executed SIB. In the next section, we discuss the relationship between jABC and jETI, a service integration platform.

2.4.2 jETI

The notion of service in jABC is fundamentally different from a web service notion. The ties to web communication protocols are not an essential part of jABC, but, instead, are provided by jETI technology [79] . The Java-based jETI [80] is a redesigned version of the Electronic Tool Integration (ETI) [81] platform, an open platform intended for interactive experimentation with and the coordination of heterogeneous software tools via the internet. jETI was designed to provide tool-users with:

- an instant hands-on experience with the tools, without the need to download and install software, which too often costs a considerable amount of effort and time, and

- an environment where they may publish and promote their tools, making experimentation available to end-users without the burden and legal issues of direct distribution, and where they may receive valuable feedback.

Although the ETI platform offered a good solution to integrate software tools remotely, its servers were too complicated for both the tool providers and the tool users. To follow the rapid development of methodologies, the jETI framework overcomes these problems by applying newer technologies and standards that are internally based on web services and Java technology. jETI replaces the requirement of 'physical' tool integration of the original ETI approach through a simple platform for registration and publishing.

As mentioned by Margaria [79], the jETI platform leverages newer technologies that are internally based on web services and Java technology for integrating functionality from the tools of different providers. jETI also builds from different

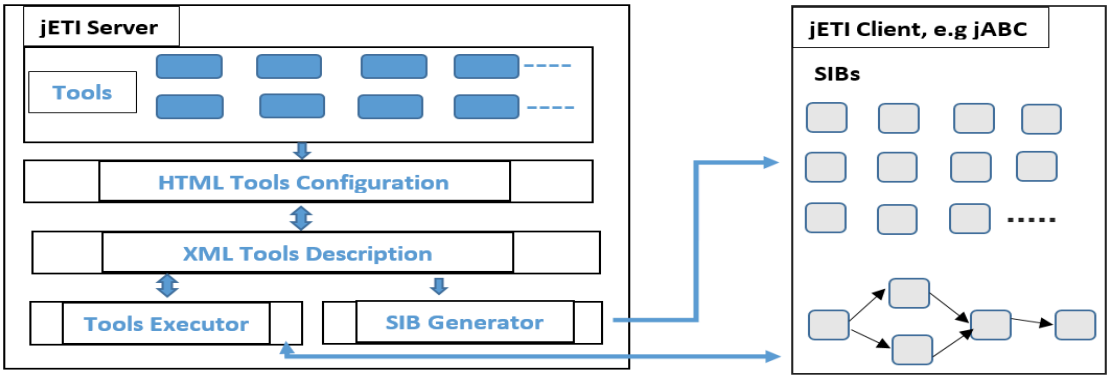


Figure 2.4: jETI overview.

application domains in order to make these tools easily accessible through a workflow framework (like the jABC) and, hence, allows one to solve complex problems. In particular, jETI is convenient, providing services for tools that work on files and have a classical command-line interface [20].

Corresponding to web services' functionality and service description standards, such as WSDL, jETI uses an HTML tool configurator to create service descriptions. Figure 2.4 provides an overview of the jETI infrastructure, which comprises the jETI server and a jETI client. The jETI server enables tool providers to register a new tool functionality by filling a simple template form in the browser that describes the service for the tool. The description comprises the interface definition (service name, tool to be called, input and output parameters), documentation text, and icons. The resulting tool configurations are maintained in XML files that are conceptually similar to, but somewhat simpler than, configurations using WSDL files that are created for describing web service interfaces. These XML files are used by the jETI server as the basis for automatically generating SIBs from service specifications. On the client side—for instance, in the jABC—users can easily use the generated SIBs to develop their own workflows.

2.4.3 PROPHETS

In addition to the manual workflow design, the jABC development platform has been extended to support the semi-automatic workflow design through a PROPHETS plugin. Based on the concepts of loose programming [30; 82], PROPHETS has

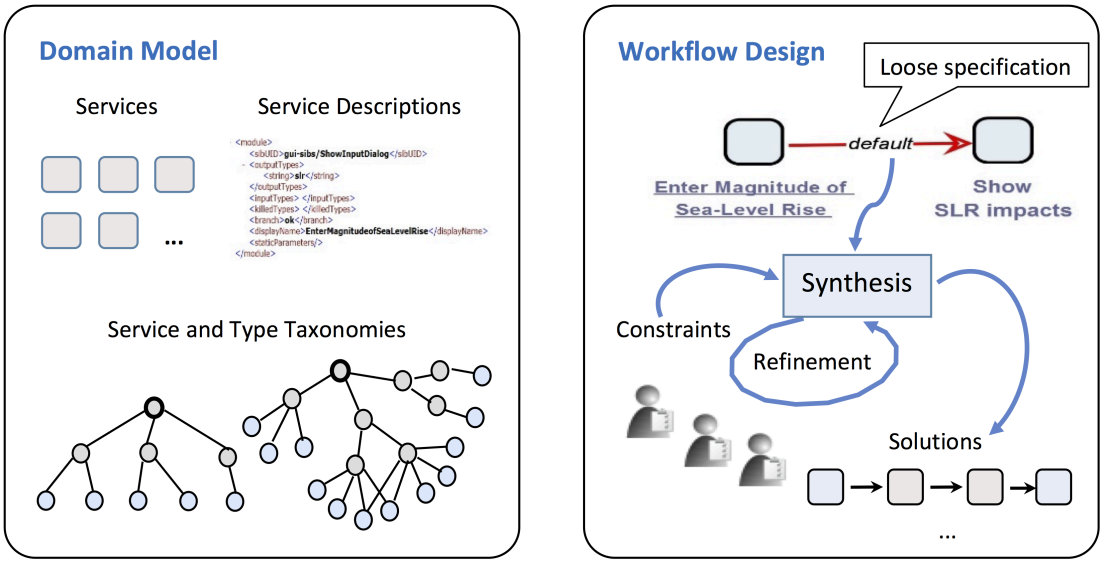


Figure 2.5: Automatic workflow design with PROPHETS.

been introduced to enable domain application experts to use a simplified workflow development technique. With PROPHETS, workflow designers are not required to implement the entire workflow manually. Instead, they can provide a sketch of the intended workflow, together with a set of constraints that further specify the user objectives. The plugin then applies a synthesis algorithm [83] to this abstract specification and returns a set of possible implementations to the user, who can select the one to be inserted into the workflow.

As illustrated in Figure 2.5, working with PROPHETS consists of two phases: *domain modelling* and *workflow design*. Thus, two major user roles are required to work with PROPHETS. While the application experts express their knowledge in a domain model, the workflow designer benefits from the domain model in defining constraints and then performing the workflow design automatically.

Domain Modelling

Before the workflow development for a particular domain, domain modelling that comprises adequate services design using semantic meta-information about the services and ontology models for types and services classification in the domain must be available. Therefore, in a domain-modelling phase, domain experts are

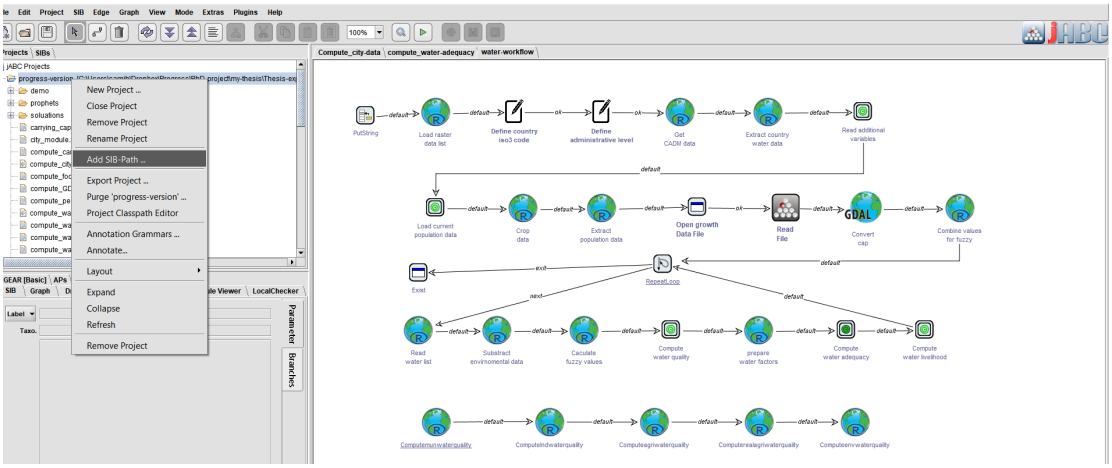


Figure 2.6: Integrating the services into jABC.

responsible for (1) providing resources (services, data, and metadata), and (2) modelling and formalizing explicit knowledge of the available resources in terms of ontologies. According to [20], domain modelling in jABC is performed through the following steps:

1. Integrating the services (the provisioning of SIB libraries in jABC): jABC's building blocks (i.e. SIBs) are implemented in Java, which means that they can use everything as an underlying service that is in some way programmatically accessible. This approach is also the case for Java APIs and web services. But in jABC, it also includes classic command-line tools, scripts, and 'headless' operation modes, as provided by some desktop applications. These blocks can be executed based on input parameters and without any user interaction.

   Depending on the technicalities of the chosen service, an SIB's integration into the jABC can be straightforward or challenging, but being able to subsequently use the services from an intuitive graphical interface typically outweighs the service integration costs. For instance, for the case studies of this thesis, in the jETI server, the services that are created for climate impact analysis are exported into a JAR file (see Section 4.2.2). In the jABC project we integrate these services as SIBs by adding the JAR file as a SIB path for the jABC project. Therefore, a list of services are become available and

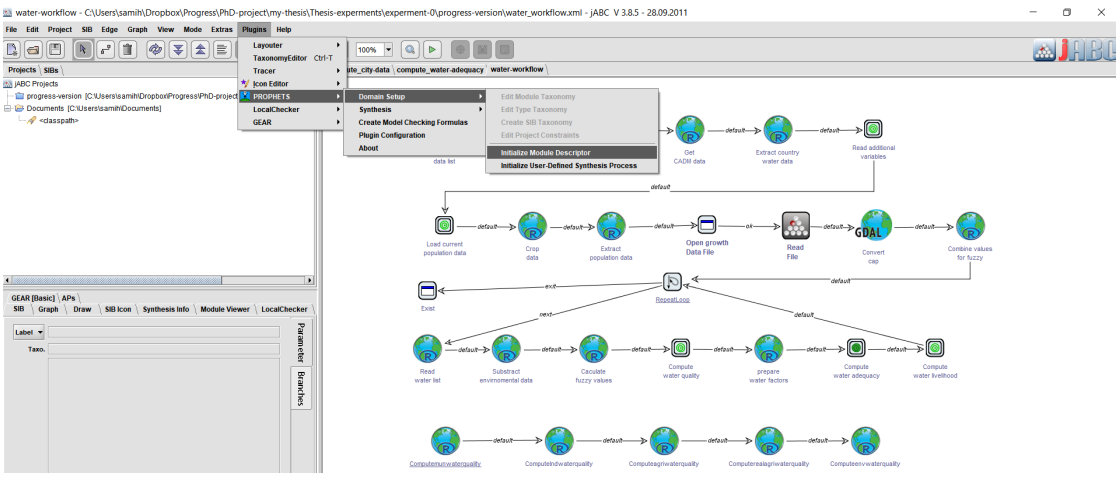


Figure 2.7: Initialization the module descriptor in jABC.

ready to use in the jABC canvas (see Figure 2.6).

2. Describing the behaviour of a service's interfaces (input and output): This description is carried out by means of a domain vocabulary. The resources (services and input/output data types) are described by the semantic domain models. As shown in Figure 2.7, by using the PROPHETS plugin in the jABC framework, once user initialized the module descriptor, a XML file is created to describe the service's interfaces of jABC project.

   In principle, modules (descriptive tags that are easy to understand by end-users), as shown in Figure 2.8, are created to describe services. Each module is linked to a concrete SIB that provides the implementation. However, the module's names and the names used to describe input and output data types for the synthesis algorithm are only symbolic. This naming scheme makes it possible to use a user's own domain-specific terminology for the module descriptions. In particular, it also allows for polymorphism in the sense that one SIB can be used by several modules. This approach is particularly useful for SIBs that provide quite a generic functionality and where several modules with specific functionality can be defined based on the same underlying implementation (modules with specific, unambiguous functionality lead to better synthesis results; cf. [20]).

   In PROPHETS, each service-interface description (module) characterizing

```
</module>
<module>
    <sibUID>gui-sibs/ShowFileChooser</sibUID>
  - <outputTypes>
        <string>GDPdatafile</string>
    </outputTypes>
  - <inputTypes>
        <string>slr</string>
    </inputTypes>
  - <killedTypes>
        <string>slr</string>
    </killedTypes>
    <branch>ok</branch>
    <displayName>SelectGDPdatafile</displayName>
</module>
```

Figure 2.8: Service interface description.

by means of three subsets:

- USE sets represent types that must be used before execution of the service (e.g. input types).
- GEN sets represent types that are generated during or after the execution of the service (e.g. output types).
- KILL sets defines types that are destroyed and, therefore, removed from the set of types that were available prior to execution of the service.

3. Defining a semantic (ontological) description for the available services and data types via taxonomies: This helps provide a controlled vocabulary for referring to entities in the domain model. Taxonomies can be seen as special kinds of ontologies—namely, ontologies with only *is–a* relations. In the jABC framework, the service management supported by a taxonomy editor plugin (see Figure 2.9)) to enable domain expert define service and type taxonomies in an intuitive interface. For instance, to define a simple form of ontology,

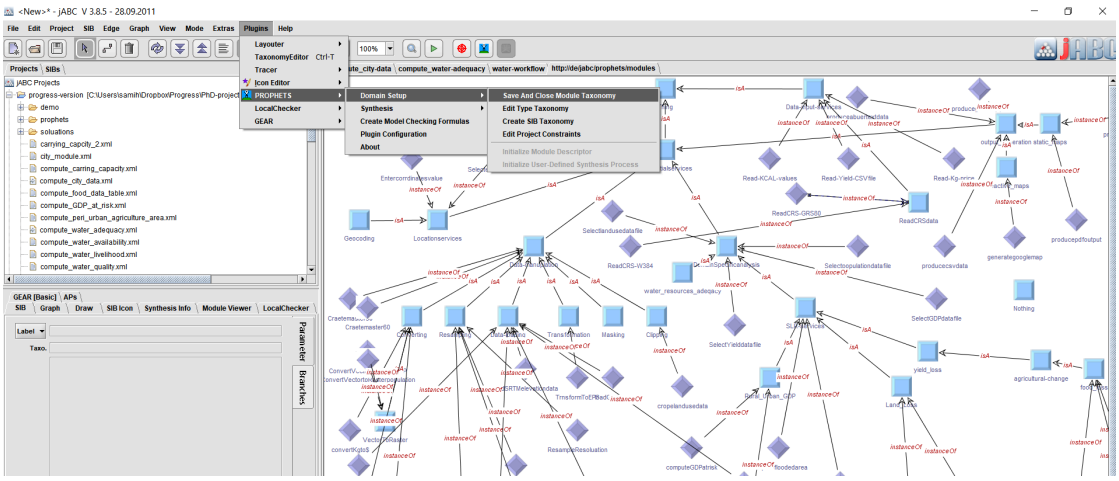


Figure 2.9: Taxonomy editor in jABC.

users can define a special-purpose taxonomies of the SIB libraries according to specific classification criteria, perspectives or input/output behavior. The definition of taxonomies is supported by PROPHETS. They are to be used in the synthesis algorithm and stored in the OWL format.

4. Providing users with domain-specific constraints formalism: In jABC, the PROPHETS enables users with a very flexible way to define constraints, it can be either during domain modeling or during the workflow synthesis. The constraints can be defined as branch conditions between two SIBs (e.g., the definition of the start and end condition(s) or as a domain constraint for the entire workflow. PROPHETS provides users with constraint templates. There, domain experts can easily use templates to define the constraints in a plain language. By supporting constraint formulation in terms of natural language templates, PROPHETS migrates users who have no background in formal methods from the complexity of understanding and writing logic languages to a user level language (English language). Table 2.1 presents the available templates that defined by Lamprecht [20] to represent different constraints in the English language. These templates are defined in an XML file. Therefore, more constraint templates can be defined easily by advanced users to handle the needs of a specific domain.

Table 2.1: Templates of constraint in PROPHETS.

Template name	Description
Service avoidance	Avoid the service s.
Conditional service avoidance	If service s1 is used, avoid the service s2 subsequently.
Mutual exclusion of services	At most one of the services s1 and s2 may be used.
Service redundancy avoidance	Do not use service s more than once.
Service enforcement	Enforce the use of service s.
Conditional service enforcement	If service s1 is used, enforce the use of service s2 subsequently.
Service succession	If service s1 is used, service s2 has to be used next.
Service dependency	Service s1 depends on service s2 (i.e., service s1 can only be used afters2).
Final service	Use service s as last service in the solution.
Type avoidance	Avoid the type t.
Type enforcement	Enforce the existence of type t.
Mutual exclusion of types	At most one of the types t1 and t2 may exist.

The Process Synthesis

According to logical specifications of the services, the term 'process synthesis' is used to refer to techniques that construct workflows from sets of services [84]. The PROPHETS synthesis plugin of the jABC workflow modeling framework benefits from the domain model to enable workflow designer obtain the adequate workflows automatically. Therefore, as in PROPHETS, the synthesis method relies on behavioural service interface descriptions—that is, services are regarded as transformations that perform particular actions on the available data. Depending on the modal Semantic Linear Time Logic (SLTL), the synthesis algorithm combines relative time with the descriptions and taxonomic classifications of types and services [85].

In order to construct the synthesis universe which represents possible solutions that contains all service sequences, in term of USE, GEN, KILL sets, the synthesis algorithm combines behavioral service interface descriptions. In principle,

according to available services, the synthesis universe is an automaton that connects states with edges. Depending on service input and output specifications, each state serves as a subset of all types and the transition on those types are represented by the connecting edges between states. The taxonomies are considered by the synthesis algorithm to construct the synthesis universe and to evaluate type and service constraints. Formally, as defined in [30] the synthesis universe is a triple

$T, S_c, Trans, where$

- T is a set of concrete and abstract types
- S_c is a set of concrete services
- $Trans = \{(t, s, t')\}$ is a set of transitions where $(t, t' \subseteq T$ and $s \in S_c)$

According to the definitions of USE, GEN, and KILL, a service $s \in S_c$ can be defined as a transformation on the set of types as follows: $s : 2^T \rightarrow 2^T$, $t \rightarrow (t\ KILL(s)) \cup GEN(s)$.

The sequences of services that meet the individual workflow specification are described in SLTL formula. The syntax of SLTL is defined by the following BNF, where t_c and s_c express type and service constraints, respectively:

$$\phi ::= \text{true} \,|t_c\,| \neg\phi \,|\, \phi \wedge \phi \,|\, \langle s_c \rangle\, \phi \,|\, G\phi \,|\, \phi \cup \phi$$

Where

- $\langle S_c \rangle\, \phi$ declares that ϕ must hold in the successor state and it reachable with the service constraint S_c
- G expresses that ϕ must hold generally.
- U specifies that $\phi 1$ has to be valid until $\phi 2$ finally holds.

SLTL includes: a) static constraints which represent the taxonomic expressions over the types or classes of the type taxonomy, b) dynamic constraints which represent the taxonomic expressions over the services or classes of the service, and c) temporal constraints that suitable to express the ordering constraints [85].

Finally, the specification formula used as input for the synthesis algorithm to integrate all available SLTL constraints.

Workflow Design

Following the concept of loose specification, in the workflow design phase, it is possible to indicate one or more branches between SIBs as loosely specified and apply the synthesis framework provided by PROPHETS, thus replacing them with appropriate concrete service sequences. Figure 2.10 (top) shows how the loosely specified branch between the SIBs is indicated (colored by red) by the PROPHETS plugin. Apparently, a loose branch represents all sequences of services that would constitute a valid connection between the respective SIBs. The PROPHETS plugin automatically transforms the domain model and constraints into the SLTL formula, thus, the specification formula encompasses all available workflow constraints such as start conditions, end conditions, and domain-specific workflow constraints.

With PROPHETS plugin in the jABC framework, the workflow designers need to only drag and drop the start and end services (SIBs) and then to follow a wizard-style user interfaces to accomplish the tasks of workflow construction. As shown in Figure 2.11, the overall synthesis process of PROPHETS performs the following steps:

1. After a loose specification is marked, the workflow designer can define additional constraints to be taken into account by the synthesis. For this purpose, PROPHETS provides users with a constraint editor to use the constraint templates (the 12 templates available in PROPHETS in Table 2.1).

2. Through the synthesis data preview window, the domain information (service interfaces and taxonomies) and the workflow specification and constraints are displayed.

3. The search dialogue is displayed to enable search parameters configuration. For example, status information—like the current search depth and the number of solutions that have been found so far—is displayed while the search runs.

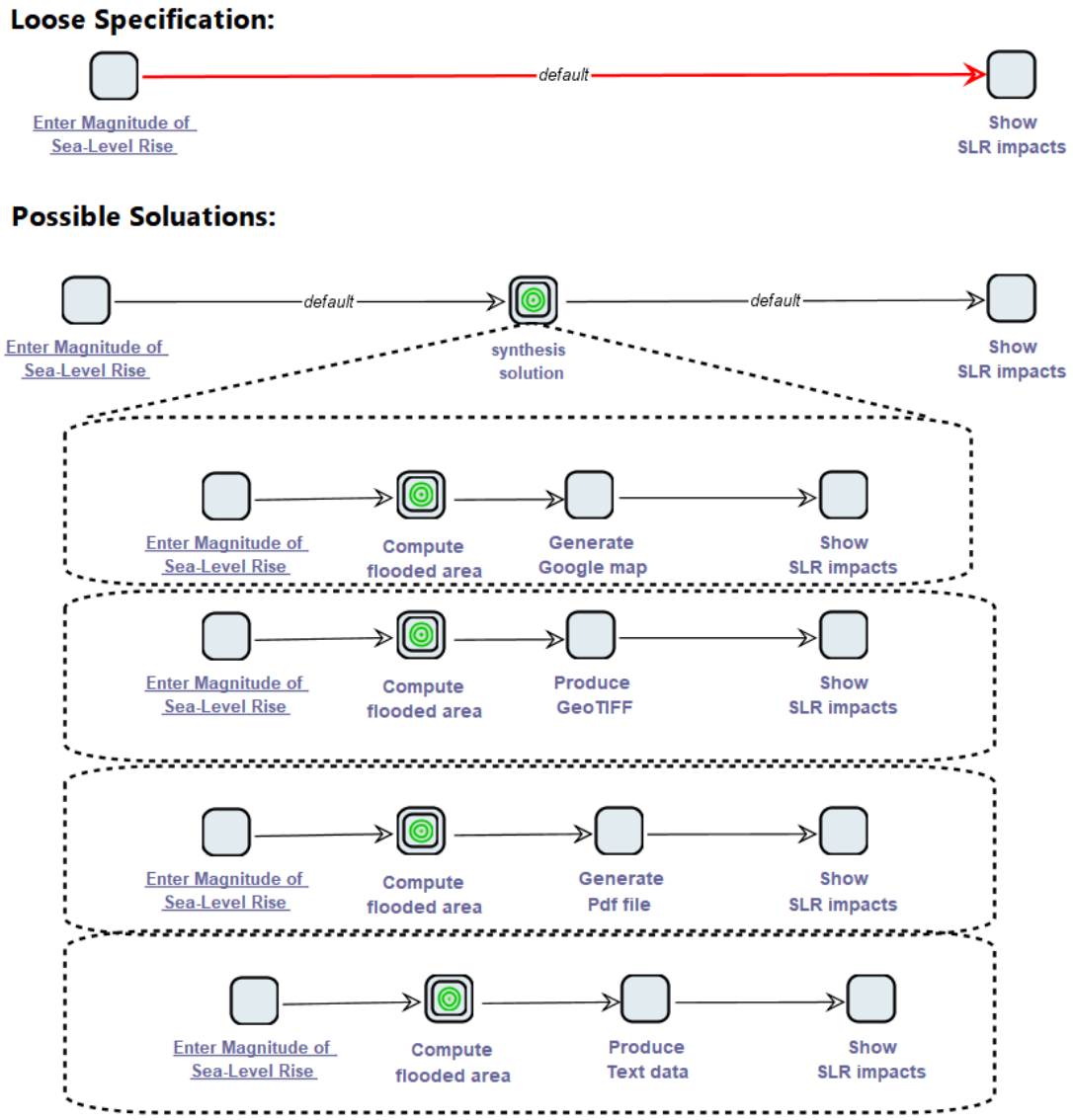


Figure 2.10: Loose specification and possible solutions.

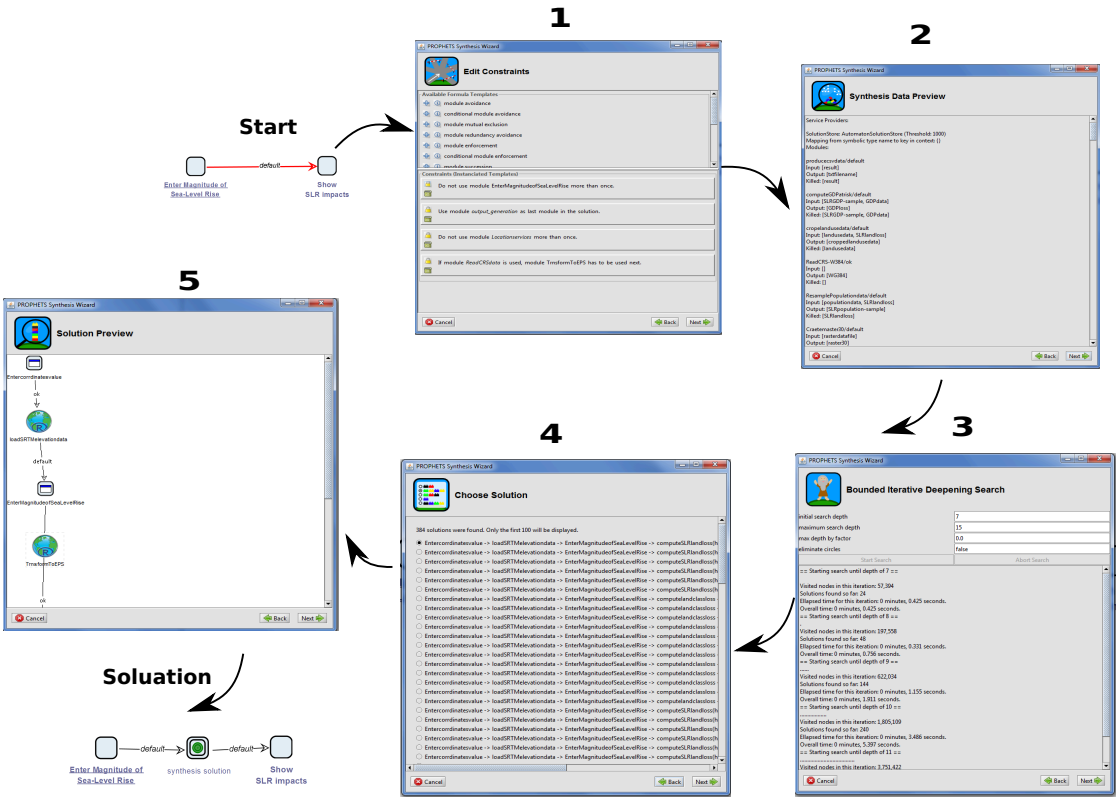


Figure 2.11: Steps of the PROPHETS synthesis process.

4. The potential solutions that have been found in the previous step are displayed in the window providing the solution choices to the user (workflow designer). The user can then directly choose an appropriate solution from the list, or decide to return to the first step to refine the specification and restart the synthesis process. It is also possible to configure PROPHETS to select one of the solutions automatically according to a particular cost function.

5. In the solution preview, the selected solution is displayed as a sequence of SIBs that will be inserted into the workflow in place of the loose branch.

Typically there are many different possibilities for workflows implementing this specification, and the adequate solutions depend on user requirements such as, analysis objectives (e.g. compute flooded area) or using certain type of data formats and resolution). Accordingly, the four exemplary possible solutions for

compute flooded area that are shown in Figure 2.10 (bottom), are only some of the many possible solutions for the synthesis problem defined by the loose specification.

Furthermore, as discussed in detail by lamprecht [20], to allow for a very specific tailoring of the synthesis process to the characteristics of the domain model, a variety of configuration options is provided by the PROPHETS. For instance, the execution of synthesis depends on the configuration of the PROPHETS installation. That is, it can be set up to the 'silent' mode in order to perform execution without more user interaction or work with different intermediate steps to customize synthesis inputs and parameters.

Chapter 3

State of the Art

In this chapter, we demonstrate the state of the art in geospatial service standards, reuse, and composition. In Section 3.1, we investigate the interoperability challenges of geospatial data and processes, as well as how SOA is used to address these challenges. In particular, the investigation focuses on how the complexity, flexibility, and reusability of geospatial service composition are handled. Section 3.2 identifies the workflow systems and technologies that are used to compose geospatial services. Section 3.3 demonstrates the current approaches and technologies of automatic geospatial service composition. This includes the semantic handling of geospatial service composition.

3.1 Geospatial Services

In the geospatial context and based on ISO TC204, the interoperability definition emerged as follows: 'The ability of systems to provide services to and accept services from other systems and to use the services so exchanged to enable them to operate effectively together' [10]. Several works have sought to improve the data and service interoperability of geospatial applications. Spatial data infrastructure (SDI), an internet-based information system, has been designed to facilitate geospatial data sharing across several interconnected systems [86]. SDI has been an accepted principle of SOAs. Thus, it offers and share spatial data through web services [87]. The advancements in general web service technologies and in GIS

service standards, such as SDI and Open Geospatial Consortium (OGC) [1] web service standards, encouraged the migration from the traditional form of standalone geospatial applications to loosely coupled components and interoperable geospatial services [2; 8; 88; 89].

Geospatial services are developed to make geospatial data and processes as interoperable piece that can be used in a wide range of applications. However, large volumes of geographic data, the lack of interoperability, and complex analysis processes constitute barriers to ensuring a successful and wide reuse of the components and services of geospatial applications. Therefore, the designing of services for geospatial applications should leverage web service technologies by considering the unique requirements of the geospatial community, which are due to the inherent complexity of geographical information, the heterogeneity of data models, large amounts of existing spatial data, and the variety of data formats that are utilized by different types of programs and applications [2; 8].

The greatest value of SOA is to enable interoperability between services. Therefore, SOA principles and Web Service technology have been embraced by the geospatial domain. Many researches quickly followed the trend of building geospatial applications by reusing components and services [8; 90; 91; 92]. In contrast to standard and general-purpose GIS applications, most current geospatial applications based on the SOA approach may provide users with just the functionality they need by combining distinct processes in one service (e.g. [5; 6; 7]), thus producing a number of domain-specific services. However, these services are restricted to specific applications and suffer from the complexity of service reuse.

Open standards, developed by the OGC and the International Organization for Standardization (ISO), are designed to specify geospatial web service interfaces. This results in a series of syntactic interface specification standards for the interoperability of geospatial web services. OGC follows the mainstream publish-find-bind paradigm represented by the SOAP, WSDL, and UDDI standards for the interoperability of geospatial web services. However, OGC web services (OWS) standards do not comply with these standards for web services in the business application domain. OWS are defined as modular components based on XML technology and the Geography Markup Language (GML), which was developed as an encoding

[1] http://www.opengeospatial.org

standard for geographical information [93]. To publish and invoke OGC services through the web, a number of OGC specification standards are defined:

1. The Web Map Service (WMS) produces and portrays geospatial data in maps in standard image formats, such as PNG, JPEG, or Scalable Vector Graphics (SVG). Through a simple HTTP interface, WMS provides a requesting for geo-map images from one or more distributed geospatial databases [94; 95].

2. The Web Feature Service (WFS) describes and manipulates data operations at the level of the geospatial feature (e.g. points, lines, and polygons) [96]. Unlike WMS, it allows users to access, retrieve, and exchange geospatial data at the feature level encoded in GML.

3. The Web Cover Service (WCS) allows users to retrieve a specialized class of features as coverage. Unlike WMS, WCS provides maps with detailed descriptions based on the GML application schema [97].

In addition to these service specifications, OGC developed catalogue interface standards to specify the interfaces. This was done by defining the application profiles required to publish and access digital catalogues of metadata for geospatial data and services [98]. The Web Processing Service (WPS) is designed and approved by OGC to specify the service functionality of geospatial services [99]. Through the 'GetCapabilities' operation of WPS, users can find the capabilities of the specific server implementation. The 'DescribeProcess' operation provides users with detailed information about the processes that can be run on the server. Finally, the user performs the 'Execute' operation to run a specific process by providing specified inputs and receiving the output. In addition, the generic interface of WPS supports syntactic geospatial process interoperability and OGC service composition [100].

Owing to the special characteristics of geospatial data, such as diversity of the massive dataset and numerous specifications of the data (e.g. formats, types, models, and scales), most approaches for geospatial domain application addressed syntactic and semantic service composition superficially and separately [21]. Several works focused on the construction of domain-specific applications by assembling and reusing geospatial processes and data as services [5; 6; 7]. Some of these

works support the generality and reuse of specific applications (e.g. environmental models [101; 102] and climate change vulnerability [7]).

Various attempts are presented to address the challenges of geospatial service composition, such as complexity, flexibility, and reusability. [90] made one of the first attempts, where he introduced a simple scenario to evaluate three approaches of geospatial service chaining: client-coordinated, static, and workflow-managed service chaining. These approaches explored the key issues of designing and implementing the geospatial service chain, such as how the users should see the service chain complexity, how the service chain and the metadata of geospatial data can be tracked, and how services can handle errors. He showed how static chaining using aggregate services can be an alternative to hide such chaining complexities; however, this does not allow the user to adjust and control the parameter. Workflow-managed service chaining, which is presented as a mediating service, makes a promise to simplify service chaining. Nevertheless, one of its drawbacks is the use of complex underlying technologies, such as XML-based service description and orchestration language.

In term of standards and to facilitate the reuse of geospatial services, OGC web services are commonly used to define the protocols of geospatial service composition [87]. Most geospatial service composition approaches, based on standard OGC service types, do not support SOAP and usually utilize OGC WPS as an interface for Web Service orchestration [91; 100; 103]. However, WPS has a limited functionality for composing geospatial services—that is, it cannot run processes asynchronously and manage them during the execution time [104]. In spite of the successful implementation of OGC standards by different providers (commercial and open-source), these standards become increasingly complex, making their implementation an arduous task [105]. According to [106], owing to the lack of documentation and semantic description, there are a limited number of WPS implementations (only 58 out 9,392). Furthermore, OGC standards do not comply with the web service standards defined by the W3C and OASIS. Therefore, developing geospatial services and composing them based on OGC standards requires additional technical efforts from both developers and users.

SOA approaches have been proposed to support and facilitate geospatial service reuse and composition. As shown in Figure 3.1, [2] presents an approach that

demonstrates how the distributed geoprocessing applications can be developed by using SDI and OGC services (WMS, WFS, WCS, and WPS).

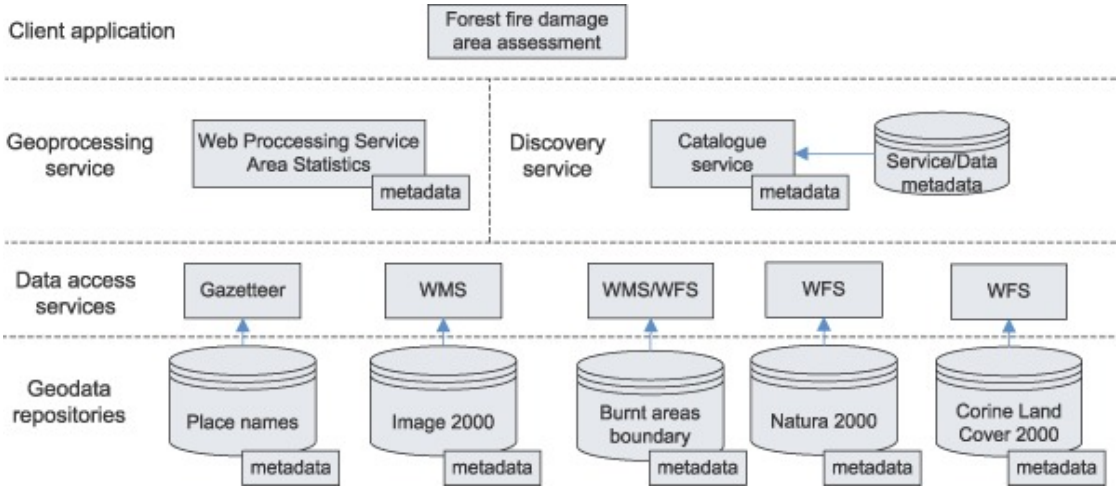


Figure 3.1: Designing Service Architectures for Distributed Geoprocessing, adopted from [2]

This approach shows that the distinct processes are combined in one geo-processing service that becomes a more specific application. However, to increase the flexibility and service reuse, a distinct service should be implemented for only distinct processes. The AWARE architecture approach, based on SOA and the principles of INSPIRE (European Union framework directive)[107], is presented to address geospatial data and operation interoperability [8]. Through a geo-portal, this approach allows users to execute geo-processing services remotely. Nevertheless, this approach is restricted to integrating and invoking the services of OGC standards.

3.2 Geospatial Workflows

In spite of the solutions offered by web service standards such as WSDL, web service orchestration, which is the harder task, remains a subject for the emerging technologies [108]. From the aspect of SOA, workflow systems are often used to handle syntactic service composition manually [14]. In these systems, to create a workflow, the services are described statically and usually a domain expert takes

into account user requirements in order to discover services. [88].

Scientific workflow technologies emerged to facilitate and support the composition and execution of complex analysis processes in a flexible fashion. This was achieved by simplifying the programming effort required by scientists or users to orchestrate computational tasks [108; 109; 110; 111; 112]. In contrast to the communication- and document-oriented workflows in the business domain, scientific workflows are data- and computation-oriented [113; 114; 115].

Geospatial workflows are considered to be a special kind of scientific workflow. Hence, workflow technologies and well-known scientific workflow systems, such as Kepler [116] and VisTrails [117], have been applied early on to deal with the coordination and the execution of complex geospatial Web processes. The GeoOpera tool, which is built up on workflow technologies, provides a fully distributed and heterogeneous computing platform. It has been presented to support aspects of geo-process management [118].

BPEL-based business workflow technology has been used to orchestrate geospatial services. On the other hand, a geospatial workflow engine based on ActiveBPEL has been proposed to enable BPEL to orchestrate OGC web services [119]. A BPEL workflow engine called BPELPower has been designed to support the features of scientific workflow engines at the implementation level (e.g. a workflow approach to coordinate between Web live sensors and earth science models (ESMs) [120], and as a general sensor Web data service framework for geoprocessing workflows [121]). A different approach is proposed and evaluated by using Simple Conceptual Unified Flow Language (SCUFL), used by the Taverna Workbench and the BPEL for developing geo-scientific workflows [122]. Recently, BPELPower was enacted to handle GML and to orchestrate OGC services, such as WPS and WFS [123]. Despite various attempts to use BPEL to orchestrate geospatial services, BPEL still has some challenges for orchestrating OGC web services, such as the lack of SOAP support, raw binary data served by OGC services, and the manual creation of the WSDL document for each process [92; 119].

Several studies have shown that orchestration using non-BPEL approaches is feasible. For instance, an approach is presented to extend the capabilities of WPS in order to use SOAP and WSDL [124]. This allows using OGC services in a generic workflow system like Taverna [125].

The Kepler scientific workflow system [126] has been applied to handle distributed geospatial data processing using web services [127; 128; 129; 130] and to compose OGC services [121; 131]. However, most of the current scientific workflow technologies in the geospatial domain are designed for the non-geospatial domain and, in particular, are used to address more specific applications, which are suitable for concrete application requirements—for instance, to couple the components of climate and hydrological models [132] for climate data analysis [133], and for the integration of complex earth-system models [130]. The scientific workflow was also utilized as a mediation service to compose and chain geospatial services for some geospatial applications [112].

Despite their promise to simplify the service composition process, scientific workflow management systems are often inherently complex and challenging in use and design, especially for heterogeneous resources. Furthermore, the majority of current workflow technologies are designed to support service composition at a lower, technical level, and not at a level where average users can handle the composition and execution tasks. Thus, users have to intervene in the composition at design time and it is time-consuming. Some attempts addressed the technical complexities of workflow systems by enabling web-based workflow composition and editing [127; 134]. Nevertheless, learning how to apply these technologies to build a system based on services remains complex for application experts, particularly given the interoperability challenges of geospatial data.

A geo-processing workflow tool named GeoModelbuilder is designed to support open standard-compliant geospatial web services and data [135]. It offers a flexible and user-friendly way to integrate sensors, geo-processing services, and environmental models. However, it is used only to integrate the environmental model corresponding to an open standard-compliant OpenMi. In addition, and up to the time of writing this thesis, it does not seem that GeoModelbuilder is used to orchestrate geospatial services based on SOAP or being OGC-compliant. A Business Process Markup Notation (BPMN) approach is introduced in the near term to flexible OGC service chaining using the standard of WPS [136]. Nevertheless, users face some challenges of using an editor and service discovery. Furthermore, this approach is tested in a specific challenging use case of crowd-sourced spatial data and its quality in terms of performance.

3.3 Automatic Geospatial Service Composition Approaches

Owing to the dramatic increase of geospatial Web services and the diversity of concept models that describe geospatial services, composing services manually is a highly complex task. As pointed out in [14], at present, there are three ways to compose a geospatial service: the manual approach for syntactic geospatial service composition, and the automatic and semi-automatic approaches for semantic geospatial service composition. Semantically interoperable geospatial services can be composed automatically or semi-automatically. Hence, it is necessary to provide geospatial services with a semantic description that is processed and interpreted by a computer.

The semantic interoperability problem is classified into three classes [10]: While the first two classes are concerned with data and service discovery, the third class is concerned with determining whether and how services can be composed to produce the desired behaviour. According to [137], handling the semantics of web services constitutes: (1) data or information semantics by annotating the semantics of input and output of service operations, (2) functional semantics by defining a formal representation for the service function, (3) execution semantics by modelling the workflow of services, and (4) QoS semantics by providing the quality criteria to select the suitable service

Several efforts have been made to handle the semantics of geospatial service composition. UDDI and the Electronic Business Registry Information Model (ebRIM) are prominent service registry models for supporting service discovery. In the geospatial domain, OGC has implemented and recommended ebRIM as a profile for the Catalogue Service for the Web (CSW) profile [138]. The CSW based on ebRIM has been semantically enhanced by adding extensions for registering services to support semantic service discovery [139]. However, the CSW based on the ebRIM profile and taxonomy models is bounded for OGC services and supports only the syntactic matching of keywords, which leads to the lack of discovery results. Furthermore, it does not deal with the content of the service, which is an essential aspect to reach full interoperability, and it does not originate directly from the users or domain experts.

Standardization and annotation approaches have been utilized to resolve syntactic and semantic geospatial data heterogeneity [140; 141]. The OGC standards help to overcome the syntactic service composition. However, it does not help to resolve the semantic service composition. Accordingly, individual services can only be evaluated for conformance to specifications, but not for interoperability with each other [10; 14; 139; 142]. One possible solution for achieving the semantics of data interoperability is the annotation approach. This can be achieved by using ontologies [143; 144].

Ontologies have been utilized as a promised solution to handle the semantic discovery of geospatial data and services [145]. An ontology-based catalogue of services proposed to capture the semantics of users' queries [146; 147; 148]. Service taxonomies have been introduced by the ISO in ISO19119 [149]. [149; 150] are also taxonomies that are used to support the semantic discovery of geospatial services.

To handle automatic geospatial service composition, few attempts have addressed the execution semantics ([semi-]automatic workflow composition) of geospatial services. The Science Environment for Ecological Knowledge (SEEK) [1] has been expressed by OWL into Kepler to enable automatic structural data transformation in the data flow among services [151].

Semantic web services (SWS) composition approaches, such as OWL-S and WSMO, have been utilized to handle the semantic and automatic composition of geospatial web services. OWL-S is used by [152] to automate the composition of geospatial web services. They propose a prototype extending the OGC CSW specification to include OWL-S. In this prototype, the authors conclude that ontologies describing data and services can support the automatic construction of geospatial workflows. However, some limitations to their prototype are noted. First, there are inconsistencies in spatial-referencing systems or geometries for input and output. Hence, a further revision is required for the functional semantics. Second, in their example, raster datasets are only considered and feature types (e.g. data scales and coordinate system) are dismissed.

In [153], a study presents two approaches for building geospatial workflows automatically: the abstract composition and the concrete composition of workflows. In concrete composition, they found that, unlike OWL-S, WSDL-S supports all

[1] http://seek.ecoinformatics.org

aspects of discovery composition and takes less effort to implement than OWL-S grounding. Nevertheless, they remarked that their approaches need a user to edit an abstract workflow design to construct the concrete composition and, thus, to perform a semi-automatic workflow design.

An approach for the semantically assisted design of geospatial workflows has been presented in [11]. Based on the semantic descriptions of resources, a prototype based on the Eclipse-based BPEL editors is designed to retrieve candidate resources. During the workflow design, calculations of the semantic similarity of candidate workflows could be triggered. However, because a test corpus of OGC web services does not exist, no test collection of geospatial services is developed to evaluate the precision of the semantically assisted catalogue service.

In recent times, AI-planning techniques have been considered to enable the automatic composition of geospatial services. In this regard, a notable study presents how OWL-S can be incorporated into an AI method to handle a semi-automatic geospatial service chaining [154]. In [12], a WSMO-based approach is proposed to integrate AI techniques for the automatic composition of geospatial and non-geospatial web services. By considering the additional semantics of geospatial data, such as geo-data quality requirements, a new approach using AI-planning methods is presented to improve the robustness of geospatial web services composition [155].

The possible composition approaches that can use existing semantic web service technologies such as OWL-S, WSMO, and WSDL-S to handle the semantics of geospatial service composition have been demonstrated in [156]. From the other side, he describes how data-type and service-type ontologies can be used to annotate the semantics of geospatial service operation. He also elaborates on possible automatic composition models. Nevertheless, only conceptual models are designed and no specific system or technology is presented to implement such approaches. Furthermore, execution semantics or semantic workflow design are not mentioned.

Although progress has been made with regard to the semantic composition approaches of geospatial services, in its current form it often deals only with service discovery. Furthermore, most of these approaches define semantics at a fine level of granularity of operations and queries. Thus, a methodology to describe and drive possible compositions of operations automatically is a critical requirement. Despite

the promising results of using SWS composition technologies (such as OWL-S and WSMO) and the adoption of OGC standards, the successful application of all these techniques for (semi-)automatic workflow composition depends on the provisioning of adequate meta-information about the involved technical entities (services, data types) of the target application. Furthermore, most of the existing geospatial software tools (either as freeware or commercial products) do not have standard interface protocols and are not OGC-compliant. These tools can be developed into web services that are not OGC-compliant. Therefore, more and individual technical efforts are required to develop OWL-S descriptions for these services.

In fact, there are still many challenges to handling the semantic composition of geospatial services and, therefore, carrying out automatic service composition. In particular, for real-world applications, the existing approaches still face some challenges:

- Formalizing of geospatial services for sustained service registry. Identifying all functional modules as services and then building the taxonomy to organize concepts of services [3].

- Addressing the execution semantics of geospatial services composition (workflow design).

- Handling QoS by considering different aspects of geospatial data quality that are related to data formats, coordinate reference systems, and spatial resolutions.

- Handling the performance of automatic service composition (workflow synthesis). For instance, decreasing the input of services and types to reduce the exploration time of adequate workflows.

- Lack of usable frameworks, such as graphical tools, to:

  - Simplify composition tasks tackled by domain experts by hiding the technical details of services and their implementation,

  - Build and integrate the required ontologies that rely heavily on domain experts. For instance, the handling of the semantics of user intents by

users who have no technical background in semantic web technologies and their use,

- Add semantics awareness into existing standards, such as OGC, and
- Let users express their constraints in a high-level way to compose services according to their preferences.

Chapter 4

Application Example: Geospatial Workflows for Climate Impact Analysis

This chapter shows how we addressed the challenges of geospatial service reuse and workflow design by using the XMDD paradigm supported by the jABC and jETI technologies. In concrete terms, it describes how we have made the functionality underlying the ci:grasp climate impact information platform available as easily reusable services, thus enabling users to design workflows in an agile manner.

The chapter is organized as follows: Section 4.1 presents an overview about climate impact applications related to ci:grasp. Section 4.2 describes a manual workflow design architecture with its components (data layer, service layer, and workflow design layer). The evaluation of using an eXtreme paradigm to improve the geospatial service reuse in climate impact applications is demonstrated in Section 4.3.

4.1 Climate Impact Analysis Applications Scenario

By means of geospatial application or GIS, climate impact analysis has been addressed as a specific topic in the broad field of environmental risk and impact

assessment. The impact of climate change is often assessed using a chain of scientific models that are not easily accessible and applicable. The results of various models are compared in a large community effort—for example, the ongoing Inter-Sectoral Impact Model Intercomparison Project [23]. Efforts are being made to design the models in a modular and reusable way [157]; however, a breakthrough has not been reached in the impact-modelling community. Further, as efficient communication of climate change insights to a broader audience, as well as to policy-makers and decision-makers, is both crucial and not trivial [158], climate information services are of high interest.

Analysing and assessing the potential impacts of climate change are crucial tasks in order to adequately plan and undertake adaptive measures [7]. These tasks require analysis processing and the integration of large and heterogeneous datasets. These analyses are particularly demanding issues because of the multi-scale and multi-objective nature of environmental modelling for climate change impact assessment [159].

The ci:grasp project[1], a web-based climate information system, aims to support decision-makers in developing and emerging countries to prioritize adaptation needs, as well as to plan and implement appropriate adaptation measures [160; 161]. Figure 4.1 gives an overview of the information presented on ci:grasp. The core contribution is to provide a web-based user interface that can interactively access information on the impacts of climate change. It models climate information and related knowledge in the form of text, maps, and graphs over a web page. SLR and more frequent extreme events causing climate change—such as flooding—present a high risk to human lives and cause massive economic damage. Thus, identifying vulnerable areas is the first step towards prevention measures [161].

SLR, changes in temperature and precipitation, and increased drought risk are the main impacts of climate change included on ci:grasp [161]. These impacts, as well as the more frequent extreme events expected under a changing climate, present a high risk to human lives and cause massive economic damage depending on the degree of global warming. Therefore, ci:grasp provides some related analysis, such as estimation of the potential of peri-urban agriculture for major

[1] http://www.cigrasp.org

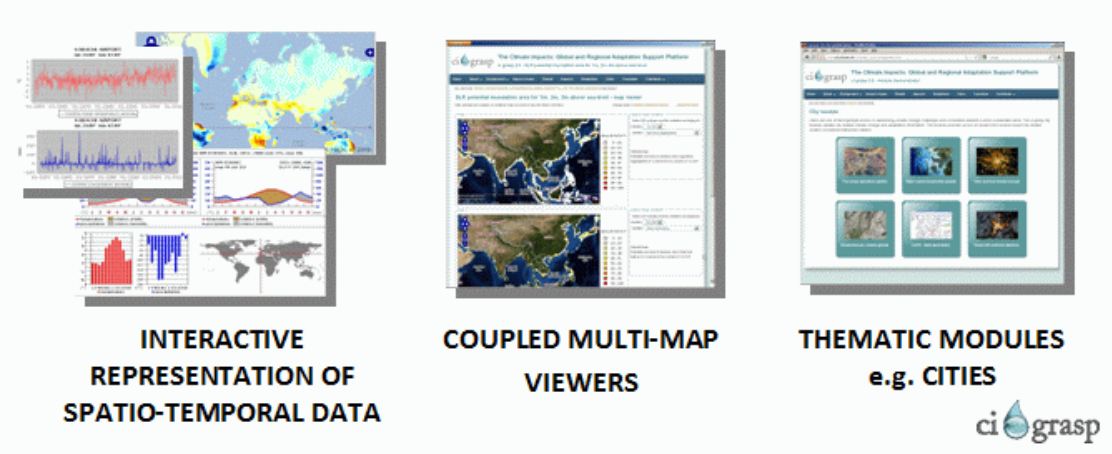


Figure 4.1: Overview of the information available on ci:grasp.

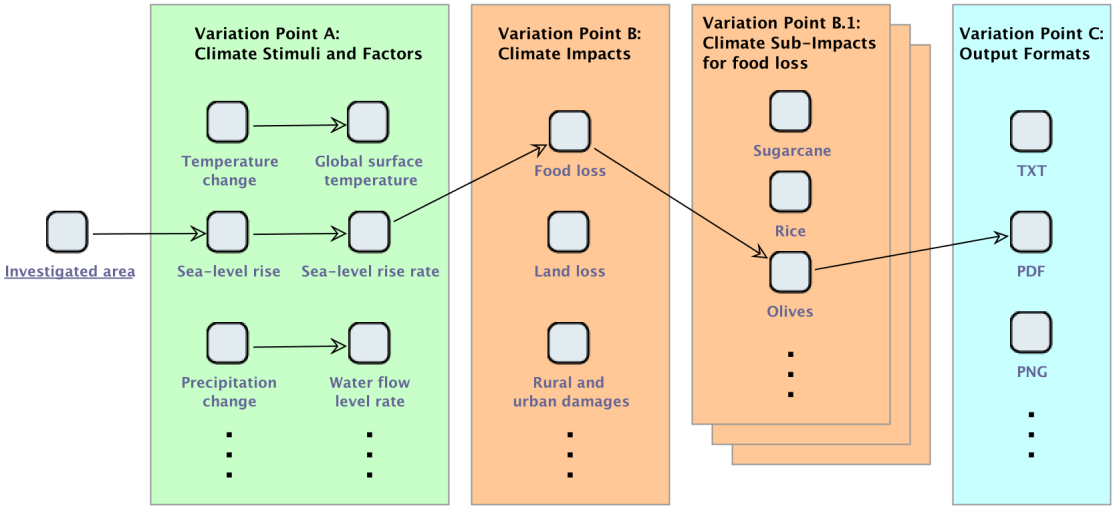


Figure 4.2: Variations in climate impact processes.

cities by determining their 'carrying capacity' and quantification of the adequacy of water resources for different user groups. ci:grasp provides users with climate impact-related information and knowledge. However, it is only based on predefined data, processes, and scenarios. Thus, users are unable to use their own data with a certain scale or with certain formats, nor to use processes in a flexible manner in order to define several scenarios. Furthermore, it is not possible for users to understand the workflow of the processes achieving the required tasks [24].

In order to depict some of these challenges, Figure 4.2 illustrates the complex-

ity of possible variations of climate impact analysis processes. For a particular investigated area, differently elevated data scales could be used to analyse the impacts of several climate stimuli, such as SLR, temperature change, and precipitation change. For a specific value of climate stimuli factors, various possible climate impact analyses are possible. For instance, based on a particular SLR rate, climate change is assessed with respect to the impacts of the potential loss of agricultural production, calories available, and the effect on food security, but also with respect to the properties of rural and urban damage functions. For each specific SLR impact (e.g. food loss), several sub-impacts are analysed depending on food type, potential time (2020, 2050, or 2080), a range of different climate scenarios, such as EHA2 and EHB2, and soil and terrain conditions [162]. Therefore, the possibility of variation analysis will be increased when considering the user preferences regarding user input and output data types and formats.

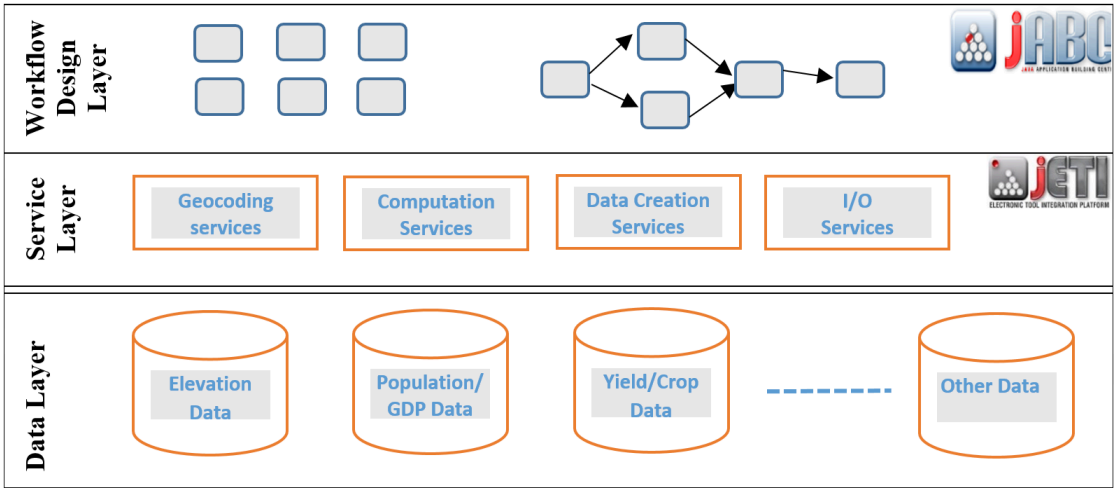


Figure 4.3: Layered architecture approach for manual geospatial workflow design.

4.2 Manual Workflow Design Architecture

To describe the components involved in the workflow development, a layered architectural organization is drawn (see Figure 4.3). This architecture is presented to depict how it is possible to use a diversity of geospatial datasets with different

types of services to design a workflow. From bottom to top, the three layers are described as follows:

4.2.1 Data Layer

Data is a crucial component when designing workflows and, often, scientific workflows are designed from the perspective of dataflow concepts. Climate datasets (geospatial and not geospatial data) exist in various formats, units, scales, and geo-reference systems. Therefore, providing users with a flexible way to handle the heterogeneity of geospatial datasets will increase the productivity of users or researchers working on the analysis processes. In this section, the variety of datasets used in the example of the climate impact analyses will be described.

In fact, the potential impacts of climate change are assessed with respect to the potential loss of agricultural production, calories available, effect on food security [163], and properties of rural and urban damage functions [164]. To this end, heterogeneous data (e.g. elevation, land use, gross domestic product [GDP], population density, yield, or water data) has to be used. The data exists in several formats (like GeoTIFF, ASCII, NetCDF, TXT, and CSV), at different scales (from 90m to 55km), and in different geo-referencing systems, requiring adequate integration and aggregation.

Table 4.1 describes six main dataset groups (elevation data, GDP data, population data, land use or land cover data, yield data, and water data). The water data used in the context of our example includes available water resources from the LPJmL model [172; 176], environmental water requirements [173], water quality [174], water accessibility [177], and agricultural water needs [175]. All datasets are pre-processed as GTiff raster data. The combination of these allows users to compute the water adequacy for the municipal, agricultural, industrial, and environmental sectors. Furthermore, food consumption data on a national level is available as a CSV table from FAOSTAT [178] and is used to compute the carrying capacities of cities.

In fact, the data also exists in different classes of types. For instance, to show the numerous types of datasets used in the example of climate impact analysis, Figure 4.4 presents a classification tree for the global information agricultural

Table 4.1: Main group of datasets.

Datasets Group	Description	Formats	Resolution	Source
elevation data (DEM)	the digital elevation model (DEM) can be used to identify land areas that are hydraulically connected to the sea and below a certain elevation	raster	$90m$	[165]
GDP data	include global gridded information about GDP	raster	$1km$	[166]
population data	include global gridded information and a table with 67,935 settlement points	raster, vector, and CSV table	$0.0417° \approx 5km$	CIESIN [167; 168; 169]
yield data	global information of actual and potential agricultural production provided by global agro-ecological zones (GAEZ), agricultural productivity	raster	$0.0833° \approx 10km$	CIESIN [170]
land use/land cover	a pre-processed MODIS land cover map, including 17 different land cover types, such as croplands and urban areas	raster, vector, and CSV table	$500m$	[171]
water data	include available water resources from the LPJmL model, water requirement, and agricultural water needs	raster, vector, and text files	$0.5°$	[172; 173; 174; 175]

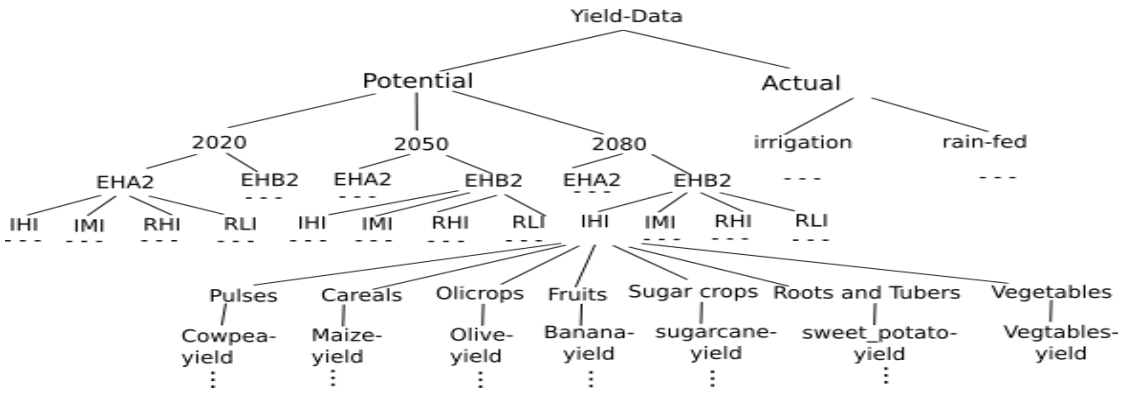


Figure 4.4: Yield data classification tree.

production of GAEZs. The data is classified into two major classes: actual and potential production classes. The actual production of several crops can be either rain-fed or through irrigation. The potential production is classified into several levels with regard to potential time (2020, 2050, or 2080), a range of different climate scenarios, such as EHA2 and EHB2, and soil and terrain conditions, such as Irrigate High Input (IHI), Irrigate Medium Input (IMI), Rain High Input (RHI), and Rain Low Input (RLI). Overall, there is information on the yield constraints, crop calendars, harvested area, and production potential estimates for 11 major crop groups, 49 major crops, and 92 crop types. Productivity estimates are made for rain-fed farming and several irrigation systems [162]. In addition to the predefined types or scales of data, other datasets can also be used.

4.2.2 Service Layer

According to the service orientation paradigm assumption, any kind of computational resource should be seen and handled as a service—that is, a well-defined unit of functionality with a well-defined interface—to provide a high level of abstraction and reusability (cf., e.g. [179]). In this section, we demonstrate how, by applying the concepts of service engineering and using jETI technology, we can make the available climate impact assessment tools based on ci:grasp accessible as remote services.

The process of turning arbitrary software components into proper services that are adequate, for example, for (re-)use in workflow management systems is referred

to as *servification* [24]. In the service layer, the *servification* processes take place for the tools that have been developed for climate impact assessment within the ci:grasp platform.

Several tools and applications have been developed to analyse the risk index of climate impacts, such as data creation, conversion, and visualization tools. In the context of this thesis, we used the scientific tools that are developed to address the analysis of the impacts of SLR, the potential of peri-urban agriculture for major cities by determining their 'carrying capacity', and quantification of the adequacy of water resources. These tools are used on the ci:grasp climate information platform. They are based on scripts in the GNU R language that comprises several tools for spatial analysis. The srtmtools package [180] used for the data analysis provides the methods required to produce results, as presented on ci:grasp. It combines various tools that are based on different packages. For instance, the raster package tool[1] [181] for data reading, writing, manipulating, analysing, and modelling of gridded spatial data, the Gdal tool[2] for data conversion [182], and other packages for data visualization, such as Png[3] [183] and plotGoogleMaps [184]. The cca package [185] implements the city clustering algorithm [186]. It is used to identify cites based on the urban class from land cover datasets and to join population information from a list of settlement points.

In concrete terms, the servification process of the climate impact assessment tools is performed as follows:

- First, understanding and analysing the scripts of climate impact assessment tools with the domain experts of the ci:grasp project to define the common GIS processes and application-specific processes.

- Second, decomposing the climate impact assessment tools into loosely coupled services. To ensure a great level of reuse for the services, the decomposition process performed a rigorous abstraction for the most frequently used process steps in various applications of climate impact assessment.

- Third, providing these services as autonomously running R scripts. In jETI,

[1] http://cran.r-project.org/web/packages/raster/
[2] http://www.gdal.org,https://r-forge.r-project.org/projects/rgdal/
[3] http://www.rforge.net/png

Table 4.2: Service library created for climate impact analysis.

Service	Description
Data manipulation services	
Load SRTM elevation data	Download the DEM for the selected area.
Load raster data	Load raster data from global map data.
Read data from a table	Read a file in table format and create a data frame from it.
Get GADM data	Get spatial country data from GADM database.
Cropping raster data	Crop or clip returns a geographic subset of an object as specified by an extent object
Masking data	Identifying areas to be included in analysis
Re-sampling raster data	Transfer values between non-matching raster objects
Raster projection	Get or set the coordinate reference system (CRS) of a raster* object.
Extent raster data	Define the spatial extent of objects of the raster data
Extract raster data	Extract values from a raster object at the locations of other spatial data.
Conversion services	Used to convert data formats and units.
Transformation services	Used to transform data projection and geo-referencing systems.
Trim	Shrink a raster object by removing outer rows and columns.
Join GRUMP point data	Join GRUMP point data with city clusters.
Computation services	
SLR applications	
Compute flooded areas	Compute the flooded areas for a region based on its DEM.
Compute population or GDP at risk	Estimate the number of people or GDP that would be affected.
Compute potential land loss (ha)	Estimate the area that will be potentially inundated.
Compute yield loss	Compute actual and potential production value affected in USD.
Compute land loss classes	Define the type of land affected, from 1–16 different land types.
City capacity applications	
Compute bounding box	Compute bounding box for a certain city.
Compute carrying capacity	Estimate the carrying capacity for a certain city.
Compute city clusters	Estimate the population of the cluster (multiple cities).
Compute peri-urban area buffer	Create buffer with specific size and exclude water bodies, as well as other urban and peri-urban areas.
Compute daily person kcal	Estimate the kcal need of one person per day for the given country.
Water adequacy applications	
Calculate water fuzzy values	Use the method of fuzzy logic to assess the state of water conditions.
Compute water adequacy	Use to compute several aspect of water adequacy.
Compute water quality	Use to compute several aspects of water quality.
Compute water livelihood	Use to compute several aspects of water livelihood.
Output services	
Generate interactive map output	Generate an interactive map output using the Google Maps API[1].
Generate static map	Create a static map in PNG or PDF format.
Produce GeoTIFF output	Create a geo-referenced file (GeoTIFF, ASCII).
Produce text output	Create a text file containing some summary and statistical information.

a description for each service, equipped with well-defined inputs and outputs, is configured on the server and connected with the corresponding script file. After that, services are generated automatically into SIBs; therefore, they can easily be consumed as remotely accessible services on a jETI server that are adequate, for example, for (re-)use in workflow management systems.

Based on this way of the servification process, around 65 services have been

created for different climate impact analysis process steps. These services are listed in Table 4.2 and can be categorized into three groups as follows:

- *Data manipulation* includes data loading and re-sampling, transformation, and conversion services. It also comprises common geospatial services, such as clipping and masking.
- *Computation* includes the domain-specific computation services that are related to the climate impact analysis of SLR, city capacity, and water adequacy applications.
- *Output generation* includes services for the creation of PNG, PDF, TXT, GeoTIFF, or ASCII output files, and for result visualization in an interactive map, such as Google Maps.

In addition to these services, the service layer comprises some web services for geo-coding and other SIBs provided by jABC to support data input or output, in order to evaluate conditions and for basic user interaction[1].

4.2.3 Workflow Design Layer

At the *workflow design layer*, the composition of the services into climate impact analysis workflows takes place. In this section, we demonstrate how the jABC process modelling and execution framework [27] supports the service reuse and agility design of climate impact analysis workflows. As jABC aims to involve the application experts throughout the development process [187], the power of the agile workflow development style will be demonstrated by addressing some use cases of climate-related impact applications.

The use cases of workflows illustrate how the newly created domain-specific services described in the previous section are leveraged to a user-accessible level and composed with the standard SIB libraries provided by the jABC into workflows tailored to their specific needs. Through the examples, we will show how the rapid development and delivery of service composition, differences in the abstraction

[1] http://jabc.cs.tu-dortmund.de/sib/

level, and intuitive graphical interface in jABC enable users to construct different workflows for climate impact assessment in an agile workflow-based way.

In the following, three main use cases of climate impact applications are presented: analysis of potential SLR impacts, estimation of the potential of peri-urban agriculture for major cities by determining their 'carrying capacity', and quantification of the adequacy of water resources for different user groups. For complete applications, we designed around 92 different models, composed of more than 321 SIBs and spanning three hierarchy levels.

SLR Impact Analysis Workflow

The primary task of SLR impact analysis is to identify the impact of the submergence of land. This is achieved by identifying vulnerable areas from a digital elevation model—e.g. the Shuttle Radar Topography Mission (SRTM) terrain elevation data [165]—to determine the potential requirements for prevention measures. This can be performed by using the srtmtools package and based on the burning algorithm presented by [186] to download necessary data from the CGIAR-CSI database [165] automatically. After that, there is the combination of the vulnerable areas with the data of interest—for example, the data specifying the type of flooded land or the potential GDP, yield, and calorie loss. The srtmtools package provides a re-sampling function that can be used to deal with different resolutions of the datasets, which gives the exact amount of flooded land per grid cell for a different resolution [28].

By reusing created services, we used jABC to design a simple workflow to assess the impact of SLR on the agricultural yield loss for a region to be selected by the user. The process is as shown in Figure 4.5, from left to right (the SIB with the underlined name denotes the starting point): (1) Select the working directory for input and output data; (2) define the investigated area by coordinates or names; (3) download the digital elevation model of the selected area; (4) enter the magnitude of SLR; (5) compute the flooded area; (6) compute the yield loss due to the flooding; and (7) generate an output file with results in the PDF format. The functionality that is actually a composite of several services (SIBs) in a separate (sub-)model is marked by a green circle. For instance, SIB (6)

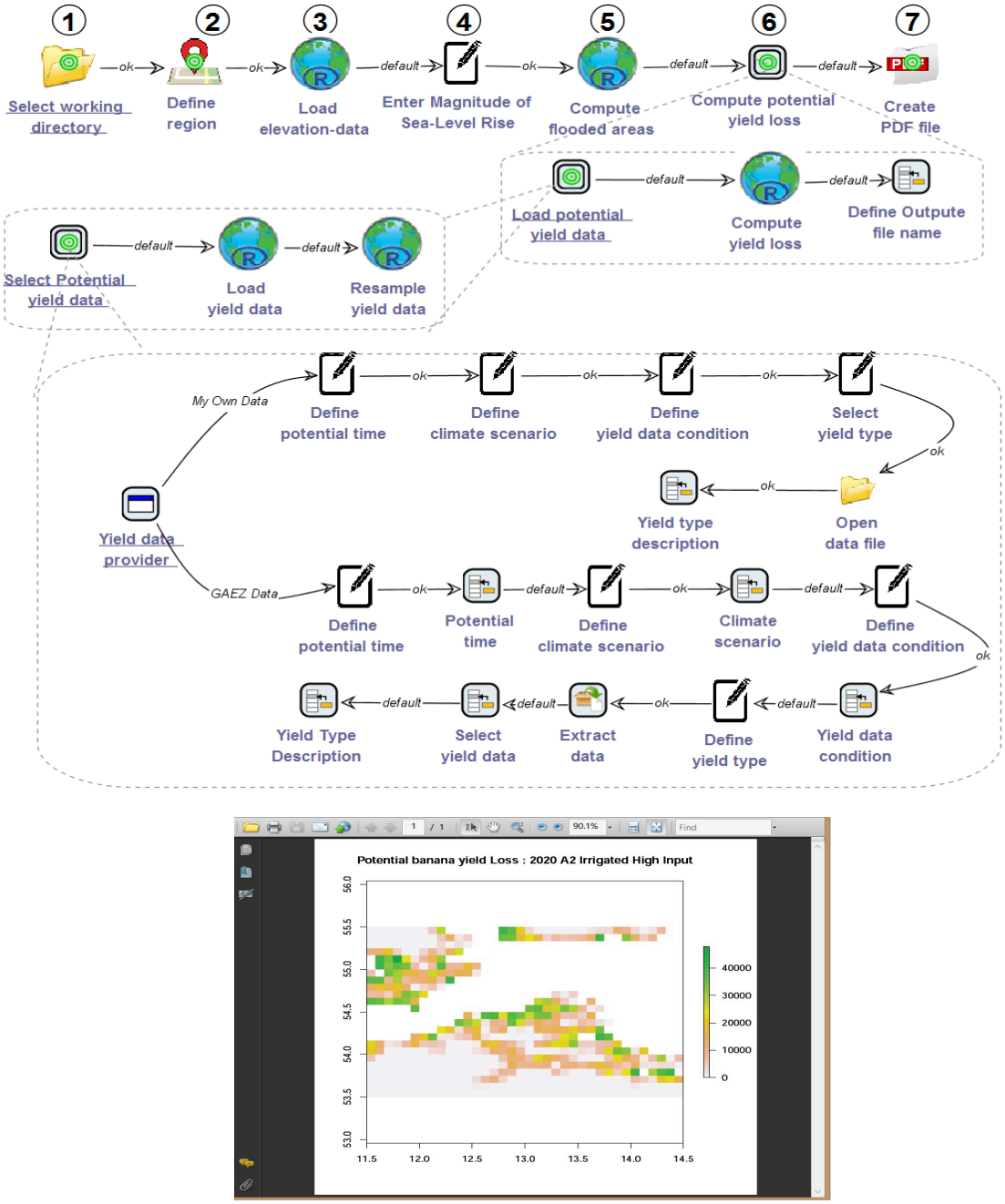


Figure 4.5: Workflow for yield loss assessment.

is a composite service that allows for the computation of several types of yield loss for different climate scenarios, which again make use of other (sub-)models. Organizing workflow applications under different levels of abstraction by enabling a hierarchical modelling style will support the flexibility of reuse from coarse-

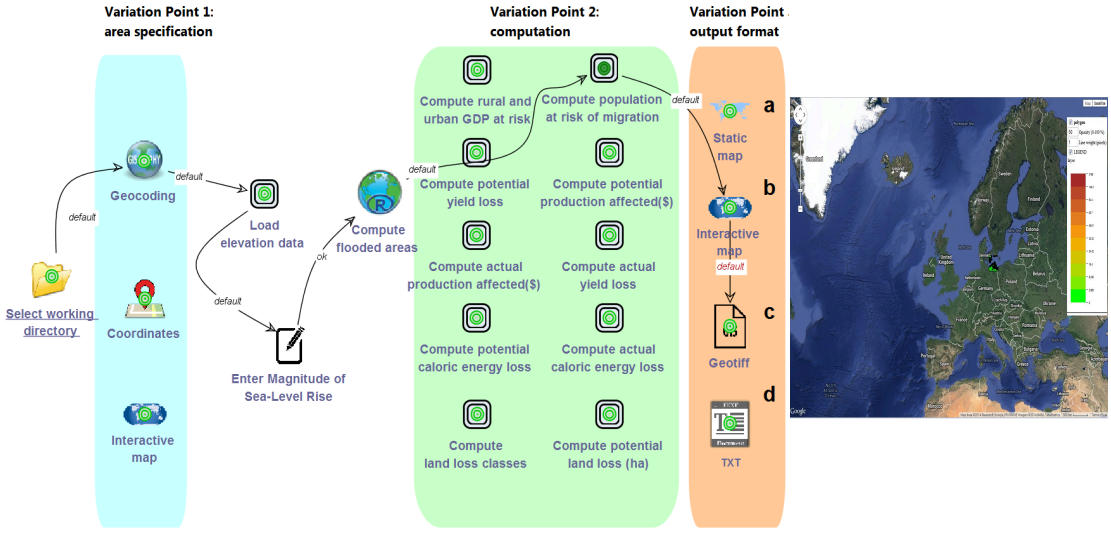


Figure 4.6: Workflow for SLR impact assessment with preconfigured variation points.

grained and more conceptual views at the higher levels, down to fine-grained and more technical views at the lower levels. For example, in the middle of Figure 4.5 is the concrete realization of the flexible selection of the yield data to be used: The user can select if he/she wants to work with her/his own datasets by defining the characteristics of the data (e.g. climate scenario and yield data aggregation condition) or based on the existing GEAZ datasets structure as described in Section 4.2.1.

In contrast to the previous example, which aims to show how the user can reuse services in a flexible manner to address a concrete application, the second example presents variations of preconfigured workflows for several SLR impact assessment applications. As shown in Figure 4.6, preconfigured variations are used to provide a selection of predefined options:

- *Location-determining variation* allows for easy modification of the definition of the region considered for the analysis. Besides entering the coordinates directly (as in the basic example), it is also possible to use the geo-coding service by entering the name of a place or an address that is then used as the centre of the region.

- *Analysis objectives variation* includes a collection of different computations,

summarized in Table 4.3, that can be selected according to the concrete objectives when assessing SLR impacts. Some of these computations use the same underlying computation service, but their interfaces are tailored to concrete use cases. For example, to compute the yield loss, two computation variations are available (compute potential and actual yield loss), corresponding to the analysis objective and the respective type of data. As presented in Figure 4.5, all these computations are realized by separate workflow models.

- *Output generation variation* provides SIBs for creating different output formats. Users can select one or more (by including a sequence of output-generating SIBs) formats for the presentation of the final results, including (a) static maps in different formats (JPEG, PDF, PNG, PS), (b) an interactive map using the Google Maps API[1], (c) a text file containing a summary, and (d) a geo-referenced file (GeoTIFF, ASCII) that can be used for further external GIS processing.

Table 4.3: Objectives of assessing SLR impacts.

Computation (VP2)	Description
compute rural and urban GDP at risk	focus on potential economic damage in coastal communities
compute population at risk of migration	focus on the number of people that would be affected
compute actual or potential yield loss	compute actual or potential production value affected in USD
compute actual or potential production affected ($)	focus on the economic value of the agricultural loss
compute actual or potential caloric energy loss	focus on the actual and potential peoples' annual diets lost
compute land loss classes	determine 1–16 different land types
compute potential land loss (ha)	determine the area that will be potentially inundated

Based on these preconfigured variations, flexible adjustment of the SLR impact assessment at the user level is possible. For instance, users can easily build variants of the workflow corresponding to their preferences: Users only need to change

[1] https://developers.google.com/maps/

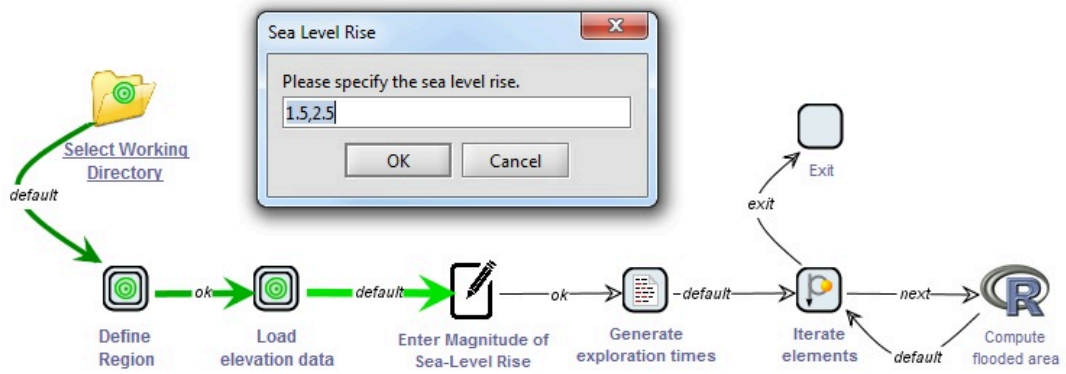


Figure 4.7: Workflow for the iteration over different values of SLR.

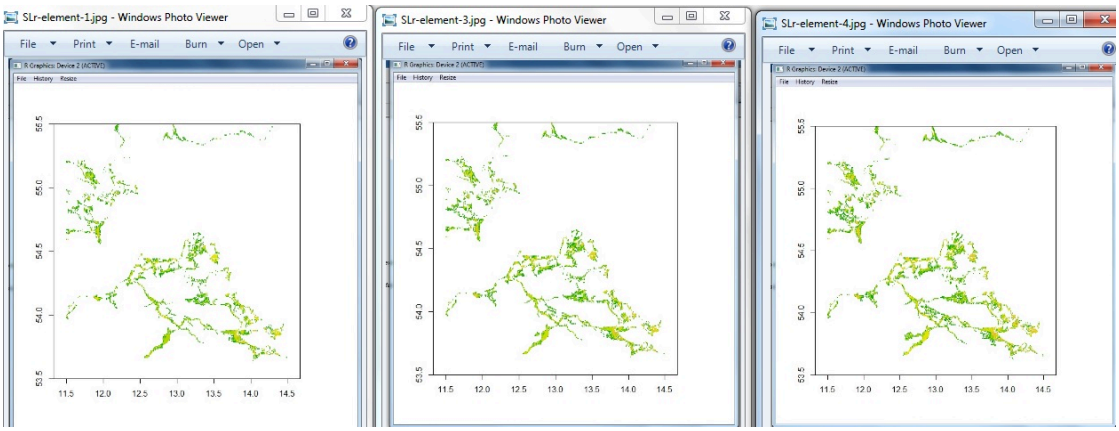


Figure 4.8: Exploration of flooded areas with different SLR values (2, 3, 4).

the execution path in order to include additional options from the preconfigured variations, simply by dragging the connecting branches to other SIBs. For example, the connections in Figure 4.6 show that the user is interested in determining the region by entering the address or name of a place to assess the number of people that could potentially be affected by a submergence of land triggered by a certain magnitude of SLR. Next, after the loading of the elevation data, entering the magnitude of SLR to be considered, and computation of the flooded areas, the computation of the potentially affected population is performed. Finally, the outputs are shown in Google Maps and a GeoTIFF file is created.

Like other scientific application domains, supporting the reproducibility of experiment results in a climate impact analysis process is a crucial task. Therefore, jABC supports users with a mechanism for parameter exploration to address the inherent uncertainties in assessing future climate change. Thus, it provides support for the common approach of generating and comparing projections for different scenarios—for instance, exploring the magnitude of SLR, different land use datasets, and different land loss classes.

Having an explicit structure of workflow instances with an executable specification and published results enables users to run and explore a variety of experiments just by changing the parameters. An example of a workflow that performs a parameter exploration by iterating over different magnitudes of SLR is depicted in Figure 4.7. Here, the user enters not only a single SLR magnitude, but a comma-separated list of potential increases in the sea level. This step of the workflow execution is shown in Figure 4.7. The workflow iterates over the list elements, computing the potentially flooded area for a different magnitude of SLR in each iteration. As shown in Figure 4.8, the user obtains a set of maps, each representing a different scenario. Although this is only a simple example, it demonstrates well the possibility to use the jABC to perform an analysis of the parameters automatically, which, in this case, provides a convenient way to generate outputs for multiple scenarios in order to deal with the uncertainty relating to the magnitude of SLR under climate change.

City Module Workflows

Another use case related to the global and regional adaptation to climate impacts is the identification of the global potential of local peri-urban food production. The *city module* of ci:grasp provides, among other topics, a dataset to identify the potential of peri-urban agriculture for 2,383 cities worldwide by determining their carrying capacity.

The carrying capacity analysis is based on the combination of several datasets as well as various food and yield scenarios. Given the large datasets—including different classification schemes of land cover types—providing users with flexible workflows to perform different analyses for cities worldwide is a crucial task. In

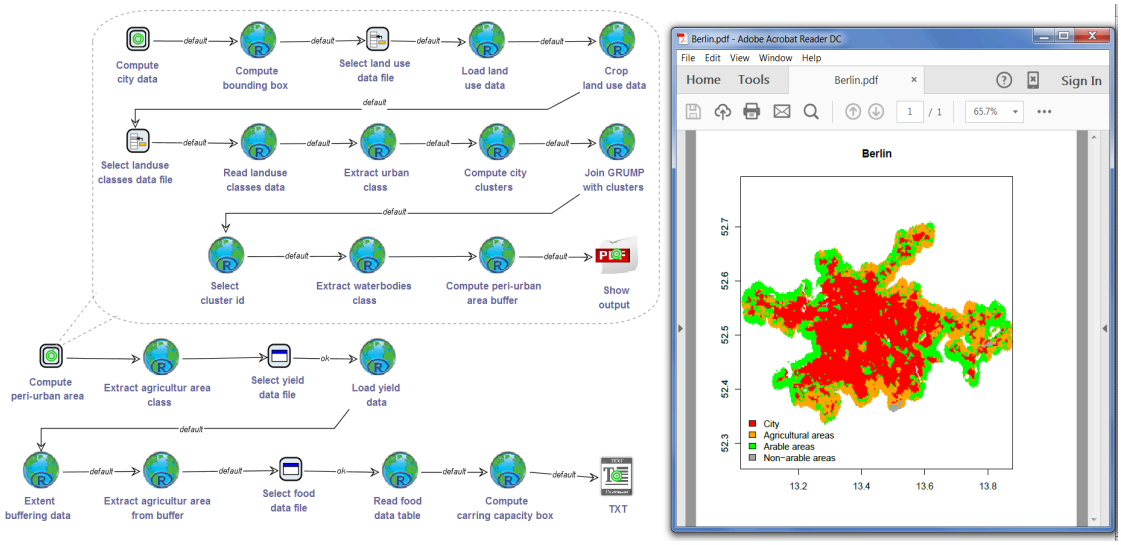


Figure 4.9: Compute peri-urban agriculture and carrying capacity for a city.

this section, we show how users can easily reuse the services described in Section 4.2.2 to assess the potential of peri-urban agriculture for a particular city based on certain food and yield scenarios, and to determine the carrying capacity of the investigated city.

The example of urban carrying capacity depicted in Figure 4.9 presents a workflow to identify the potential of peri-urban agriculture for a certain city. By starting from the upper left, users should define the investigated city by name (in this example, the city of Berlin in Germany). Based on the coordinates of the chosen settlement point, an area of interest is defined as a bounding box and the corresponding land use or land cover data will be loaded. After that, the urban class will be extracted from the land cover data and the cca will perform the identification of the corresponding urban cluster. The next step is the creation of the peri-urban area as a buffer around the urban cluster, excluding water bodies. Finally the peri-urban agriculture area is computed and stored in a file or generated as a map in a PDF file.

Computing the carrying capacity for a specific city can be easily done by reusing the potential peri-urban agriculture area identification workflow described above as a composite service (marked by a green circle), as shown in the button of Figure 4.9. By extracting the yield data from agricultural areas within the buffer, and by loading population and food data to identify the kilocalorie need for each person

per day, the carrying capacity for the investigated city can be computed. Note that the functionality of the carrying capacity example is more complex and is then defined by a separate (sub-)model that is represented by a building block, with the SIBs marked by a green circle.

Water Resource Adequacy Workflows

Following the approach presented in [188], the scenario described in this section assesses the water adequacy for three major water users—namely, the domestic, agricultural, and industrial sectors—as well as the total water adequacy using fuzzy logic. This approach can be applied to 174 countries to compute the water adequacy for a baseline (1981–2010) and two future scenarios (2011–2040, RCP 2.6, and RCP 8.5) based on the output of two climate models. These results should help to determine the main limiting factors relating to adequate access to water in a specific region and could help select the most efficient countermeasures.

Providing users in the different sectors with flexible workflows for the analyses will improve their productivity and will support them to consider additional requirements, such as climate models and population growth, in order to assess the adequacy of water resources. The exemplary water adequacy analysis workflow depicted in Figure 4.10 computes the water adequacy for South Africa. Users should define the path to the files that include the water data of several scenarios (RCP 2.6 and RCP 8.5 of 1981–2010 and 2011–2040). Next, the workflow loads and stores files as a list of raster data. Iso3 code and administrative level are required to obtain the country data from GADM [189]. For the chosen administrative regions, fuzzy logic is used to model water adequacy based on available water resources from the LPJmL model [172], as well as environmental water requirements [173], water quality [174], water accessibility [177], and agricultural water needs [175].

Several data creation services, as listed in Table 4.2, are reused to read and load various datasets as well as shrink the data to the defined region. Additional new services are introduced to convert the total data to per capita values and for the computation of fuzzy values. Figure 4.10 shows the exemplary results of these steps for water adequacy in the municipal, agricultural, industrial, and environmental sectors.

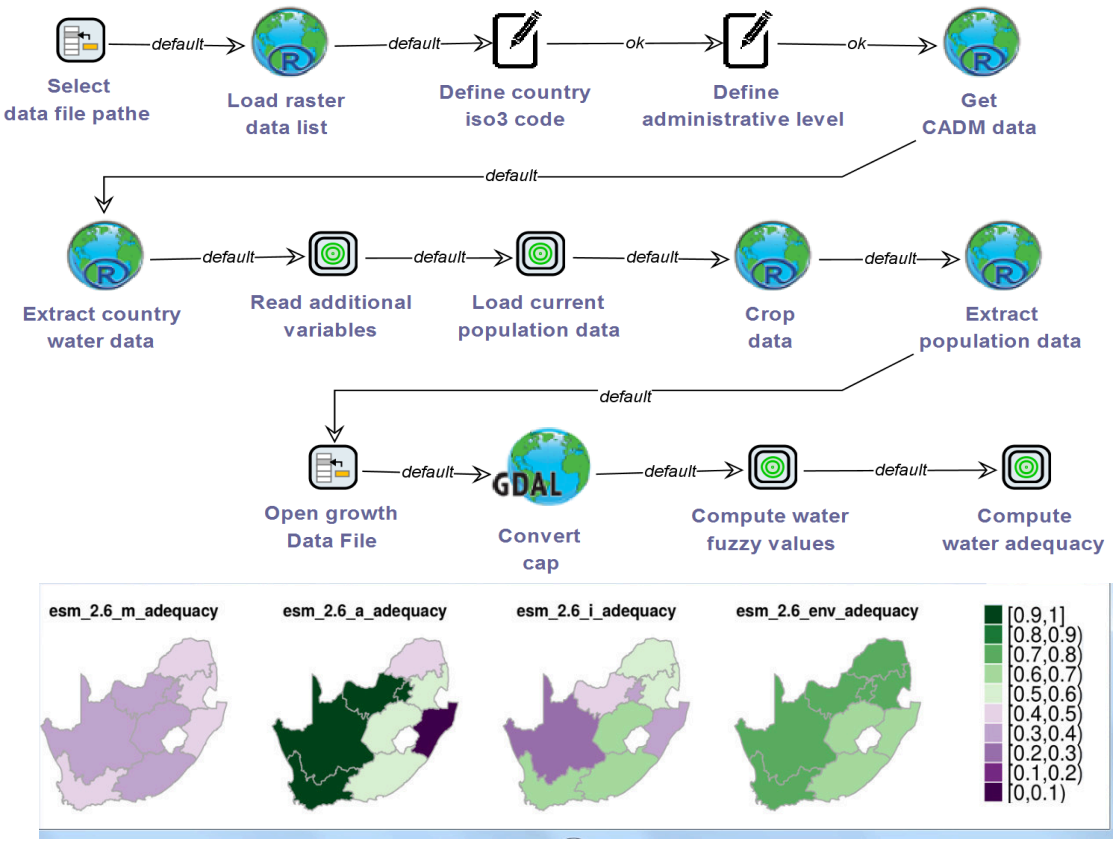


Figure 4.10: Compute water resource adequacy in South Africa.

4.3 Handling Geospatial Service Reuse

The geospatial services have their own characteristics, such as complex processes and big datasets, which hamper service reuse. Therefore, we believe that, to increase service reuse, a significant factor would be facilitating service reuse and making the composition easy and flexible for a wide range of communities and people, so that the scientific community (e.g. geospatial application experts) can use and understand service principles and build applications based on service composition. This section demonstrates how the XMDD paradigm, supported by jABC and jETI technologies, makes an essential contribution to the increasing service reuse of climate impact analysis processes. In concrete terms, Figure 4.11 presents an approach to evaluate how the challenges of geospatial service reuse can be addressed from three perspectives: (1) *servification*, by turning basic components into reusable pieces of functionality; (2) enabling flexible and easy service reuse

to compose services in agile workflows, which frees end-users from the burdens of learning programming or scripting languages and other required technologies to design and adapt workflows; and (3) offering a suitable environment to handle comprehensive geospatial processing by supporting remote execution and the integration of services.

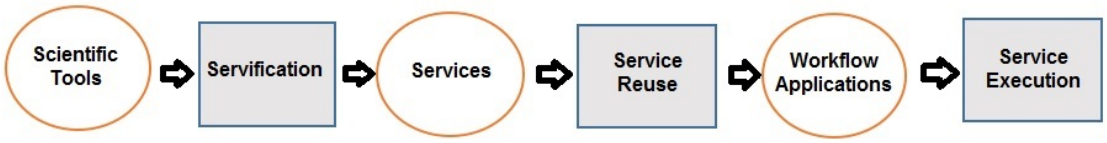


Figure 4.11: From geospatial tools to running workflows.

To assess the number of service reuse instances of created workflows for climate impact use cases (SLR impact analysis, city module, and water resource adequacy), we used the jABCstats framework [190]. For 13 of the workflows, Table 4.4 shows the frequency of reuse of the created services (see 4.2.2. According to the statistical values, these services have been reused 248 times in total for only 11 workflow variations. However, several variations of workflows can be designed based on analysis objectives and user preferences of data input.

The initial statistical results of service reuse reflect that the services have contributed to a significant number of reuses in the different workflow applications. Not surprisingly, data manipulation and output generation services are reused for all climate impact-related applications. Note that these services could also be reused in other analyses of climate change drivers included on ci:grasp, such as changes in temperature and precipitation, and increased drought risk, and in other risk analyses related to climate impacts. Furthermore, the services of data manipulation and of output generation and visualization are more likely to be reused in the geospatial application domain in general.

We are convinced that performing rigorous servification, and providing an easy and flexible way to compose services in geospatial applications significantly improves their reuse. However, geospatial services deal with large datasets and need comprehensive computing resources. Thus, even runtime performance and memory requirements should be considered when working on improving the reuse of geospatial services. In the examples used in this work and as mentioned in Sec-

Table 4.4: Frequency of service reuse.

Application	No. of Reuses of Services	No. of Workflows
Data manipulation services		
SLR applications	55	7
City module applications	33	2
Water resource applications	47	4
Computation services		
SLR applications	18	7
City module applications	10	2
Water resource applications	20	4
Output services		
SLR applications	35	7
City module applications	10	2
Water resource applications	20	4

tion 4.2.2, the created services are based on several packages and use a diversity of datasets (e.g. elevation, land use, population density, or yield data). Consequently, these packages and data and the pre-configuration corresponding to the operating system platform are required to perform the execution of services. The jETI platform offers a lightweight remote component (tool) to further simplify the integration and execution of software tools. It can be seen as a tool that enhances other tools and frameworks by the integration, organization, and execution of remote functionalities, so that users do not have to deal with the required configuration to execute the services. In our case, using jETI to integrate and execute geospatial services makes it possible to deploy these services in the cloud, which is especially useful for resource-intensive computations.

On the jETI server, script files for created services are installed and wrapped to enable convenient automated invocation. The required configuration includes the installation of the GNU R language and packages such as Raster, Rgdal, ClassInt, PNG and plotGoogleMapall are hidden from users. The jETI server itself runs in a virtual machine image based on a Debian Linux operating system. Managing the underlying infrastructure can be an issue as well. Hence, we follow the recent

trend of using cloud technology to host our server and services. The cloud directly addresses cost reduction and allows the rapid deployment of new services in a matter of minutes. Geospatial application usage is constrained by factors such as excessive cost, unnecessary complexity, and lack of accessibility. Thus, as shown in Figure 4.12, in this solution, the users can benefit from cloud technology by hosting jETI services in the cloud server and then submitting jobs of workflow execution to the cloud server. This enables users to benefit from the advantages of the cloud, such as resource scalability and data availability, whereby other users can run their own workflows.

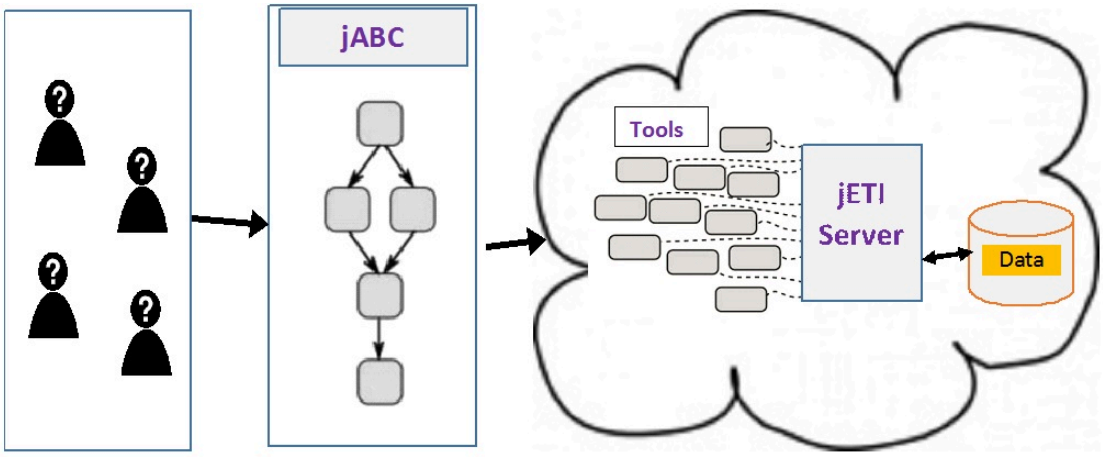


Figure 4.12: Interaction of users with the jABC and jETI environments.

Chapter 5

Semantic Domain Modelling of SLR Applications

The previous chapter shows how users can use XMDD technologies to create a manual workflow design in a flexible manner. However, end-users, such as domain experts, face challenges in constructing workflows to match the multi-scale and multi-objective nature of environmental modelling for climate change impact assessment. In principle, a large number of possible workflows can be designed for SLR impacts. Hence, users cannot manually discover all the services and data available and design the most suitable workflow. For example, based on a certain value of SLR, as shown in Figure 5.1, for a specific climate SLR impact, such as food loss, there are different sub-impacts depending on the food type (e.g. sugarcane, rice, or olives). The analysis of food loss varies corresponding to aim of the analysis (production loss in USD , caloric loss or yield loss in tonnes). Considering additional constraints, such as varying values of SLR and the climate time scenario, various data input and output formats and units increase the complexity of obtaining an adequate design of the workflow.

Therefore, constructing workflows manually and discovering the proper services and data that match user intents and preferences will most certainly be time-consuming, particularly for users who have insufficient knowledge in the domain. The automatic or semi-automatic workflow composition approaches have proven to be a solution for overcoming these challenges. We believe that a modelling domain

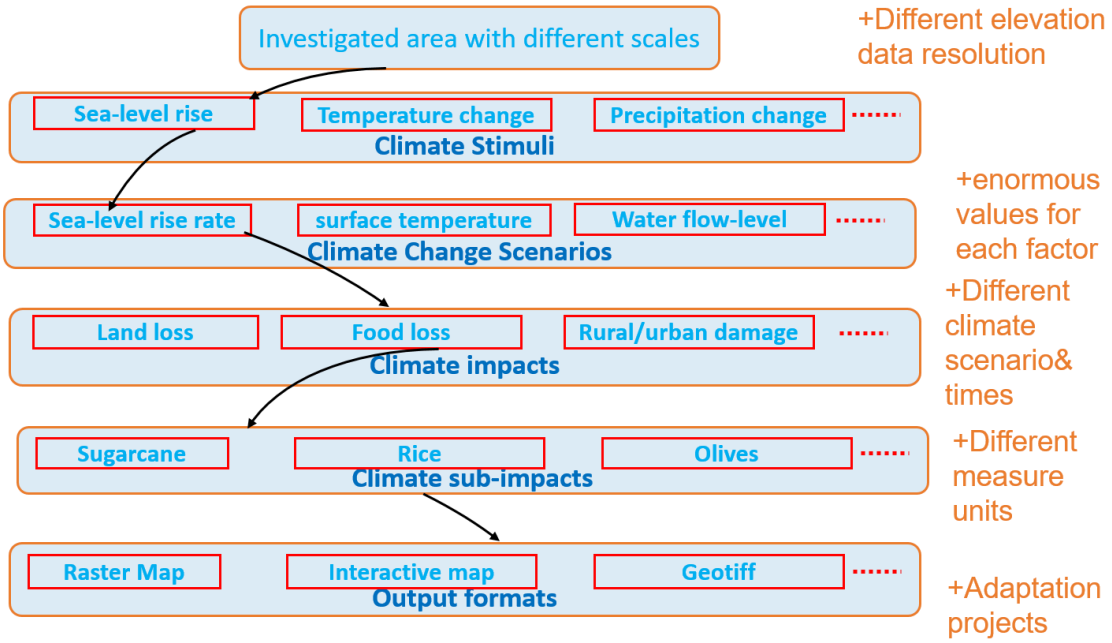


Figure 5.1: Configuration complexity example of climate SLR impact analysis.

application for describing workflow resources (data and services) syntactically and semantically is a solid basis for obtaining an adequate workflow design.

As described earlier, modelling a domain with PROPHETS aims to achieve an adequate workflow design by providing semantic meta-information about the services and ontology models for types and services classification in the domain. Thus, in the SLR application example, we realize the requirements originating from the application experts that define the system functions. Working in an agile development style could help analyse current processes and data use to assess the impacts of SLR and perform servification, thus finally designing manual SLR workflows (see Chapter 3). This provides us with adequate knowledge to model the domain of SLR applications.

In the following, we describe the domain modelling for the SLR application example—that is, the provisioning and the description of the services, the design of taxonomies for services and data types, and the formulation of constraints.

5.1 Services Interface Description

After turning the SLR tools into services adequate for (re-)use as SIBs in the jABC framework, the next crucial task for the domain modeller is to provide the services with semantic meta-information. In PROPHETS, as elaborated in Chapter 2, in terms of the domain vocabulary, the service interface description is accomplished by defining a symbolic name for the service and its input or output types.

Technically, we created adequate modules to represent the service's meta-information for available SLR services. The information includes appropriate symbolic names for the service entities and data types in the domain. Then, this information is stored in a separate file, `modules.xml`. Each module is linked to a concrete SIB that provides the implementation, but the module's names and the names used to describe input and output data types for the synthesis algorithm are symbolic.

This makes it possible to use one's own domain-specific terminology for the module descriptions. It also allows for polymorphism in the sense that one SIB can be used by several modules. This is particularly useful for SIBs that provide quite generic functionality, and where several modules with specific functionality can be defined based on the same underlying implementation.

In addition to the modules, for SIBs that were created specifically for the SLR application example, we defined different modules for the standard SIB libraries of jABC. Table 5.1 presents the names, descriptions, and input or output information for services that were used to read, select, or enter data inputs. While *ShowInput-Dialog and ReadFile* are standard SIB libraries of jABC, the *Read-from-CSVfile* is created in the context of this project to read data from CSV files. As these SIBs are general-purpose services for data input, we defined several modules to describe them.

Data manipulation services, which are generic in geospatial domain applications, are also described by creating several modules. For example, as shown in Table **??**, five modules refer to the *Rasterdata-loading* SIB—namely *load-elevation-data, load-population-data, load-GDP-data, load-landuse-data, and load-yield-data.* As another example, seven modules refer to the *Resampling* SIB—namely *ResampleResoluation, Resample-landuse-data, Resample-population-data, Resample-*

Table 5.1: SLR data input services interface description.

Service Name (SIBs)	**Module Name**	**Inputs**	**Outputs**
ShowFileChooser	SelectRasterdatafile		Rasterdatafile
	SelectElevationdatafile		Elevationdatafile
	SelectPopulation datafile		Populationdata file
	SelectGDPdatafile		GDPdatafile
	Selectlanduse datafile		Landusedatafile
	SelectYield datafile		Yielddatafile
ShowInputDialog	Enter-coordinates-values		Coordinates
	Enter-magnitude-of-sea-level-rise	SLR	
	Enter-landclass-number		Class-number
ReadFile	Read-CRS-GRS80		GRS80
	Read-CRS-WG384		WG384
	Read-Yieldfile		Yield-CSVdata
Read-from-CSVfile	Read-Kcal-values	Yield-CSVdata	Kcal-value
	Read-Kg-price	Yield-CSVdata	Kg-price

GDP-data, Resample-yield-data, Resample-yield-agriculturaldata, and Resample-flooded-yield-data. It is also possible to define more modules that refer to *Rasterdata-loading, Resampling, Clipping, or Making*. However, we only describe the services used in the context of our SLR workflow example.

We would define just one module for services with a highly specific functionality (e.g. computational services). As shown in Table **??**, all SIBs are domain-specific services tailored to analyse SLR impacts. Depending on the rural and urban infrastructure that is connected to the coastal area (e.g. shelters, energy production, and sanitation infrastructure), the *Compute-Rural-Urban-risk* SIB can be used to compute and analyse various types of damage. It is also used to compute people

Table 5.2: SLR data manipulation services interface description.

Service Name (SIBs)	**Module Name**	**Inputs**	**Outputs**
Load SRTM elevation data	Load-SRTM-elevation-data	Coordinates	SRTM-data
Rasterdata-Loading	Load-elevation-data	Elevationdatafile	Elevationscale
	Load-population-data	Populationdata-file	Populationdata
	Load-GDP data	GDPdatafile	GDPdata
	Load-landuse-data	Landusedatafile	landusedata
	Load-yield-data	Yielddatafile	Yielddata
Create Master Resolution	Createmaster90	Rasterdatafile	Raster90
	Createmaster60	Rasterdatafile	Raster60
	Createmaster30	Rasterdatafile	Raster30
Clipping	Crop-landusedata	Landusedata, SLRlandloss	Croppedlanduse-data
Resampleing	ResampleResoluation	Elevationscale, *Datascale*	Elevationusr
	Resample-landuse-data	SLRlandloss, Croppedlanduse-data	SLRlanduse-sample
	Resample-population-data	SLRlandloss, Populationdata	SLRpopulation-sample
	Resample-GDP-data	SLRlandloss, GDPdata	SLRGDP-sample
	Resample-yield-data	SLRlandloss, Yielddata	SLRYield-sample
	Resample-yield-agriculturaldata	SLRYield-sample, Agriculture-area	ResampledAgri-data
	Resample-flooded-yield-data	Yielddata, Agri-floodedarea	Resampledflooded-yielddata
Masking	Masklandusedata	Croppedlanduse-data, SLRlanduse-sample	Maskedlanduse-data
Convert-Raster-to-Vector	ConvertVector-to-RasterGDP	GDPdatafile	GDPdata
	ConvertVector-to-Rasterpopulation	Populationdatafile	Populationdata
	ConvertVector-to-Rasteryield	Yielddatafile	Yielddata
ConvertKgtoUSD	ConvertKgtoUSD	Yielddata, Kg-price	Production-data
ConvertKgtoKcal	ConvertKgtoKcal	Yielddata, Kcal-value	Caloric-data
Transformation	TransformToEPS	*CRSdata*, *Output-data*	Result

Table 5.3: SLR computation services interface description.

Service Name (SIBs)	**Module Name**	**Inputs**	**Outputs**
Compute-flooded-area	Compute-flooded-area	*Elevationdata* , SLR	SLRlandloss
Compute-Rural-Urban-risk	Compute-population-risk	Populationdata, SLRpopulation-sample	SLRpopulation-risk
	Compute-GDP-risk	GDPdata, SLRGDP-sample	SLRGDP-loss
Compute-landloss(ha)	Compute landloss(ha)	*Elevationdata*, SLR	SLRlandlossha
Compute-landloss-class	Compute landloss-class	*Elevationdata*, SLR, Classnumber	SLRlandclassloss
Identify-AgricultureArea	Identify-AgricultureArea	Croppedlanduse-data	Agriculturearea
Identify-Flooded-AgricultureArea	Identify-Flooded-Agriculturearea	Maskedlanduse-data	Agrifloodedarea
Compute-yieldloss	Compute-yieldloss	Yielddata, ResampledAgridata, Resampledflooded-yielddata	SLRyieldloss
Compute-productionloss	Compute-productionloss	Production-data, ResampledAgridata, Resampledflooded-yielddata	SLRproduction-loss-USD
Compute-caloric-energyloss	Compute-caloric-energyloss	Caloric-data, ResampledAgridata, Resampledflooded-yielddata	SLRCalories-loss

Table 5.4: SLR output generation services interface description.

Service Name (SIBs)	**Module Name**	**Inputs**	**Outputs**
Show-google-map	Show-google-map	Result	Google-map
Generate-Png-file	Generate-Png-file	Result	Png-file
Generate-Pdf-file	Generate-Pdf-file	Result	Pdf-file
Produce-GeoTIFF-data	Produce-GeoTIFF-data	Result	Geo-referenced-file
Produce-Text-data	Produce-Text-data	Result	Text-file

at risk of migration. Therefore, we can define several modules to describe it.

Note that some services seem to have the same input data types but, principally,

they share the same class of data in an abstract data type (highlighted by italics). Hence, the respective entry represents different possible concrete data types. For example, output generation services have the same input (*Result*) type. However, the content of (*Result*) type depends on the input entry of the *TransformToEPS*) module, which is the abstract *Outputdata* data type.

As in PROPHETS, in terms of the USE (input types) and GEN sets (output types) interface annotation, we create 53 modules to describe 28 SIBs in different behaviour aspects:

- The services perform the same process but with different input and output data (e.g. *Resample-GDP-data and Resample-yield-data*)
- The services have the same input and output data but perform different processes (e.g. *Load-GDP data and ConvertVector-to-RasterGDP*)
- The services take the same input but perform different processes and generate different output data (e.g. *Show-google-map and Produce-GeoTIFF-data*).

Another behaviour aspect of the service interface is described by defining KILL sets (types that must be removed from the set of types that were available prior to execution of the service). In principle, this method relies on realizing the semantics of ordering between services. KILL sets are useful to avoid the redundancy of services and to define a particular order of services in a certain workflow. For modules that have no input types, when it makes sense, we use some existing types as inputs. After that, a kILL type is defined when a module is used. Table (5.5) shows the KILL sets of types that are defined for an SLR service's interface (modules).

5.2 Taxonomies

Taxonomies[1] are used to provide abstract classifications for the terms used in the module descriptions, which are particularly useful for the formulation of constraints relating to groups of data types or services.

[1]In the context of this thesis, we use the term 'taxonomies' and 'ontologies' interchangeably.

Table 5.5: KILL sets of SLR services.

Module Name	Input types	Killed types
Selectpopulationdatafile	SLR	SLR
SelectGDPdatafile	SLR	SLR
SelectYielddatafile	SLR	SLR
Read-Kcal-values	Yield-CSVdata	Yield-CSVdata
Read-Kg-price	Yield-CSVdata	Yield-CSVdata
Load-SRTM-elevation-data	Coordinates	Coordinates
Load-elevation-data	Elevationdatafile	Elevationdatafile
Load-population-data	Populationdata-file	Populationdata-file
Load-GDP data	GDPdatafile	GDPdatafile
Load-landuse-data	Landusedatafile	Landusedatafile
Load-yield-data	Yielddatafile	Yielddatafile
Createmaster90	Rasterdatafile	Rasterdatafil
Creatmaster60	Rasterdatafile	Rasterdatafile
Creatmaster30	Rasterdatafile	Rasterdatafile
ResampleResoluation	Elevationscale, Datascale	Elevationscale, and Datascale
Resample-population-data	SLRlandloss, Population-data	SLRlandloss
Resample-GDP-data	SLRlandloss, GDPdata	SLRlandloss
Resample-yield-data	SLRlandloss, Yielddata	SLRlandloss
Masklandusedata	Croppedlanduse-data, SLRlanduse-sample	SLRlanduse-sample
ConvertVector-to-RasterGDP	GDPdatafile	GDPdatafile
ConvertVector-to-Rasterpopulation	Populationdatafile	Populationdatafile
ConvertKgtoUSD	Yielddata, Kg-price	Yielddata
ConvertKgtoKcal	Yielddata, Kcal-value	Yielddata
TransformToEPS	CRSdata, Outputdata	CRSdata, Outputdata
Compute-flooded-area	Elevationdata, SLR	Elevationdata
Compute landloss(ha)	Elevationdata, SLR	Elevationdata, SLR
Compute landloss-class	Elevationdata, SLR, Class-number	Elevationdata, SLR, Classnumber
Compute-population-risk	Populationdata, SLRpopulation-sample	Populationdata, SLRpopulation-sample
Compute-GDP-risk	GDPdata,SLRGDP-sample	GDPdata,SLRGDP-sample
Show-google-map	Result	Result
Generate-Png-file	Result	Result
Generate-Pdf-file	Result	Result
Produce-GeoTIFF-data	Result	Result
Produce-Text-data	Result	Result

In the proposed ontology model (service and type taxonomies), the general geospatial features and types are realized. As depicted in Figure (5.2), we define *Geospatial-services* as an abstract and a main class comprises *location determining*, *data manipulation*, *OGC services*, *domain-specific analysis*, and *output generation* services. Thus, most possible variations of services that can be reused in the entire domain of geospatial applications are considered in the service taxonomy.

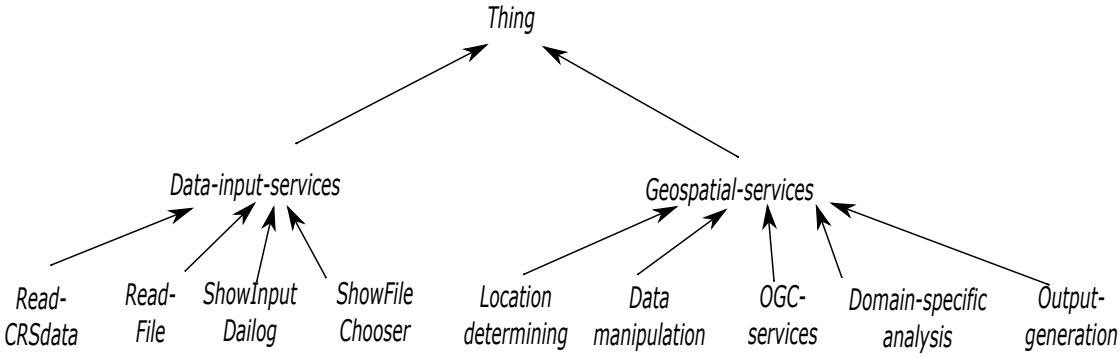


Figure 5.2: Part of the proposed geospatial service taxonomy.

General services that can be used in all application domains, such as services to read or load input data from files, are classified in a separated class named *Data-input-services*. As shown in Figure 5.3, modules created to describe services that read and load SLR data are defined as subclasses of this class. The classification of input services (taxonomy) is useful to add constraints for the entire SLR application domain.

The *Domain -specific analysis* class is defined and depicted in Figure 5.4 to address the reuse of complex geospatial services. Instead of describing the geospatial analysis operations (e.g. buffering and overlapping) of the service, we describe it as a domain-specific computation service. Consequently, the service is composed based on its input types rather than its entire operation. For instance, the **Compute-Rural-Urban-risk** SIB performs an intersection operation between two raster polygons. However, in our model, we describe the functionality of the service related to the SLR application domain rather than its operation. Therefore, services that are described to compute rural and urban damages are classified under the *SLR-rural-urban-damages class*.

In addition to SLR services, several domain-specific analysis services can be

added under *Domain-specific analysis*. For example, services that are created for carrying capacity analysis and water resources adequacy analysis applications (see Chapter 4) can be added under *City-module services* and *Water resources adequacy services*, respectively. Also, services that can be used to address other climate impact stimuli, such as temperature change and precipitation change, are classified under their classes.

Some applications may share the same computation services. Nevertheless, they will have different behavioural aspects or different interpretations for these services. For instance, the impacts of land loss, yield loss, or rural-urban damages may be caused by different events or stimuli. Therefore, we use specific terminologies to define computation services that are only used to analyse the impacts of the SLR stimulus and define them under the *SLR services* class. This class, which is a subclass of *domain-specific analysis services*, comprises services specifically for SLR impact analysis. Thus, it contains SLR-related specific application services, such as *SLR-landloss*, *SLR-urban-rural-damages*, and *SLR-yieldloss*. The leaves of the service taxonomy tree correspond to concrete modules of the domain model: *Compute-SLR-landloss-class*, *Compute-SLR-landloss(ha)*, *Compute-SLR-population-risk*, *Compute-SLR-yieldloss*, *Compute-SLR-productionloss*, and *Compute-SLR-caloric-energyloss*.

Furthermore, we define some relevant terminologies for the SLR analysis process to enable users with different interpretations or perspectives to use the related SLR impact analysis. For instance, users can select *SLR-yieldloss* or *SLR-foodloss* to use services related to the yield loss impacts.

The data manipulation class is defined to encompass the services that are required to manipulate geospatial data. Figure 5.5 illustrates six subclasses of manipulation services: *Rasterdata-Loading*, *Transformation*, *Masking*, *Clipping*, *Converting*, and *Resampling*. The leaves of the service taxonomy tree of manipulation services represent only services that we use for the SLR example in the context of this thesis. However, various manipulation services can be defined under the six main classes. For instance, considering additional formats of data, different data scales and coordinate-referencing systems need to define more service instances respectively under *Converting, Create-master-resolution, and Transformation*.

The two classes (*Location determining* and *Output generation*) of geospatial

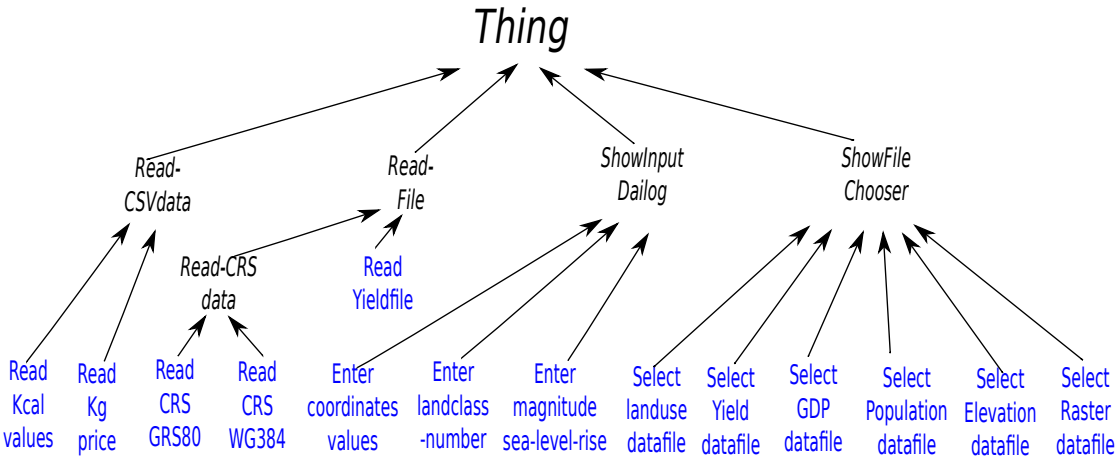


Figure 5.3: The taxonomy of SLR input services.

services are illustrated in Figure 5.6. Services that help identify the location of a particular area on the earth are classified under the *Location determining* class. The classifications of location services enable users to employ several options to identify the location—for instance, by defining coordinate values (north, east, south, and west), defining area information (country, city and address), or using services to load elevation data for a certain region. The OGC services class is defined to show that it is also possible to consider OGC web services as a separate class of geospatial services. A more detailed discussion is presented in Chapter 7.

Figure 5.7 illustrates the type taxonomy defined for the geospatial data. In this taxonomy, unlike some existing models, we also treat geospatial features (e.g. format, resolution, and geo-referencing systems) as types. Hence, six classes are defined to address the major characteristics of the geospatial data: *Location-data*, *Formats*, *Data-scales*, *Domain-specific data*, *Output-data*, and *Coordinate Reference Systems (CRS) data*. However, its subclasses are reduced to the parts that are relevant to the SLR example.

In addition to the *SLR data* class that is only used for SLR impact analysis , data that might be used in several domains—such as rural-urban data, population data, and agricultural data—are categorized and classified under the *Domain-specific data* class. Figure 5.8 shows the taxonomy of agricultural data and SLR data. The land use data and yield data classes comprise agricultural data that can be used in various domains. However, agricultural data that has been processed for SLR analysis is classified as SLR data (e.g. *SLRlanduse-sample, SLRYield-*

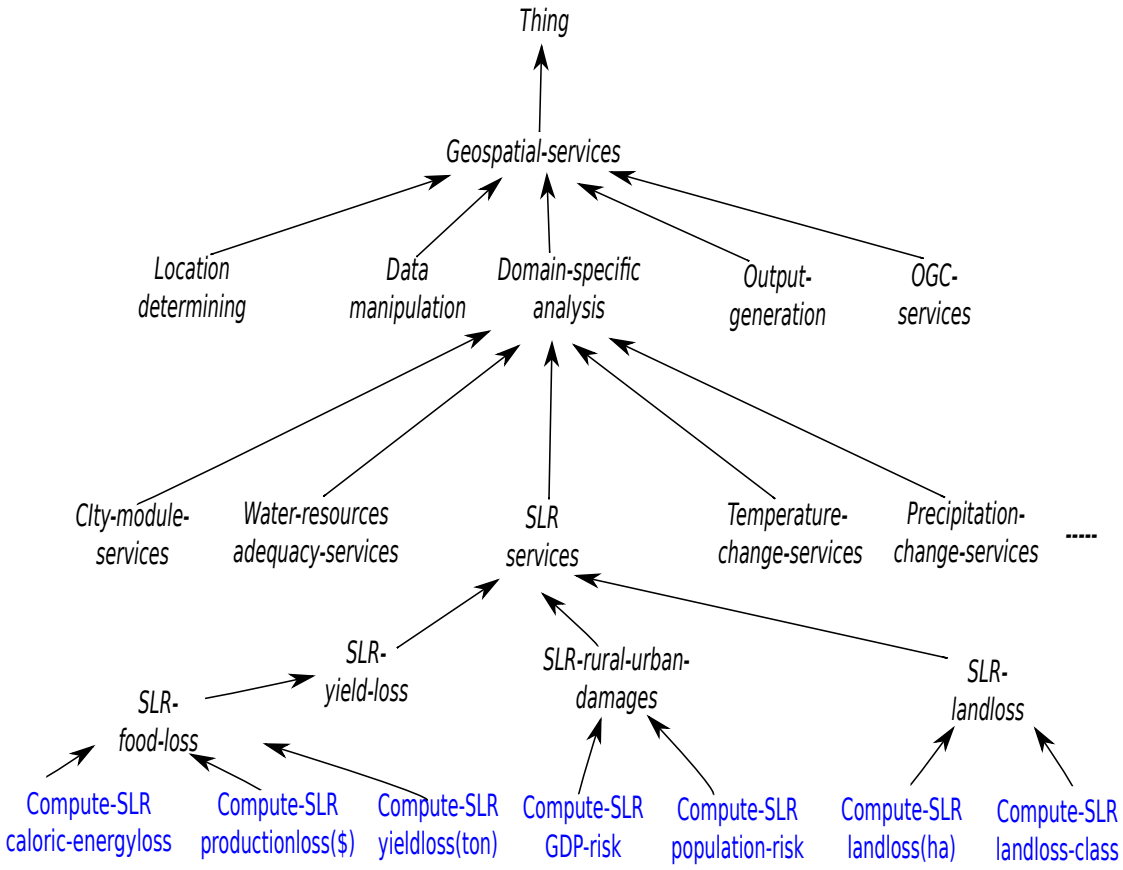


Figure 5.4: The taxonomy of SLR-specific services.

sample). The *Domain-specific data* class is useful for reducing the time required for data and service discovery. Based on its classes, users can indicate via constraints which data are required for a specific analysis of SLR impacts, or avoid unnecessary data. To enable users who are using different formats of geospatial data and handling their constraints regarding the available data formats, we define the **Formats** class, which comprises various data formats (only raster and vector classes are depicted in Figure 5.7). It is also possible to define several types of raster and vector formats under their respective classes.

On the other hand, to address the respective data resolution, the *Data-scales* class enables users to specify a certain scale of data resolution. The *ResampleResolution* module takes *Data-scales* as an abstract input data type that comprises various types of data resolutions (e.g. GRS80 and WGS84). We define the *CRS-data* class as the input type for the *TransformToEPS* module, which includes

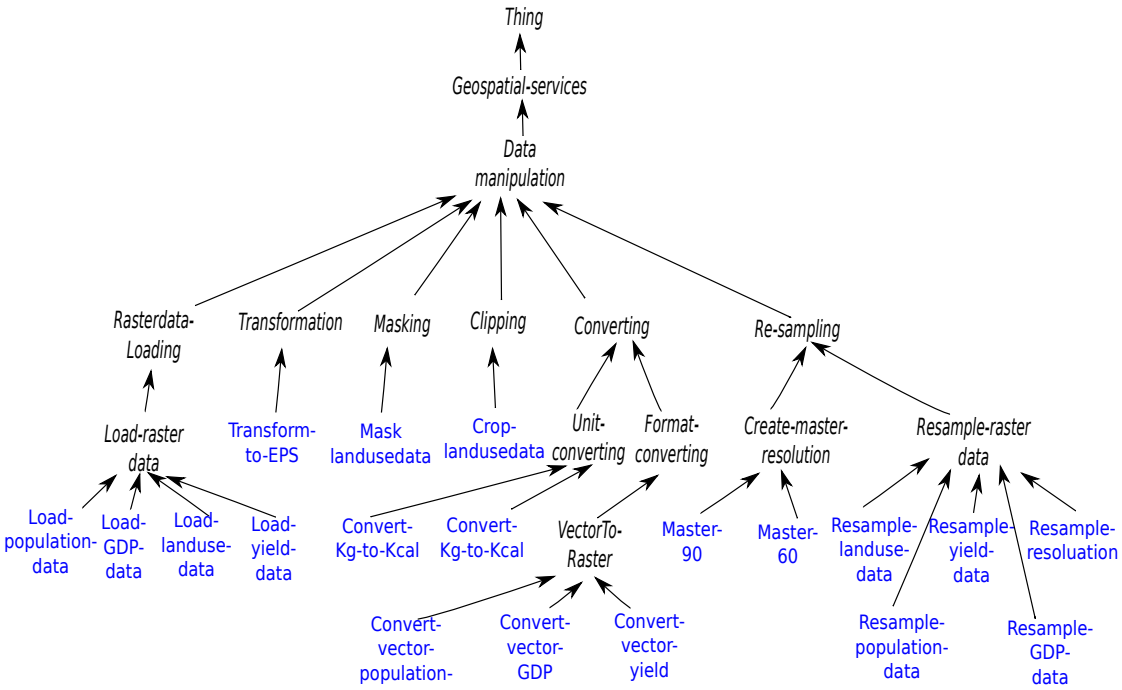


Figure 5.5: Part of the taxonomy of data manipulation services.

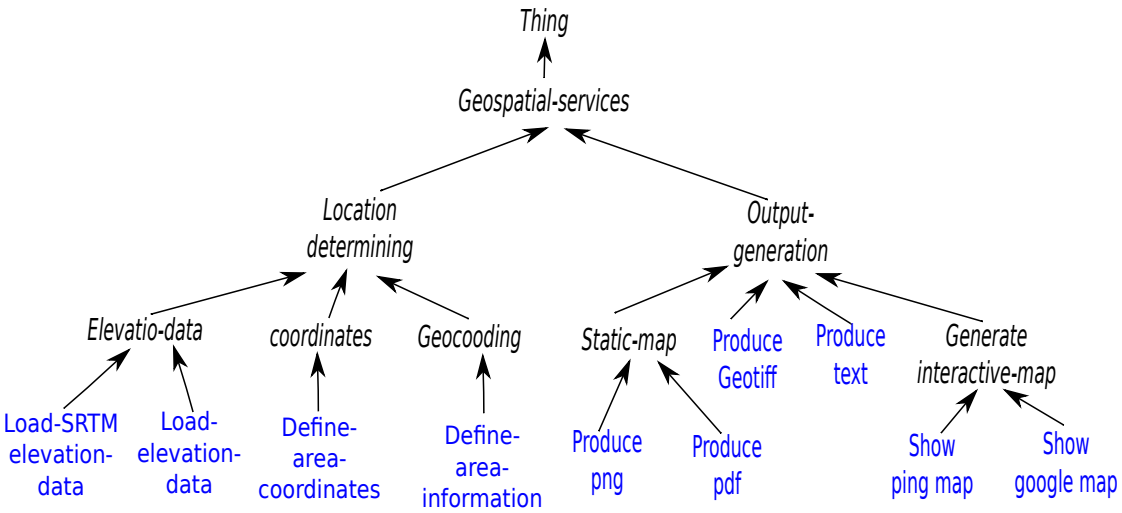


Figure 5.6: Part of the location-determining and output generation services taxonomies.

geo-referencing systems (e.g. GRS80 and WGS84). Thus, the consistency of data projection is assured by pointing to a specific type of available geo-referencing system.

The *location data* class, which comprises geo-coding, coordinates data, and

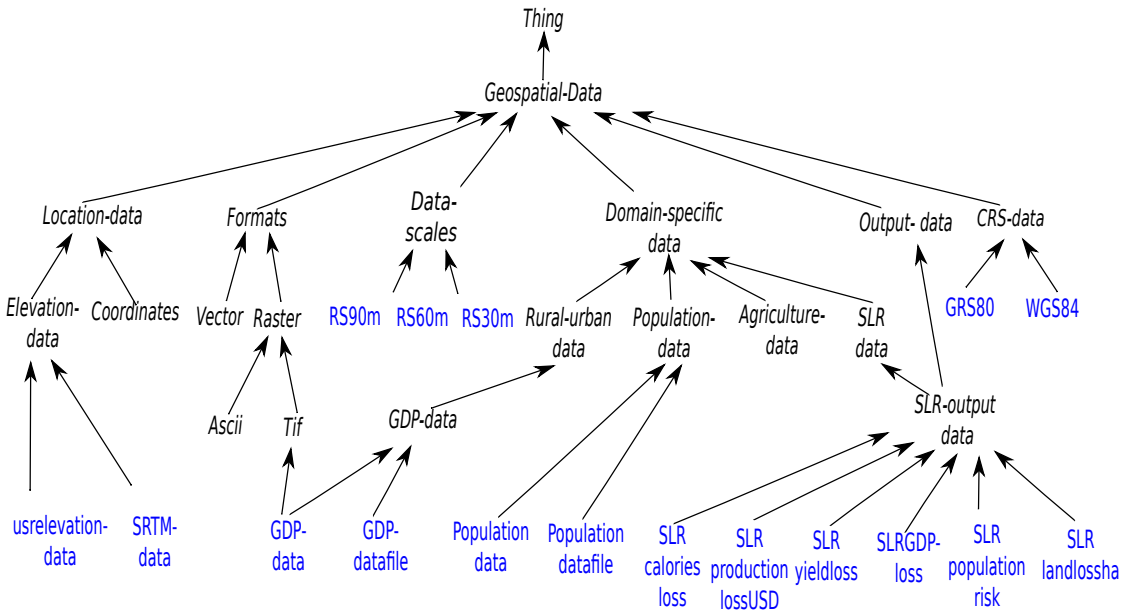


Figure 5.7: Part of the proposed geospatial type taxonomy.

elevation data, provides users with several options to determine the location of the region. It also allows using different types of elevation data to identify the location, such as by using SRTM elevation data or the user's own elevation data.

The *Output-data* class is used to group outputs of computation services, which are again inputs for the *Transformation* services (see Table 5.4). In order to make *output generation* services reusable in several domains, a specific class called *SLR-output-data* is defined to comprise the output of SLR computation services.

Owing to the limited scope of this research paper and the facility presentation , we only depict the essential excerpts of taxonomies that are relevant for the example discussed in Chapter 6. For more details about complete service taxonomies, see Figure 5.9.

5.3 Domain Constraints

During the domain-modelling phase, the general constraints that can be applied to the entire application domain (in this case, SLR impact analysis applications) are defined. For instance, we tackle the redundancy of services that have no input types by defining dummy USE sets and KILL sets in the service interface description

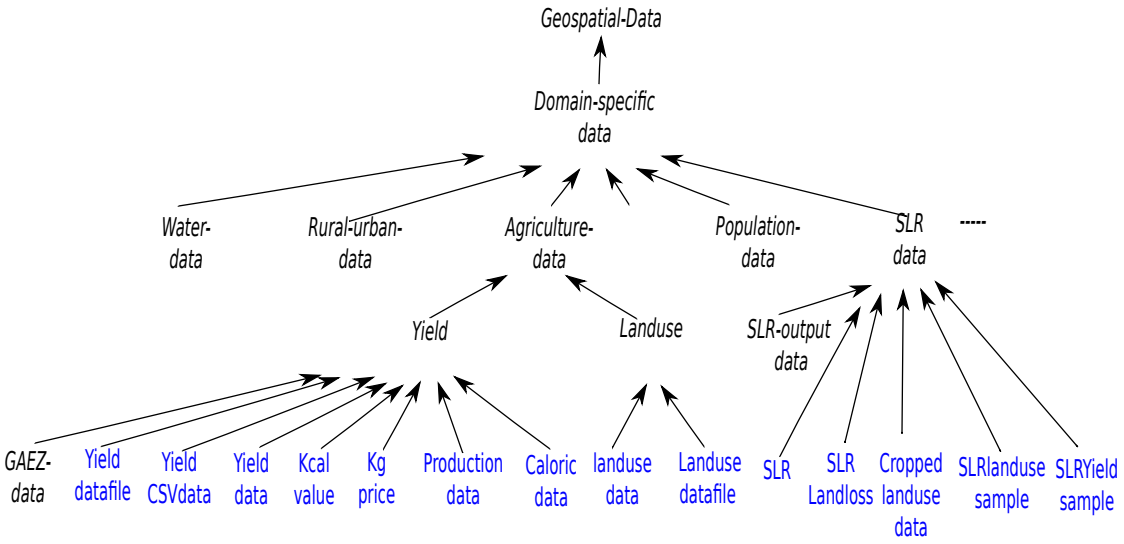


Figure 5.8: Part of domain-specific type taxonomy.

phase. However, this method is not suitable for some services. In particular, this includes services where the output types (GEN sets) are used by several services in the workflow instance. Thus, the following constraints should be added to avoid redundancy:

- Do not use module *Enter magnitude of sea-level rise* more than once, which represented in SLTL by:

  $$G\,(\,\langle\text{Enter magnitude of sea-level rise}\rangle\,\text{true}\;\Rightarrow\; X(G(\neg\,\langle\text{Enter magnitude of sea-level rise}\rangle\,\text{true}\,))).$$

  Where G expresses that the *Enter magnitude of sea-level rise* module must be not hold generally more than once over the entire domain.

- Do not use module *location-determining* more than once, which represented in SLTL by:

  $$G\,(\,\langle\text{location-determining}\rangle\,\text{true}\;\Rightarrow\; X(G(\neg\,\langle\text{location-determining}\rangle\,\text{true}\,))).$$

  Where G expresses that the *location-determining* module must be not hold generally more than once over the entire domain.

We suppose that for each workflow instance, the services of output generation have to be the final service in the workflow; therefore, we add the following constraint:

- Use module *output-generation* as the last module in the solution, which represented in SLTL by:
  $F\,(\,\langle\text{output-generation}\rangle\,\text{true}) \wedge G\,(\neg\,\langle\text{output-generation}\rangle\,\text{true}) \vee (\neg\,\langle\rangle\,\langle\rangle\,\text{true}))$.
  Where F expresses that *output-generation* must hold as the last module in the workflow.

Other constraints can be added in the domain-modelling phase to exclude useless or unwanted services but this should be true and valid for all possible workflows of the SLR application domain. In fact, using the constraints of *module avoidance and type avoidance* is not possible in some cases in order to exclude unwanted services. This holds true when a dependency relation exists with other services. For instance, the service *Read-CRSdata* is used only before the *TransformToEPS* module. Therefore, we add the following constraint:

- If module *Read-CRSdata* is used, module *TransformToEPS* has to be used next. The SLTL representation of this constraint is:
  $$G\,(\,\langle\text{Read-CRSdata}\rangle\,\text{true} \Rightarrow X(\,\langle\text{TransformToEPS}\rangle\,\text{true}\,)).$$

Depending on the intended solution, in addition to user intent constraints, more constraints may be added to exclude useless or unwanted services in the workflow design phase. The goal of the next chapter is to describe the designed domain model that is utilized to perform the automatic workflow design.

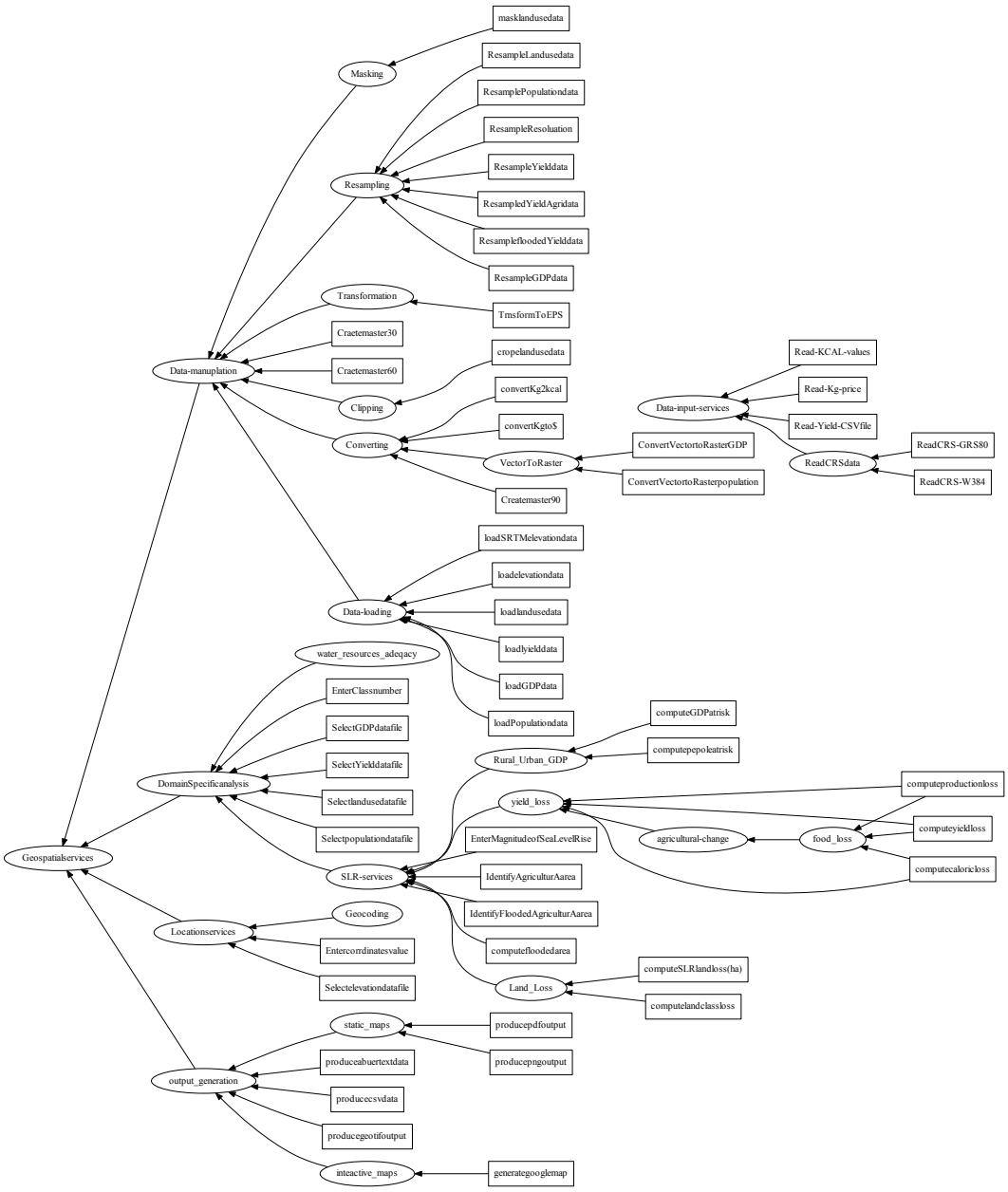


Figure 5.9: Geospatial service taxonomy.

Chapter 6

Synthesis of SLR Analysis Workflows

Based on the domain model of SLR applications (see Chapter 5), this chapter elaborates on how we achieved automatic geospatial service composition by synthesizing SLR workflows. It also describes how different types of semantics-based geospatial service composition are handled.

By referring to the SLR scenario presented in the introduction of this thesis, the synthesis process is applied to the SLR example from Figure 1.1. As shown in Figure 2.5, the approach is based on the PROPHETS plugin of the jABC workflow-modelling framework and comprises two main parts: domain model (on the left side) and automatic workflow design (on the right side). Mainly, this approach is based on domain models that specify the overall domain services and data in a well-structured knowledge representation form, such as ontology. By addressing the example of analysing the impacts of SLR, we will show how users can benefit from the provisioning domain model to define constraints, thus obtaining the intended workflow design.

The right side of Figure 2.5 depicts how the overall synthesis process of PROPHETS performs the following steps: (1) interpreting the branch between *enter magnitude of sea-level rise* SIB at the beginning and *show-SLR-impacts* SIB at the end as a loose specification, (2) enabling users to add constraints in an iterative manner, (3) generating the possible solutions, and (4) inserting the selected solution in the loose branch automatically.

Table 6.1: Synthesis statistics (solutions) for the SLR applications.

Search depth	Number of Solutions	
	Unconstrained synthesis	Constrained synthesis
7	24	24
8	132	24
9	492	96
10	1620	96
11	4716	144
12	12168	144
13	29172	288
14	64932	300
15	182,189	312

By applying the synthesis process on the loose specification branch between highly abstract services, such as `enter magnitude of sea-level rise and show SLR impacts` SIBs, we can automatically explore a large number of possible variations in the workflows. Considering only the input or output specification defined by the loosely specified branch, the synthesis encounters more than 160,000 solutions, starting from a depth search of seven up to 15.

The explored solutions are technically possible and correct combinations of services. Furthermore, it is desirable to explore the possibilities that the domain model provides. However, the large number of solutions is unfeasible for users. Therefore, it is useful to restrict the synthesis process to return fewer but more adequate solutions.

The domain constraints (defined in Section 5.3) have a significant impact on the synthesis solutions. Table 6.1 presents the number of solutions of SLR workflows. The search depth ranging from seven to 15 shows the difference between unconstrained synthesis (possible implementations of the specification) and con-

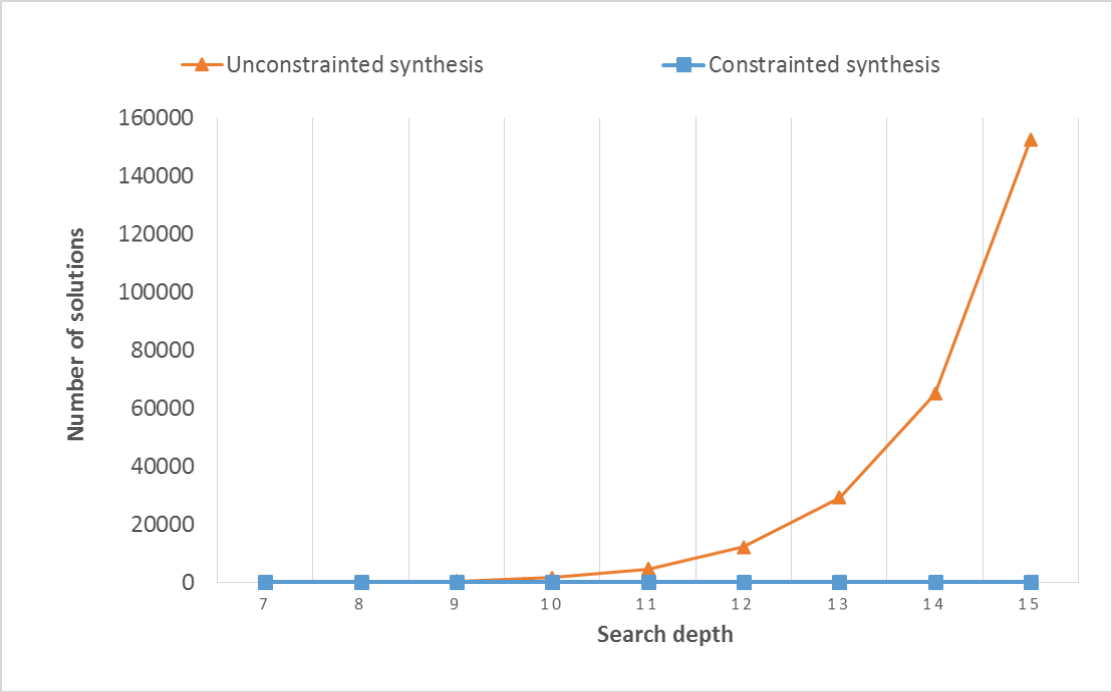


Figure 6.1: Synthesis statistics of obtained SLR solutions.

strained synthesis (after the applied domain constraints). At a depth of seven, the synthesis explores the first 24 candidate solutions. This reflects the minimum number of possible services (SIBs) that can exist in the workflow. Hence, at least seven services should be available to construct the SLR workflow. By increasing the depth level, more solutions can be discovered. Depending on the available types of datasets and their features, additional services should be used for conversion, re-sampling, and transformation. At such a depth level, when services are available to handle more variations in workflows, the number of solutions will increase.

In addition to particular synthesis configurations, domain constraints have a great impact on the number of solutions—that is, they effectively reduce the number of returned solutions. For instance, as visualized in Figure 6.1, instead of 1,620 solutions, 96 solutions are explored at a search depth of 10 and the 186,000 solutions at depth 15 are reduced to 312. Furthermore, only 672 solutions are explored in total at search depth 21 and the synthesis was automatically aborted because no more solutions can be explored after search depth 21.

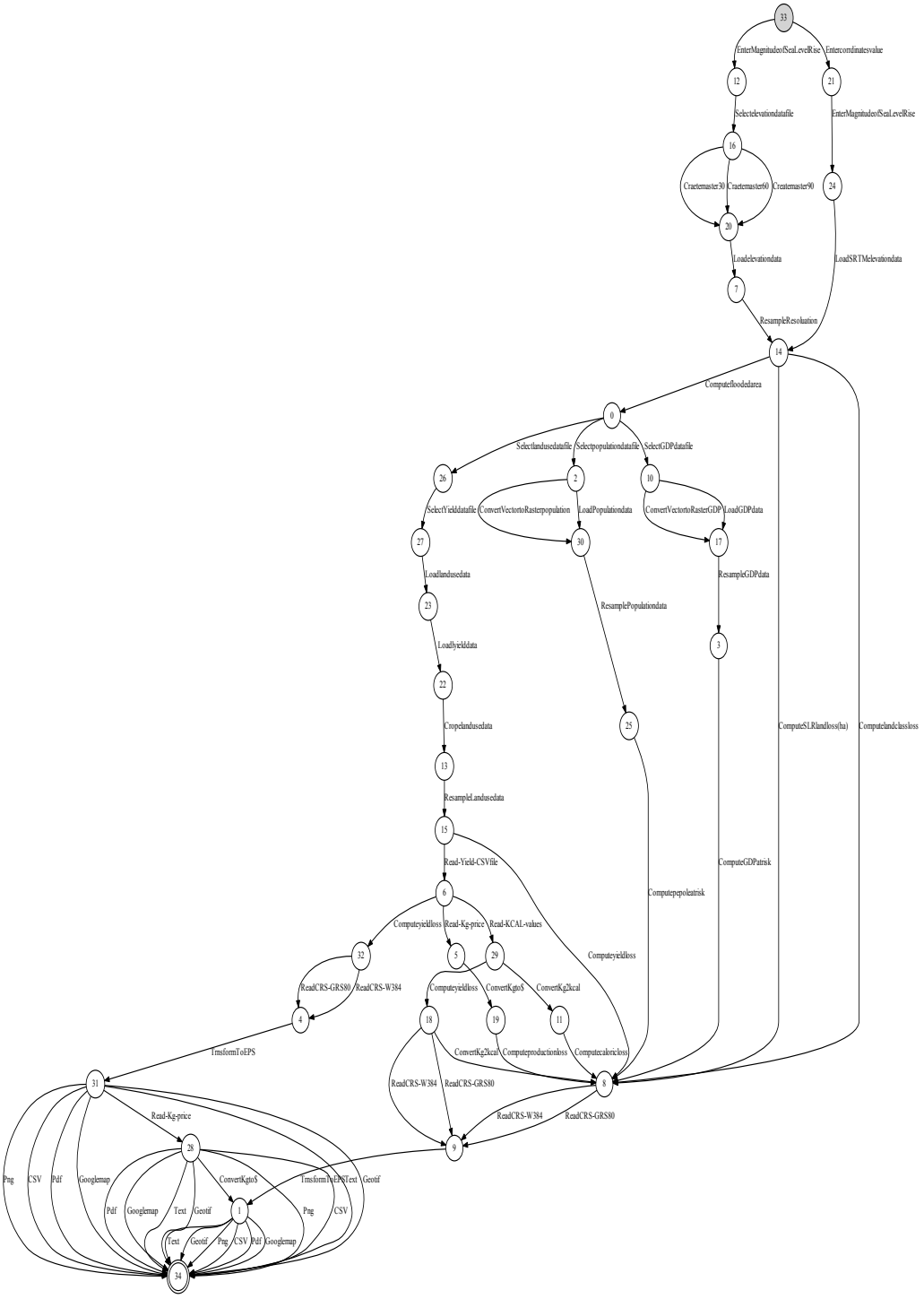


Figure 6.2: The solution graph of potential solutions of SLR workflows based on domain constraints.

Besides the number of solutions given in Table 6.1, we generate solution graphs to illustrate 672 of the potential solutions of SLR workflows. Figure 6.2 depicts a solution graph to represent how consolidated result representations of all possible solutions can be aggregated in a single graph structure. Indeed, these solutions are technically possible and correct combinations of services that solve the request. Nevertheless, additional constraints are required to target a specific solution matching user objectives and preferences. In the following, we discuss some more about the change of potential solutions and we elaborate on the impact of several refinements of constraints on the synthesis solutions.

Owing to the limited number of solutions that can be displayed in the PROPHETS window (only 100 solutions can be displayed), the solution graph is useful to depict all obtained solutions of each SLR application.

6.1 Handling Semantic Reuse of Geospatial Services: Refinement(a)

Typically there are many different possibilities for workflows implementing the specification based on the domain model of SLR applications and the same services can be reused in several workflows. However, depending on the analysis objectives of the SLR applications (such as SLR-landloss, SLR-rural/urban damages, and SLR-yieldloss), the services have different behaviour in the workflows. For example, in the SLR impact analysis application, the 'compute-flooded area' and 'compute-landloss' services are semantically similar but syntactically different. On the other hand, the 're-sampling' services are syntactically similar but semantically different, which may refer to 'resample landuse data' or to 'resample population data'. Thus, to ensure a semantic geospatial service reuse in SLR workflows, we benefit from the decoupling feature and using symbolic names in PROPHETS to define the precise vocabulary for each module name and type.

In order to show how these services are reused semantically, we ran three experiments to synthesize SLR workflows. In addition to the domain constraints, by pointing out the synthesis algorithm to one of the main classes of SLR analysis services, we add the constraints of refinement(a), which are concerned with the

intended goal of SLR analysis. For instance, for each SLR application, we can add the constraints enforcing the use of at least one module from the *SLR-services* class from the service taxonomy in the solution:

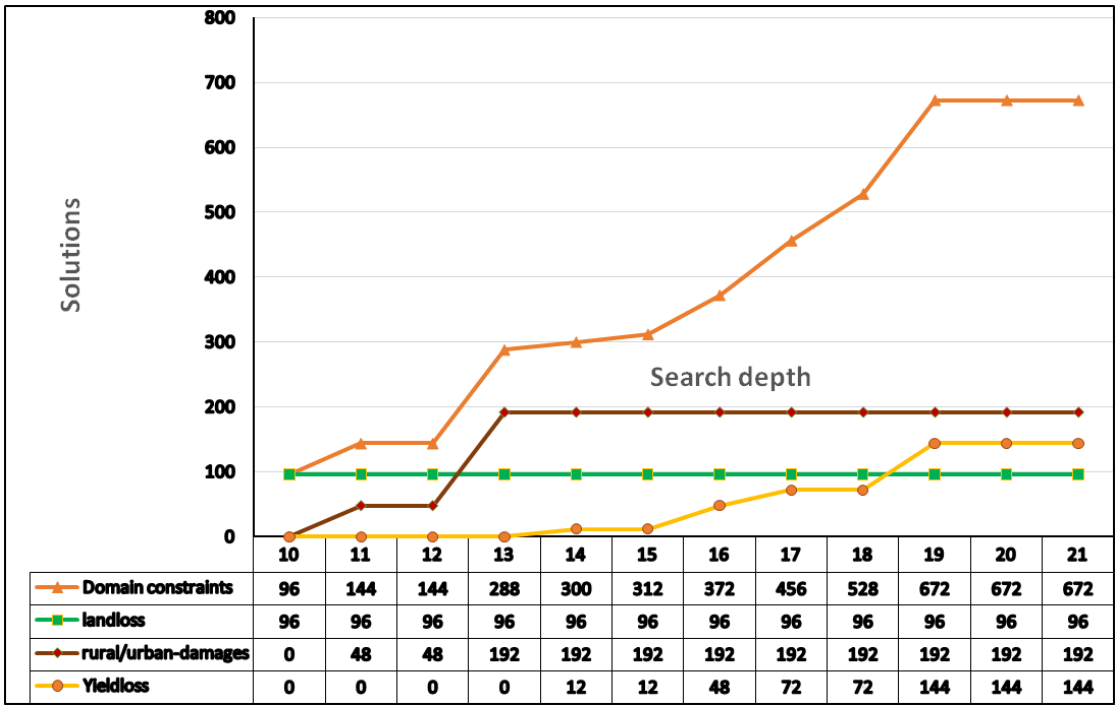
- *SLR land loss impact analysis*: Enforce the use of module *SLR-land-loss*, which represented in SLTL by:
$$F\,(\,\langle SLR-land-loss\rangle\,\text{true}).$$
Where F expresses that the *SLR-and-loss* module must be hold eventually in the solutions.

- *SLR rural or urban damages impact analysis*: Enforce the use of module *SLR-rural-urban-damages*, which represented in SLTL by:
$$F\,(\,\langle SLR-rural-urban-damages\rangle\,\text{true}).$$
Where F expresses that the *SLR-rural-urban-damages* module must be hold eventually in the solutions.

- *SLR yield loss impact analysis*: Enforce the use of module *SLR-yieldloss*, which represented in SLTL by:
$$F\,(\,\langle SLR-yieldloss\rangle\,\text{true}).$$
Where F expresses that the *SLR-yieldloss* module must be hold eventually in the solutions.

Figure 6.3 illustrates the impact of the constraints of refinement(a) on the synthesis solutions. Compared to solutions that are explored based on domain constraints, fewer numbers of solutions are returned by targeting specific SLR applications. At a search depth of 10, a total of 96 solutions are returned for SLR-landloss workflows and no more results can be explored after that. With the analysis of the rural or urban damage impact, the synthesis starts exploring 48 solutions at depth 11. At depth 13, it returns 192 solutions, which is the maximum number of potential solutions for rural and urban damage workflows. In addition to the main types of yield loss impact analysis (e.g. yield loss, production value affected in USD, and caloric energy loss), several types of data are used for yield loss impact analysis.

Furthermore, additional services, such as cropping and masking, are needed to process land use data. Thus, at least 14 services (SIBs) are required to synthesize



	10	11	12	13	14	15	16	17	18	19	20	21
Domain constraints	96	144	144	288	300	312	372	456	528	672	672	672
landloss	96	96	96	96	96	96	96	96	96	96	96	96
rural/urban-damages	0	48	48	192	192	192	192	192	192	192	192	192
Yieldloss	0	0	0	0	12	12	48	72	72	144	144	144

Figure 6.3: Impact of constraints of refinement(a) on the synthesis solutions.

its workflows. Hence, the synthesis explores the first 12 solutions at depth 14 and 144 solutions are returned in total from depths 19 to 21.

The solution graphs in Figure 6.4 depict the impact of applying the constraints of refinement(a) on the synthesis and how its graphic representation becomes clear. Instead of one solution graph representing 672 solutions, only three graphs are generated individually to represent: (a) 96 solutions of SLR-landloss workflows, (b) 192 solutions of SLR-rural-urban damages workflows, and (c) 144 solutions of SLR-yieldloss workflows. Note that more solutions can be expected of SLR-yieldloss workflows. However, in our example, we did not consider the variations in the data formats of yield and land use data.

In order to assist the semantics of service reuse in the synthesized SLR workflows, we need to identify services that have a high frequency of reuse. Hence, we use statistics to calculate the service reuse frequency. Corresponding to the created services classification in Chapter 4, Table 6.2 presents the number of service reuses in the workflows of SLR applications. Again, not surprisingly and due to their general purpose, data manipulation services are reused more frequently than

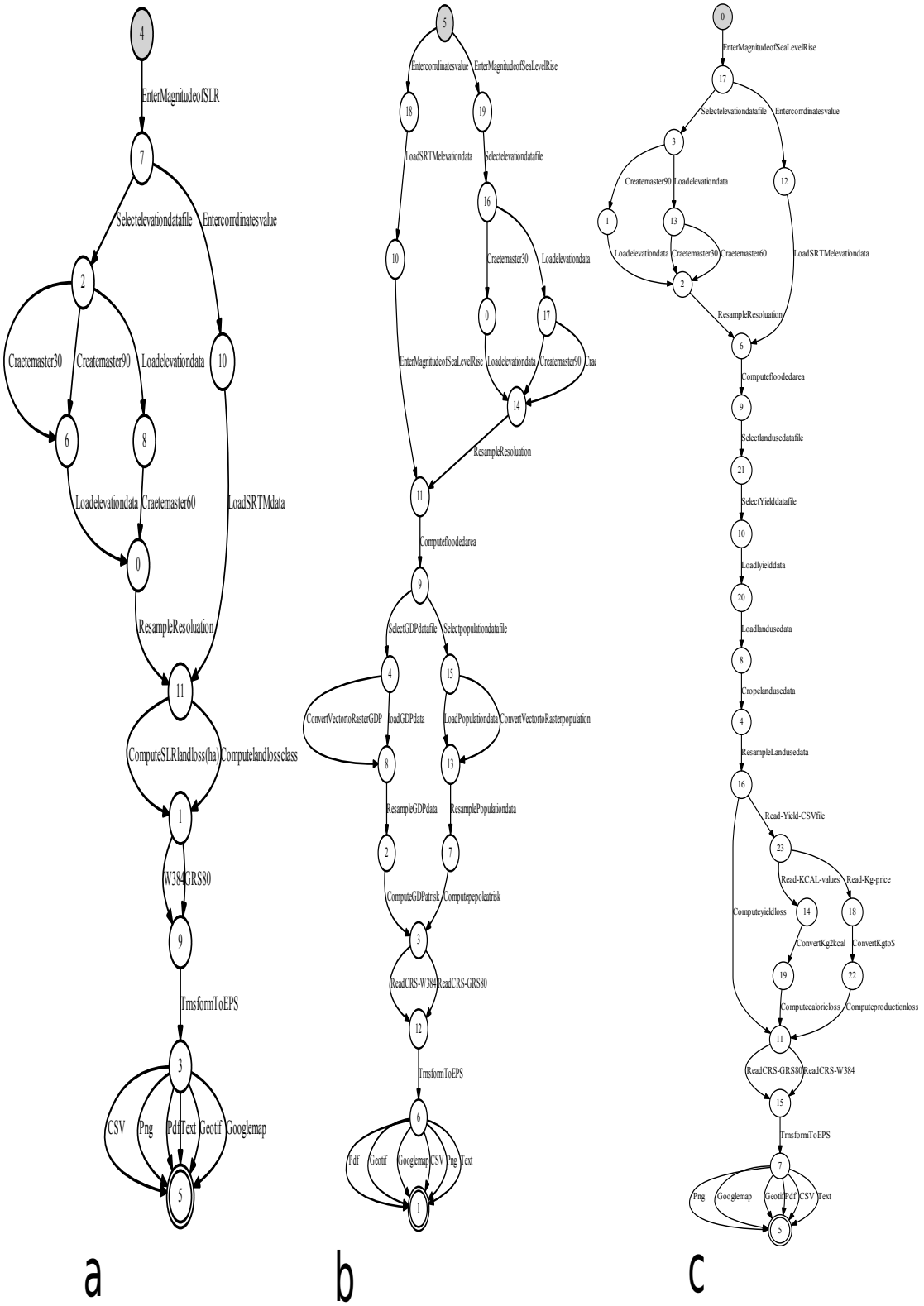


Figure 6.4: Solution graphs of refinement(a). (a) SLR-landloss. (b) SLR-rural-urban damages. (c) SlR-yieldloss.

Table 6.2: Frequency of service reuse

SLR Applications	Number of Service Reuses		
	Manipulation Services	Computation Services	Output Services
Land-loss	336	96	96
Rural-urban-damages	504	200	100
Yield-loss	832	200	100
Total number of reuses	1672	496	296

other services.

The semantics of the service reuse of the manual work design are defined by users; however, in the synthesized workflows, the semantics of service reuse are achieved automatically. This is achieved by providing the service interface with semantic annotations that ensure a correctness of service matching and the user's intents. Owing to the current configuration of PROPHETS (only 100 solutions are displayed), statistics are calculated for only the first 100 solutions of each SLR application (except land loss, which has only 96 solutions in total). Compared to the statistics of Table 4.4 in Chapter Chapter 4, the number of service reuses is significantly increased. Within 296 workflows of SLR applications, seven workflows of manipulation services are reused 1,672 times, six of computation services are reused 406 times, and five of output generation services are reused 296 times.

In essence, although most created geospatial services have the same signature and functionality, they are composed to have different behaviours based on their interface description. For instance, Figure 6.5 illustrates three different reuses of the re-sampling service: resample GDP-data, resample population data, and resample landuse data. Depending on its semantic annotations, the re-sampling service is composed semantically in three workflows of SLR applications with different behaviours.

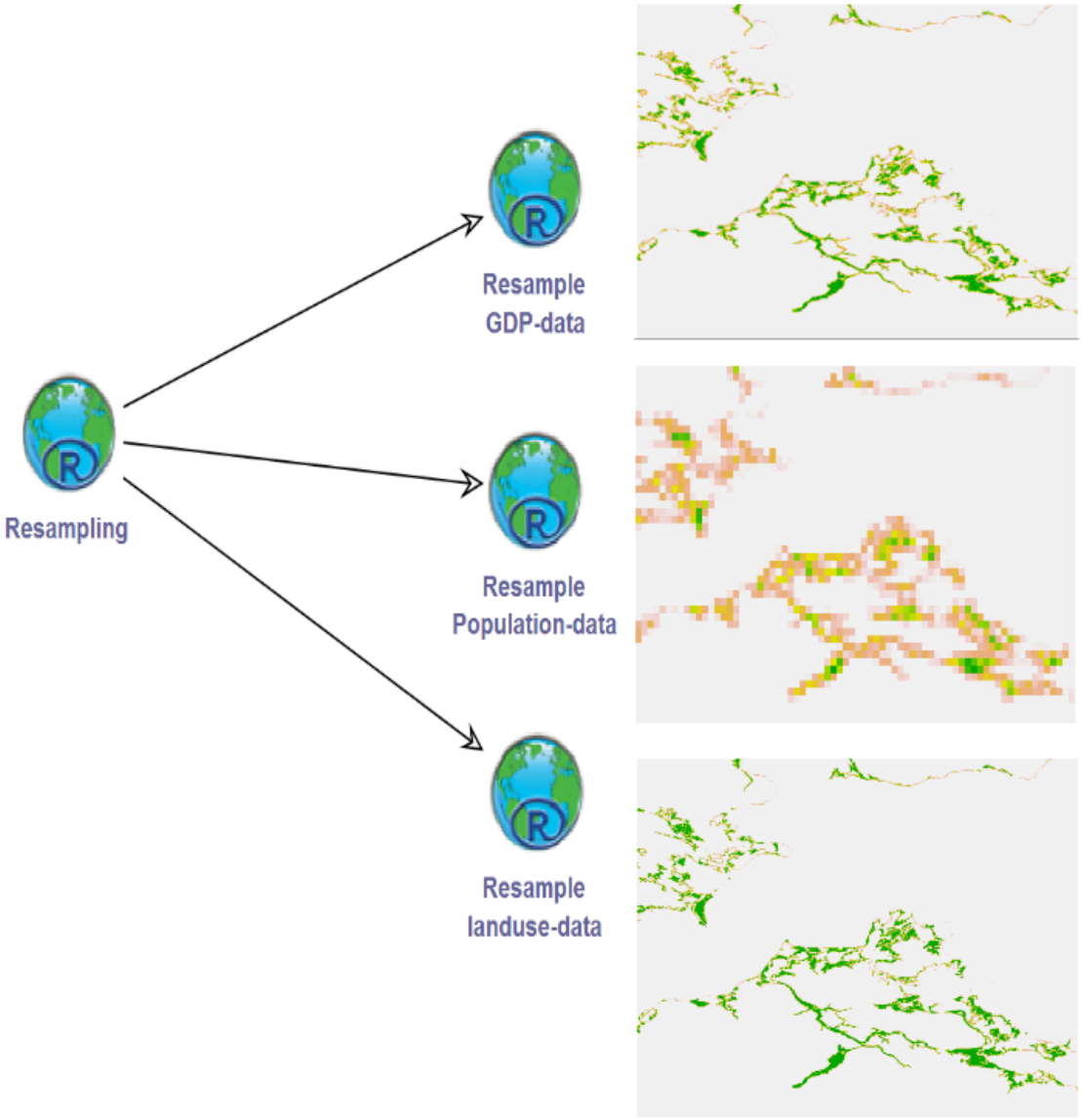


Figure 6.5: Semantic reuse of re-sampling service.

6.2 Handling the Semantics of User Intents: Refinement(b)

Even though all obtained solutions in refinement(a) are relatively accepted and semantically correct to solve the request, most of them are not adequate solutions—that is, they do not represent what the user actually wants to see. Therefore,

in addition to the domain constraints and instead of adding the constraint of refinement(a) which is concerned with the more general intended goal of SLR analysis, we add the constraints of refinement(b), which express the user's intention of excluding uninteresting solutions.

In this refinement, we expect that users have more specific objectives related to the main SLR impact analysis. Accordingly, the following constraints are added to explicitly include and exclude particular (groups of) modules from the solutions of each SLR application:

- **SLR-Land-loss**:
  - Enforce the use of module *Compute landloss(ha)*, which represented in SLTL by:
    $$F\,(\,\langle Computelandloss(ha)\rangle\,\text{true}).$$
    Where F expresses that the *Compute landloss(ha)* module must be hold eventually in the solutions.
  - Avoid the type *SRTM-elevation-data*, which represented in SLTL by:
    $$G\,(\neg\,\langle\,SRTM-elevation-data\,\rangle\,\text{true}).$$
    Where G expresses that the *SRTM-elevation-data* type must be not hold globally in the solutions.
- **SLR-rural-urban-damages**:
  - Enforce the use of module *compute-GDP-risk*, which represented in SLTL by:
    $$F\,(\,\langle compute-GDP-risk\rangle\,\text{true}).$$
    Where F expresses that the *compute-GDP-risk* module must be hold eventually in the solutions.
  - Avoid the type *Coordinates*, which represented in SLTL by:
    $$G\,(\neg\,\langle\,Coordinates\,\rangle\,\text{true}).$$
    Where G expresses that the *Coordinates* type must be not hold globally in the solutions.
- **SLR-yield-loss**:

– Enforce the use of module *Compute-caloric-energy-loss*, which represented in SLTL by:

$$F\,(\,\langle Compute - caloric - energy - loss\rangle\,\text{true}).$$

Where F expresses that the *Compute-caloric-energy-loss* module must be hold eventually in the solutions.

– Avoid the type *SRTM-elevation-data*, which represented in SLTL by:

$$G\,(\neg\,\langle\,SRTM - elevation - data\,\rangle\,\text{true}).$$

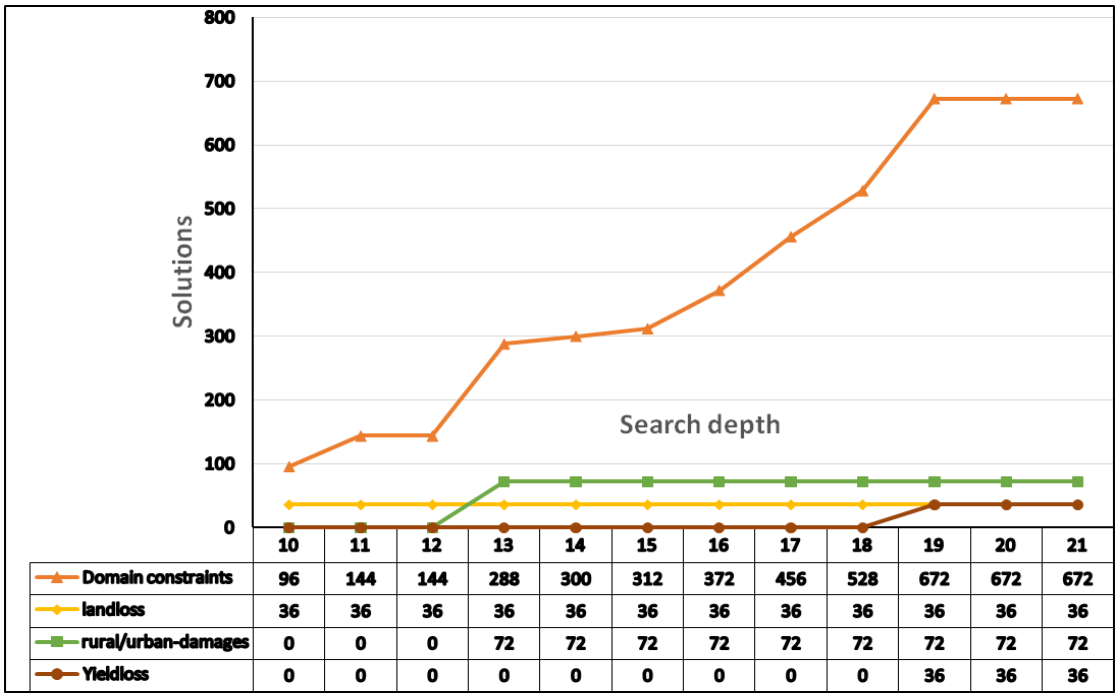
Where G expresses that the *SRTM-elevation-data* type must be not hold globally in the solutions.

r

As the constraints of refinement(b) focused on more specific SLR impact analysis to tailor the specification closer to the user's specific intentions, the number of obtained solutions is reduced further. In fact, already manageable sets of possible implementations are obtained and illustrated in Figure 6.6. A constant number of 36 solutions for SLR-land-loss workflows are returned among all search depths. Only 72 solutions are returned for SLR-rural/urban-damages workflows. The number of solutions of yield loss workflows is significantly reduced to 36.

Alongside this, the solution graphs in Figure 6.7 illustrate the impact of the constraints of refinement(b) on the synthesis solutions. For each SLR application, we generate the solution graph to represent the returned solutions. Figures 6.7a and 6.7c represent the 36 solutions of SLR-landloss and 36 solutions of SLR-yield-loss, respectively. On the other hand, Figure 6.7b represents the 72 solutions of SLR-rural/urban-damages. The solution graphs of refinement(b) indeed make the solutions more limited; users can easily identify the point of variation in each solution graph. For instance, Figure 6.7a shows that at node 9 there are three different branches reflecting the types of data resolutions (raster30, raster60, and raster90); at node 4, two branches state the possible geo-referencing systems; and six branches at node 3 depict alternatives for output generation (GeoTIFF, Google Map, PDF, PNG, Text, and CSV).

Obviously, the functionalities of more complex geo-processing operations (such as buffering or overlay, but also topological operators [54]) pose even harder semantic challenges. If the functionality descriptions become too complex, they are



	10	11	12	13	14	15	16	17	18	19	20	21
Domain constraints	96	144	144	288	300	312	372	456	528	672	672	672
landloss	36	36	36	36	36	36	36	36	36	36	36	36
rural/urban-damages	0	0	0	72	72	72	72	72	72	72	72	72
Yieldloss	0	0	0	0	0	0	0	0	0	36	36	36

Figure 6.6: Impact of constraints of refinement(b) on the synthesis solutions.

unlikely to be produced by service providers and understood by clients. But if they are too simple, recall and precision in discovery and evaluation are reduced. Furthermore, the descriptions need to support the reasoning necessary to match service offers to requests. If this reasoning becomes too expensive, it threatens the efficiency of service discovery.

Meanwhile, handling the semantic user's intents already tackled the semantic reuse of complex geospatial processing (such as buffering and topological relations between spatial objects). Tacitly, by adding the constraints of the user's analysis objectives of refinements(b), geo-processing operations designed to compute SLR impacts are semantically reused and composed in the solutions. In principle, users are unaware of the content and nature of the spatial operation of computation services, such as *Compute landloss(ha), compute-GDP-risk, and Compute-caloric-energy-loss*. More precisely, instead of defining semantics to support the reasoning of spatial processes, users employ constraints to target a domain-specific service relating to their objectives. For instance, the same geospatial processes are used for services that compute rural or urban damage and those that compute population

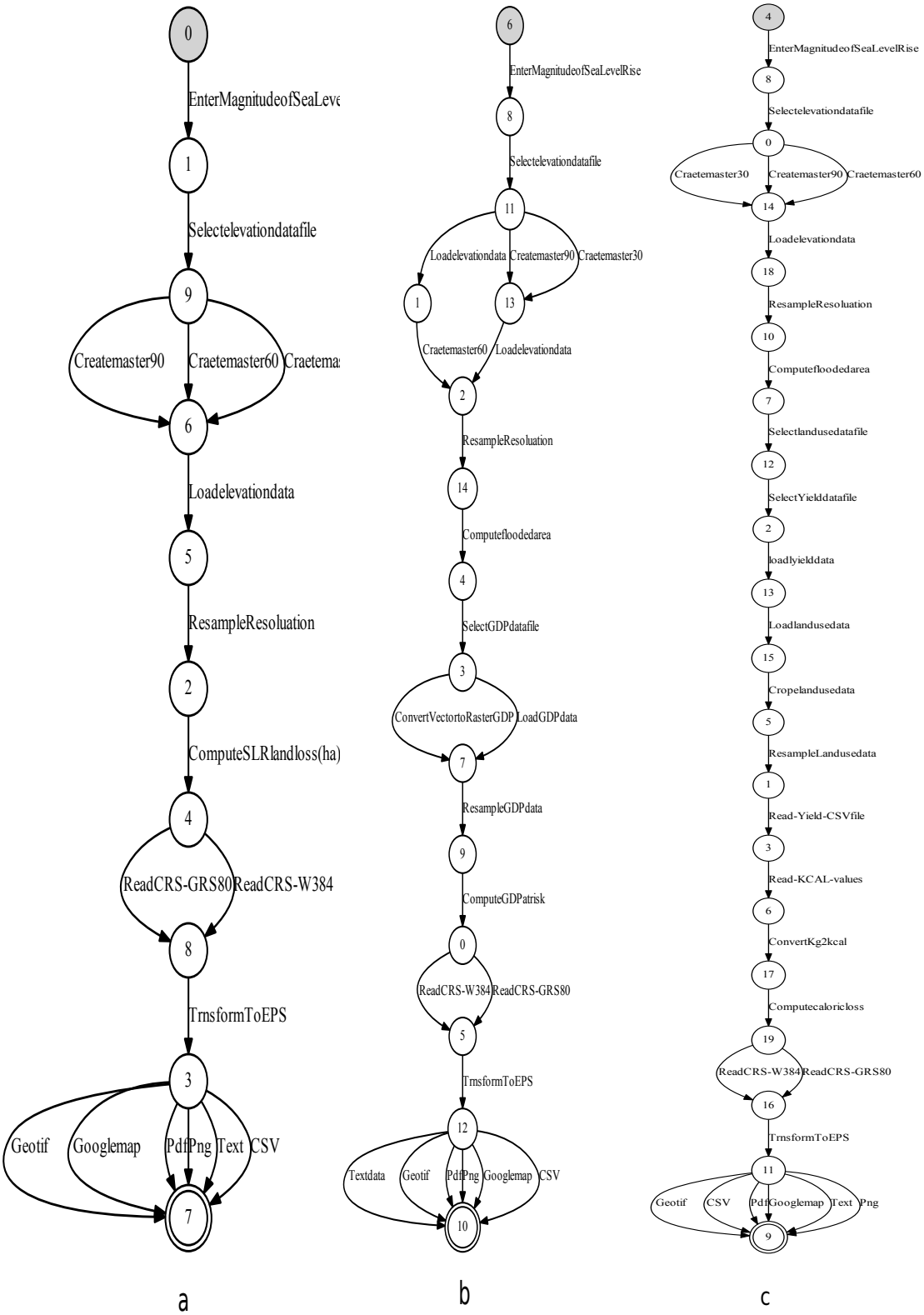


Figure 6.7: Solutions of SLR application b.

at risk. However, without information needing from users about geo-processing operations, these services are reused and composed semantically in different behaviours of workflows.

6.3 Handling the Semantics of Geospatial Data: Refinement(c)

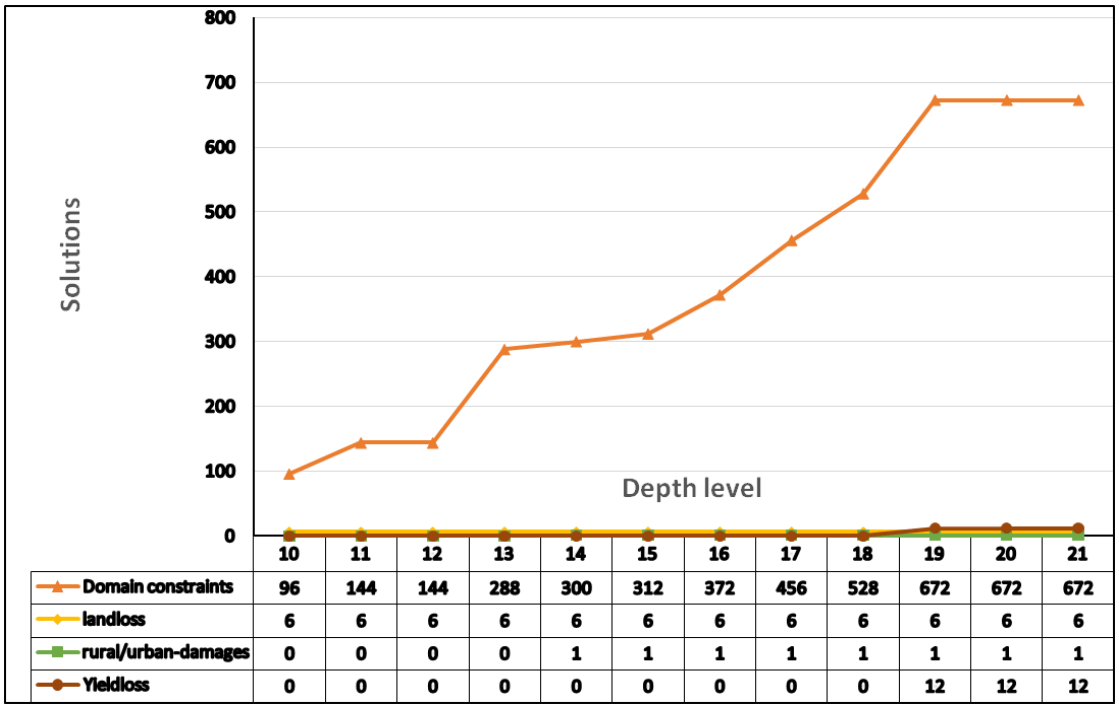
The geospatial application domain belongs to a data-centric field. Furthermore, the characteristics of geospatial data, such as formats, spatial data resolution, and geo-referencing systems, are very demanding with regard to handling consistent data during the workflow synthesis. Therefore, in Chapter 5, we showed, in terms of semantic annotation relating to the existing data types of SLR applications, how the semantics of geospatial data compatibility are handled.

Usually workflows are constructed based on matching the input and output data types. However, providing a data type with the semantic description is not always sufficient to identify the match between the input and output service parameters [191]. Nevertheless, this depends on the domain model that should have the adequate annotations relating to service input and output. As mentioned earlier, in our domain model, we include the features or properties (e.g. formats, scales, and geo-referencing systems) of the geospatial data type into the type taxonomy. Hence, the user can employ constraints to define data type properties.

To address the variety of existing SLR impact data input (raster data resolutions, geo-referencing systems and formats) and the various data output formats, we add the constraints of refinement(c). These constraints enable users to define their own preferences to use and obtain the desired input and output data. In concrete terms, we add the following constraint(s):

- **SLR-Landloss**:
  - Enforce the existence of type *RS90m* in the process, which represented in SLTL by:

$$F\,(\,\langle RS90m\rangle\,\text{true}).$$



	10	11	12	13	14	15	16	17	18	19	20	21
Domain constraints	96	144	144	288	300	312	372	456	528	672	672	672
landloss	6	6	6	6	6	6	6	6	6	6	6	6
rural/urban-damages	0	0	0	0	1	1	1	1	1	1	1	1
Yieldloss	0	0	0	0	0	0	0	0	0	12	12	12

Figure 6.8: Impact of constraints of refinement(c) on the synthesis solutions.

Where F expresses that the *RS90m* type must be hold eventually in the solutions.

– Enforce the existence of type *WG384* in the process, which represented in SLTL by:

$$F\,(\,\langle WG384\rangle\,\text{true}).$$

Where F expresses that the *WG384* type must be hold eventually in the solutions.

- **SLR-rural-urban-damages**:

– Enforce the existence of type *GDP-vector-data* in the process, , which represented in SLTL by:

$$F\,(\,\langle GDP-vector-data\rangle\,\text{true}).$$

Where F expresses that the *GDP-vector-data* type must be hold eventually in the solutions.

– Enforce the existence of type *RS90m* in the process,which represented in SLTL by:

$$F\,(\,\langle RS90m\rangle\,\text{true}).$$

Where F expresses that the *RS90m* type must be hold eventually in the solutions.

– Enforce the existence of type *WG384* in the process, which represented in SLTL by:

$$F\,(\,\langle WG384\rangle\,\text{true}).$$

Where F expresses that the *WG384* type must be hold eventually in the solutions.

– Enforce the use of module *generate-interactive-map*, which represented in SLTL by:

$$F\,(\,\langle generate-interactive-map\rangle\,\text{true}).$$

Where F expresses that the *generate-interactive-map* module must be hold eventually in the solutions.

- **SLR-yield-loss**:

  – Enforce the existence of type *GRS80* in the process, which represented in SLTL by:

  $$F\,(\,\langle GRS80\rangle\,\text{true}).$$

  Where F expresses that the *GRS80* type must be hold eventually in the solutions.

  – Avoid the type *raster30*, which represented in SLTL by:

  $$F\,(\,\langle raster30\rangle\,\text{true}).$$

  Where F expresses that the *raster30* type must be hold eventually in the solutions.

Now, lower numbers of solutions are returned when users target specific characteristics of SLR data by using the constraints of refinement(c). In this refinement, users focus more on data taxonomies to define user input or output constraints and preferences that include data input formats, scales, and different geo-referencing systems. As shown in Figure 6.8, a significant impact on synthesis solutions has been realized. Likewise, Figure 6.9 shows an explicit graphical representation of returned solutions.

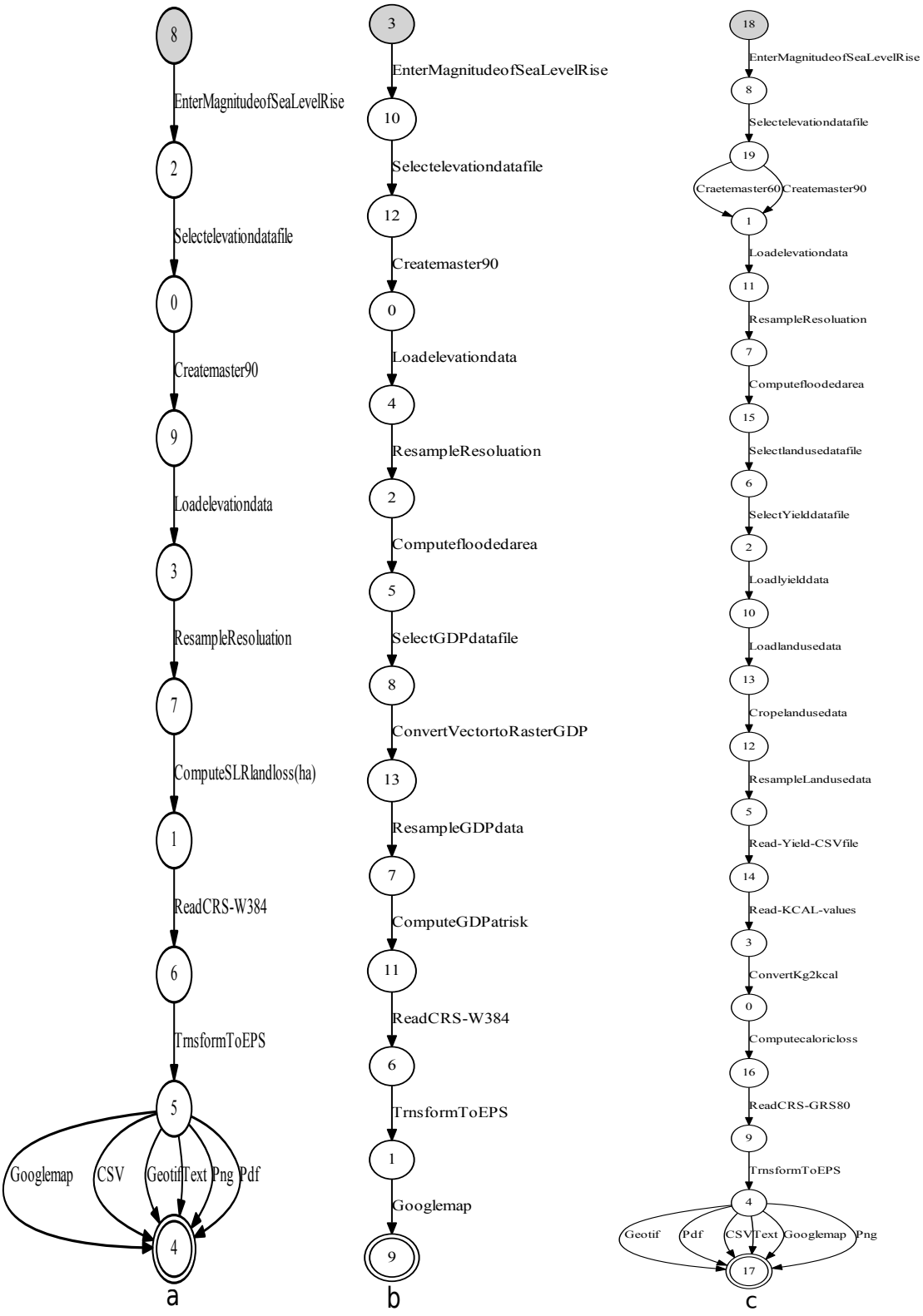


Figure 6.9: Solutions of SLR application c.

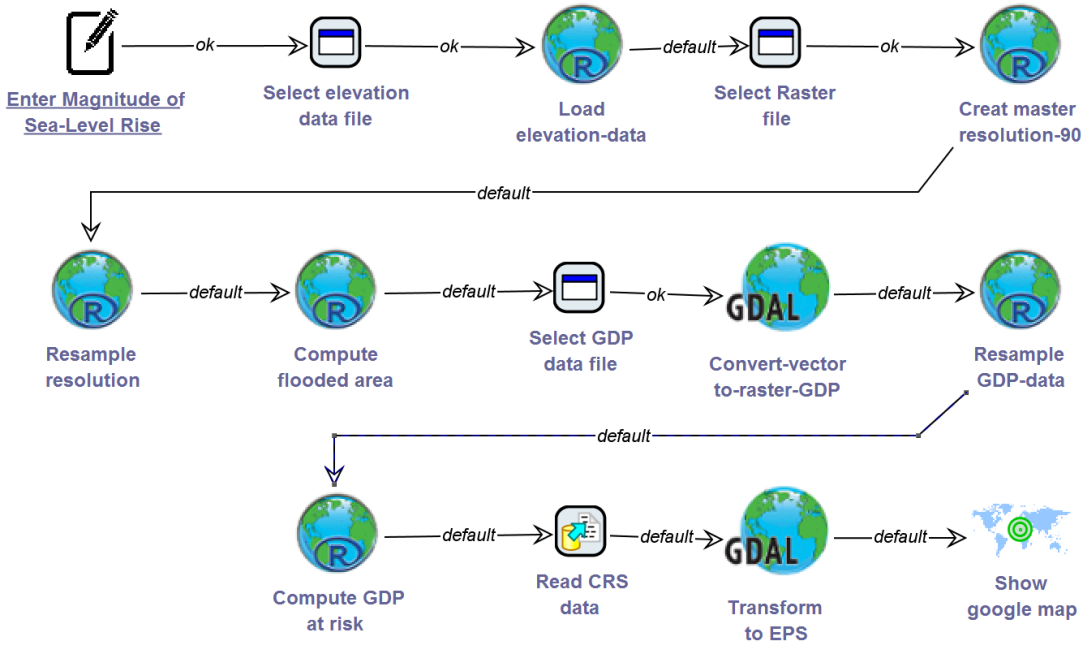


Figure 6.10: Synthesized workflow of SLR-GDPloss.

To determine the SLR-landloss in hectors and produce outputs in several output files, but based on particular data resolution and geo-referencing systems, six solutions of workflows are graphically represented in Figure 6.9a. In SLR-yieldloss workflows, the data of resolution 30 is excluded from the analysis process and the GRS geo-referencing system is used. However, no specific output file is selected; thus 12 solutions are returned and are graphically represented in Figure 6.9c.

More constraints are added to the synthesis workflows of SLR-rural-urban-damages analysis. In this analysis, we suppose that users have more specific requirements regarding data types and features. Hence, only one solution remains up until a search depth of 21 and this is graphically represented in Figure 6.9b.

Figure 6.10 shows the workflow that composes the required services for the selected solution. This solution describes the computational steps that are needed to carry out the analysis of SLR-rural-urban-damages that adequately matches the user's requirements and preferences. These steps reflect the services, already described in Table 4.2, that are required for the desired workflow design. The output results of the main computational steps are depicted in Figure 6.11

Note that there is neither a guarantee that such refinements will lead to the

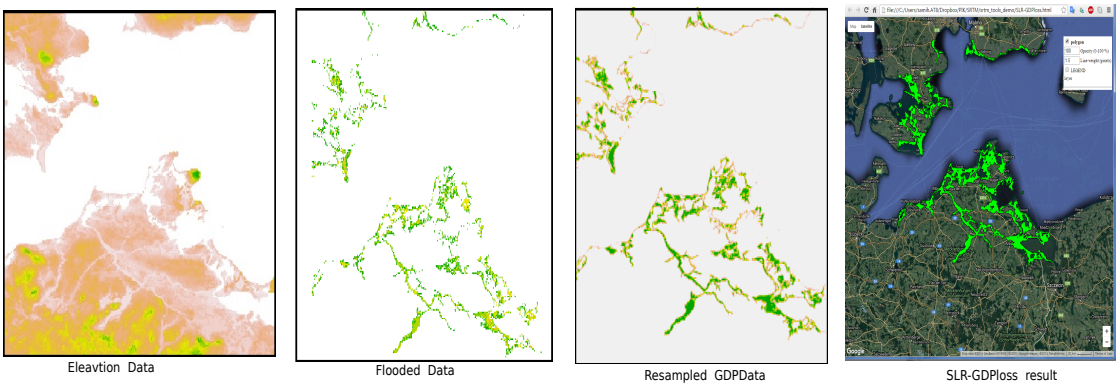


Figure 6.11: Output results of the main steps of synthesized SLR-GDPloss workflow.

exact fixed number of solutions as in the case of SLR example scenario, nor that there will be a solution at all. According to the domain model for the current specification, together with all the additional constraints, as well as the available datasets and services, the possible solutions are returned. Moreover, the user's experience and knowledge within the domain are crucial requirements for gaining an adequate number of solutions.

6.4 Handling Performance of Workflow Synthesis

Without a doubt, the runtime performance of the synthesis algorithm depends on the domain model. As described in Chapter 2, in terms of input and output data types, the directed graph is constructed to constitute the synthesis universe. This leads to the state explosion [192], which means, exponentially, that the number of states in the synthesis universe is increased in the domain model [193]. In other words, during solution exploring, more nodes are expanded to generate the synthesis universe. Hence, the overall execution time of the synthesis is reflected by the execution time of the solution exploring (searching).

According to [193], there are three main reasons for the exponential growth: (1) the existence of similar services in most application domains—that is, services consuming similar input types or producing similar output types; (2) the accu-

mulation of data with the increase in solution length; and (3) the redundancy of services.

In principle, using constraints can effectively restrict the number of visited (expanded) nodes during solution exploring, thus improving the synthesis performance. Nevertheless, adding more constraints to attain adequate solutions can also influence the performance of synthesis algorithms. Thus, it would be highly desirable if we could model the domain that handles the semantics of service composition and considers the required time of the workflow synthesis. Therefore, in the domain model of the SLR application, we address the performance of service composition by reducing the time required for the workflow synthesis in two steps: (1) improving the behavioural characterization of the service interface by defining killed types and (2) using problem-specific constraints.

Interface behavioural characterization: In SLR applications, in addition to services that have no input types and are used frequently in the same solution, a lot of general-purpose services that work on similar input types are used in SLR solutions. This increases the number of visited nodes and the elapsed time to synthesis workflows. However, prior knowledge about the behaviour of such workflows helps the domain modeller to characterize the service interface description. As in PROPHETS, by defining killed types, we can remove types that only use one time in the solution from the set of types that were available for synthesis. Hence, the number of visited nodes will be reduced and less time is required for synthesis.

Fortunately, working together with domain experts to design SLR workflows manually provides us with adequate knowledge about the behaviour of the service interface (input or output description). Accordingly, during the domain modelling of SLR applications, we used killed types to improve the behavioural characterization of the service interface (see Chapter 5). For instance, several services work on the same input types but, at most, one of these services can exist in a particular solution (workflow). Therefore, we can use killed types to eliminate input types that were already consumed by one service from the set of types that were available for synthesis.

In some cases, in order to define killed types for services that have no input values, we benefit from the isolation of the service description from its implementation in PROPHETS by defining dummy input types. However, keep in mind

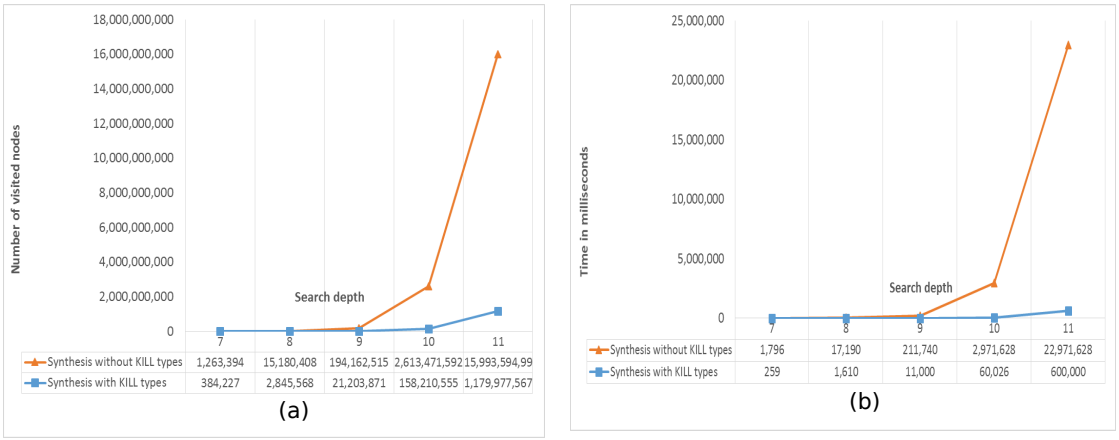


Figure 6.12: The impact of killed types on the synthesis performance of SLR workflows. (a) Number of visited nodes. (b) Elapsed time in milliseconds.

that the dummy input types should be already defined as the output for other services and these services have to be used together in the same solution.

For assessing the impact of using KILL types on the execution time of the synthesis, we carried out two experiments synthesizing the workflows of SLR applications with different settings—namely, with KILL types and without KILL types. Figure 6.12a visualizes how significantly the number of visited nodes decreases after using killed types. It shows that defining killed types reduces by more than 10 times the number of visited nodes during synthesis. Hence, as shown in Figure 6.12b, the elapsed time is also decreased in each iteration from the search depth of seven up to 11.

Despite the significant impact of using killed types on the execution time of the workflow synthesis, plenty of time is still needed to explore solutions after depth 12. For instance, exploring solutions took about 83 minutes at the search depth of 12 and we need several days to explore all solutions of SLR workflows at depth 21. In fact, using killed types is not applicable for all services, such as services where the outputs are consumed by several services in the same solution or services that work on different input types but have a similar functionality and produce the same output types (e.g. location services). Killed types are not effective in avoiding the redundancy of these services.

Domain and application constraints: Constraints are very useful for improving the synthesis performance. In other words, by targeting an adequate and

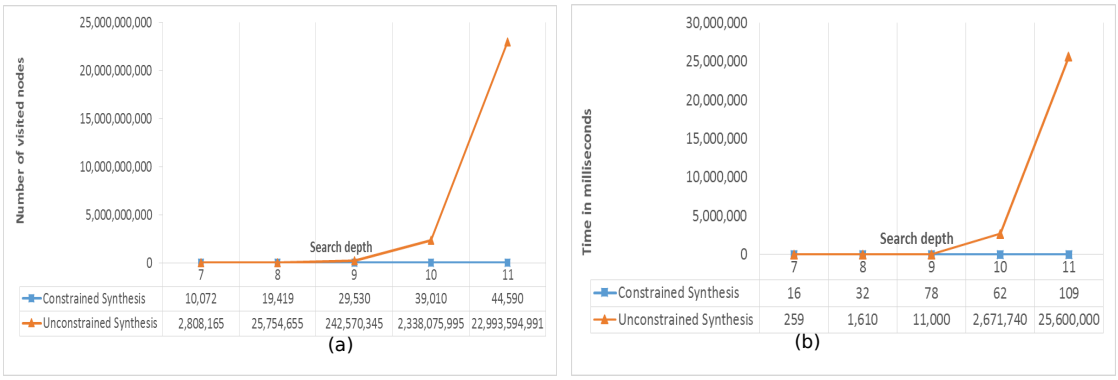


Figure 6.13: The impact of domain constraints on synthesis performance of SLR workflows. (a) Number of visited nodes. (b) Elapsed time in milliseconds.

smooth workflow design, only limited services are required; thus, manageable time can be needed for workflow synthesis. In essence, we have discussed the execution time of the synthesis during the ontology design phase. Thus, some service and type taxonomies are organized for performance issues. For example, users can benefit from the domain-specific classes of both services and types by defining the constraints to exclude a set of services or types from the synthesis.

Domain constraints provide the synthesis with more restricted information on the types and services of SLR applications: avoiding redundancy of modules, determining the last module in the solution, and defining a successive relation between modules. Therefore, alongside the use of killed types, applying domain constraints improves the synthesis performance by significantly minimizing the number of visited nodes during synthesis.

Figure 6.13a illustrates the significant impact of the domain constraints on the number of visited nodes during synthesis. After the search depth of 10, compared to the unconstrained synthesis, the results become more striking. Instead of 22,993,594,991 nodes, only 44590 nodes are visited at depth 11. Hence, as depicted in Figure 6.13b, instead of 25,600,000 milliseconds (about seven hours), we need only 190 milliseconds to explore solutions at level 11. Moreover, with domain constraints, the synthesis is aborted at depth 21 and consumed about three seconds in total to return all potential solutions of the SLR workflows.

Adding more constraints to point the synthesis to one of the main SLR analysis applications can also affect the synthesis performance. As discussed, several

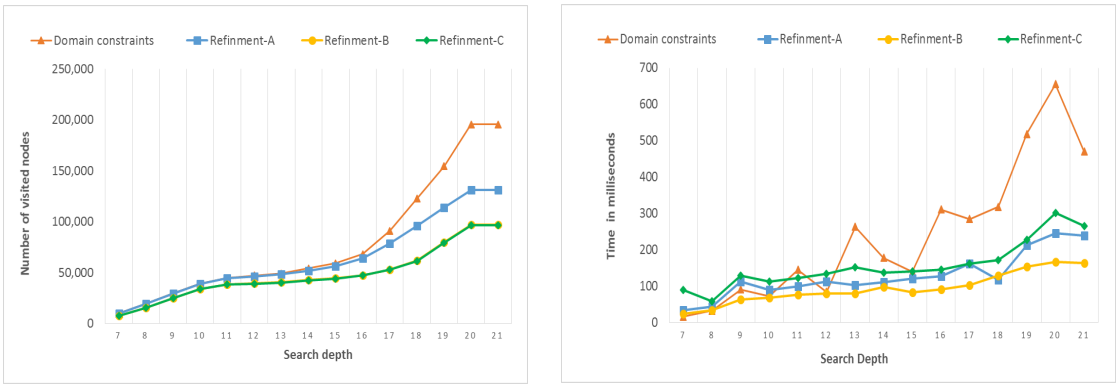


Figure 6.14: The impact of constraints of refinements(a, b, and c) on synthesis performance of SLR-landloss workflows. (a) Number of visited nodes. (b) Elapsed time in milliseconds.

refinements of constraints for specific-domain applications (SLR-landloss, SLR-rural/urban-damages, and SLR-yieldloss) are defined to target an adequate number of solutions. Basically, reducing the number of solutions leads to a reduction in the size of the search space, which speeds up the search for solutions.

Figure 6.14a illustrates how the number of visited nodes has been affected on applying the constraints of refinements(a, b, and c) to the synthesis workflows of SLR-landloss. In fact, the impact of the constraints of refinement(a) shows up after depth 11 and it increases when the synthesis discovers different alternatives of solutions for SLR workflows. For instance, while the solutions of SLR-landloss are discovered at depth 7, the solutions of SLR-rural/urban are discovered at depth 11 and the solutions of SLR-yieldloss discovered at depth 14.

The results of the application of refinements(b and c) are relatively similar and have a better impact on the number of visited nodes than refinement(a). For example, at depth 21, when the synthesis has already completed searching for all potential solutions, instead of 195,697 nodes, with domain constraints 97,042 nodes are searched with refinement(a) and 96,742 nodes with refinement(b). Accordingly, as shown in Figure 6.14b, instead of 3.5 seconds of the overall time to explore all solutions based on domain constraints, about two seconds are needed with the constraints of the SLR-landloss application.

In SLR-rural/urban-damages, we obtain relatively the same average reduction in the number of visited nodes as in SLR-landloss. Figure 6.16 shows that, at

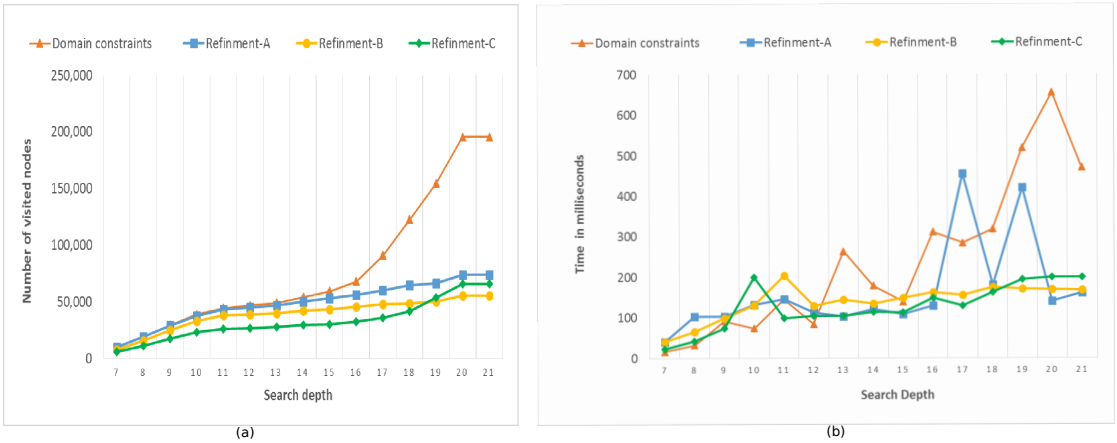


Figure 6.15: The impact of constraints of refinements(a, b, and c) on the synthesis performance of SLR-yieldloss workflows. (a) Number of visited nodes. (b) Elapsed time in milliseconds.

depth 21, the number of visited nodes reduces from 195,697 to 132,101 after applying the constraints of refinement(c) and, respectively, 96,226 and 95,516 with the constraints of refinements(b and c). Accordingly, the synthesis needs only three seconds to return one adequate solution and complete searching at depth 21. The great impact of the constraints of refinement(c) on the number of returned solutions (only one solution), the number of visited nodes, and the consumed time are relatively the same as for refinement(b). For instance, from depth seven to depth 13, Figure 6.16a visualizes the same number of visited nodes for refinements(b and c). However, the corresponding numbers of solutions are clearly different (cf. Figure 6.6 and Figure 6.8).

In principle, it is not always true that adding more constraints reduces the number of visited nodes. In some cases, the constraints allow the synthesis of new information about service and types, thus expanding the number of nodes. For example, as visualized in Figure 6.15a, when we apply the constraints of refinement(c) for SLR-yieldloss, at depths 19, 20, and 21, the number of visited nodes is greater than in refinement(b). As a consequence, Figure 6.15b illustrates that more milliseconds are needed when we apply the constraints of refinement(c).

Owing to the limited number of services and types of data used in SLR applications, there is no significant impact on the time needed compared to the domain constraints. However, we believe that when more variations of services and various

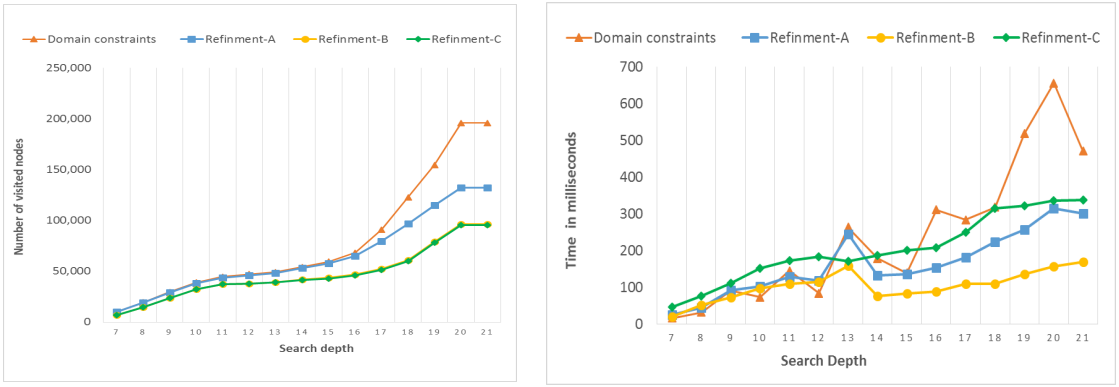


Figure 6.16: The impact of constraints of refinements(a, b, and c) on the synthesis performance of SLR-rural-urban damages workflows. (a) Number of visited nodes. (b) Elapsed time in milliseconds.

types of data are available in the SLR domain application, a viable impact of using domain-specific application constraints can be recognized. In addition, we can add constraints that are concerned with the performance issue. More precisely, based on our ontology model, users can add constraints to exclude either a specific class of services or types. For instance, in the example of SLR-rural-urban-damages application, we add the constraint (avoid the type *Agriculture data*) to the constraints of refinement(c). Therefore, a better synthesis performance is achieved. Only 14,564 nodes are visited instead of 95,516 and we need only 500 milliseconds instead of three seconds to return all potential solutions. Moreover, the synthesis completes searching at depth 15 when we add the constraint (avoid the type *Agriculture data*). Thus, no more nodes can be visited after depth 15.

In essence, in addition to the domain model that includes the behavioural description of service interface and constraints, further refinements are already applied to improve both the execution time and the accuracy of the returned solutions:

- Combining services: In substitution for several atomic services, we compose services that are only used for a particulate analysis of SLR impacts and are not reused in other SLR applications within one service. For example, the services (*Identify-AgricultureArea, Identify-Flooded-AgricultureArea, ResampledAgridata, and Resampledflooded-yielddata*) that are used to analyse the

SLR yieldloss are combined into *Compute-yieldloss*. Consequently, instead of depth 24, the synthesis reaches all potential solutions of SLR-yieldloss at depth 21.

- PROPHETS plugin configuration: In addition to the domain model, we can configure the PROPHETS plugin to improve the synthesis process. For example, this may be by removing the permutations of services from solutions, identifying the synthesis processes, and selecting the synthesis algorithm.

Chapter 7

Results and Discussion

This chapter discusses the results of this thesis by answering the main research questions: (1) How do agile workflows handle the complexity, flexibility, and reusability of geospatial service composition? (2) What characteristics of our domain model play substantial roles in addressing the semantics of geospatial service composition? (3) How are the different sorts of semantics achieved by using constraints to perform automatic geospatial service composition? This discussion also evaluates and compares the results with the state of the art in geospatial service composition.

7.1 Agile Workflows for Geospatial Applications

Despite notable efforts to migrate the traditional form of standalone geospatial applications to interoperable geospatial services, users have access only to a limited number of geo-processing services. Based on the investigation in Chapter 3, users face several barriers in consuming and composing available services:

- Most existing geospatial services are based on OGC, which suffers from the technical complexity of its standards.
- Geospatial services that are designed to handle domain-specific functionality turned into large-grained components, resulting in inflexible service reuse. Furthermore, these services are often not based on freeware tools.

- In particular, for users that have no sufficient background in programming languages, there is a lack of frameworks and tools that may be employed to reuse and compose geo-processing services.

Most existing approaches aim to handle the complexity of geospatial workflows for modelling or visualizing data between different systems or environmental models, though not from the aspect of the service-ordination paradigm. In contrast to these approaches, the layered architecture approach presented in Chapter 4 shows how the data layer, the service layer, and the workflow design layer are separated from each other. This enables users to reuse different types of service standards (whether OGC-complaint or not) and to deal with a large number of heterogeneous datasets. The three use cases of climate impact applications have already made a good impression with regard to adapting the power of the XMDD-oriented workflow development style in the geospatial domain. The use case examples showed how the functionalities underlying the ci:grasp climate impact information platform work as easily reusable services and are assessed in an adequate user-level granularity. The presented use cases show how the particularly agile workflow development style supported by the jABC process-modelling and execution framework enables users to flexibly define and perform multi-objective workflows tailored to their specific needs.

To handle the complexity and inflexibility of composing geospatial services, the presented approach is essentially characterized by:

- Rigorous *service orientation*, turning basic components into flexibly reusable pieces of functionality. We turn geo-processing components that are used for climate impact analysis into distinct processes (loose coupling services). With jETI, to simplify the integration and execution processes of services, a description of each service, equipped with well-defined inputs and outputs, is configured on the server in an easy-to-use web interface. After that, the services are generated automatically into a library of atomic geo-processing services (SIBs), so that they can easily be accessed from the jABC workflow system. During *servification*, we also consider service classification in order to distinguish domain-neutral services from domain-specific services.

- *Hierarchical modelling* that allows representing processes on different levels of abstraction (ranging from completely user-accessible top-level workflows that hide all underlying technical details to fine-grained service compositions on the lower levels). It is also possible to compose atomic services into reusable pieces of functionality. This would enable flexible and easy service consumption when composing services in agile workflows.

- The *intuitive graphical interface* makes it easy for non-programmers to design and adapt workflows according to their specific preferences and constraints. It frees end-users from the burdens of learning programming or scripting languages and other required technologies to design and adapt workflows. Thus, with jABC, users need to only drag and drop different geo-processing services and chain services into the panel to generate workflows. Then the jABC will deal with the execution of workflows.

- The *remote service integration* available on using jETI supports a convenient and flexible platform that enables users to execute geospatial services without dealing with the related configurations.

In addition to handling the complexity and the inflexibility, by using statistical methods to assist in the geospatial service reuse, the results of using XMDD technologies in climate impact applications also show how the reusability of geospatial services has improved [29].

Unlike jABC, the majority of existing workflow systems that are used in geospatial applications are concerned with the representation of data flow between systems or its components (pipeline), and are more interested in environmental models and data simulation issues than in the behavioural aspects of a workflow. In fact, such conditional branching or loops that are required for supporting the nature of scientific computations, which are needed to support the explorative nature of scientific computations, are difficult or even impossible to represent in data-flow modelling (cf., e.g. [20; 194].

Therefore, this thesis shows how to use the capabilities of the jABC modelling framework to design several possible workflow variations and instances:

- Parallel execution is also supported by a fork or join mechanism that distributes the execution flow into different threads. For instance, a user could employ a fork SIB to generate multiple output formats for the same climate impact analysis result.
- Flexible inclusion of different data. This is easiest if the data is available in a machine-readable format through the web (e.g. yield data workflows).
- A combination of an iterative execution with predefined variation points, to make it easier to vary the parameter exploration workflow.
- Interactive mechanism to keep the workflow instance alive and reuse it again with a new workflow variation.

From the other side, this allows us to address the complexity-related challenges of climate impact risk assessment discussed earlier. Instead of restricting access to a preselected dataset, the described workflows allow the user to employ datasets that are available globally and to generate tailor-made analyses. The WMS technology used in ci:grasp does not provide a powerful interface for further detailed analysis. The workflow with variation points provides flexibility in the choice of the output format and enables the user to perform unexpected additional analyses that are currently not provided through the Web page. The reproducibility of scientific analyses, which forms the core of the scientific process (cf., e.g. [113]) is directly increased. This is achieved by encouraging the sharing and reuse of code through a user-friendly and flexible environment for climate impact analysis.

Furthermore, as the original ci:grasp is a Web-based system, some described workflows have already been published over the Web by using a code generator [195] that automatically creates a server-side code and a web interface for a given jABC workflow. Thus, *arbitrary users* (the general public) who visit the ci:grasp website can access already available information or execute predefined workflows just with their own data and parameters.

Owing to the increasing tendency of using OGC services, there is no doubt that using jABC to compose OGC services will tackle the complexity-related challenges of using OGC standards. Therefore, in the bomb threat scenario [100], Figure 7.1 shows how services (WPS, WFS, and WMS) are composed in a workflow within

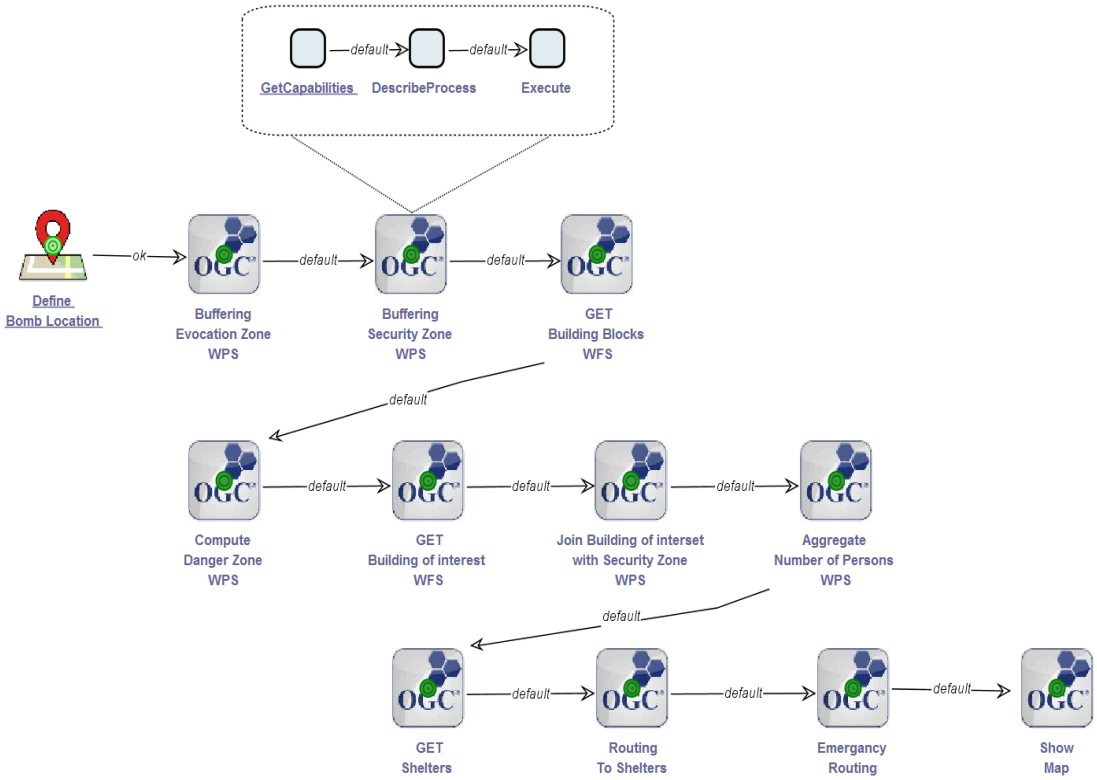


Figure 7.1: A workflow for composing OGC services within the bomb threat scenario.

the jABC framework. In fact, the bomb threat scenario is presented to show how the concept of a 'Composite-WPS' is introduced as an orchestration service for chaining other services. However, more technical efforts are required to work with WPS specifications and implementation. Instead of utilizing UML to model the interaction process between services and then using WPS to compose services, we use jABC to model the workflow that compose services for a bomb threat scenario.

In the jABC workflow of the bomb threat scenario, we show how OGC services are represented in a composite service. For instance, the WPS service, which combines the three mandatory operations—namely GetCapabilities, DescribeProcess, and Execute—can be represented in a separate (sub-)model. This is only to show that it is possible to orchestrate OGC services with jABC. However, the execution of the workflow can be achieved when its implementation is available.

7.2 Semantics-based Geospatial Service Composition

In general, the semantic issue of service composition remains a challenge. As indicated in Chapter 3, additional challenges are specific to geospatial processes and data needs, such as the heterogeneity of geospatial data types, models, and features, and the complexity of GIS operations.

In the geospatial domain application, the identification of geo-processing functions in an adequate unit of functionality is one of the most complicated tasks prior to the semantic creation of descriptions about how services can be reused and composed in a workflow. Furthermore, it is a complex task for users to manually design workflows that compose a large number of services and more effort is required from users to ensure a semantic workflow design. Hence, semantic models must be prepared to describe services and construct workflows automatically for a specific application.

As detailed with examples in Chapter 5, in contrast to existing semantic description approaches of geospatial services and data, the method presented in this thesis uses PROPHETS to model the domain of SLR applications. This model comprises a novel ontology, service interface descriptions, and constraints that are used as guides to synthesize workflows.

Novel Ontology

Since no application-specific ontological models about the services and types in the SLR application domain were available, we designed the required taxonomies ourselves—that is, they have not been derived from any existing ontology models. In addition to particular features of the concrete applications, it is often useful to incorporate knowledge from the application domain in general—in this case, the geospatial application domain. The designed ontologies essentially have the following characteristics:

- *Comprehensive models*: Unlike some existing ontology models of geospatial services that focus on taxonomies of geo-data, in our ontology model, we consider both service and type taxonomies. In the service taxonomies,

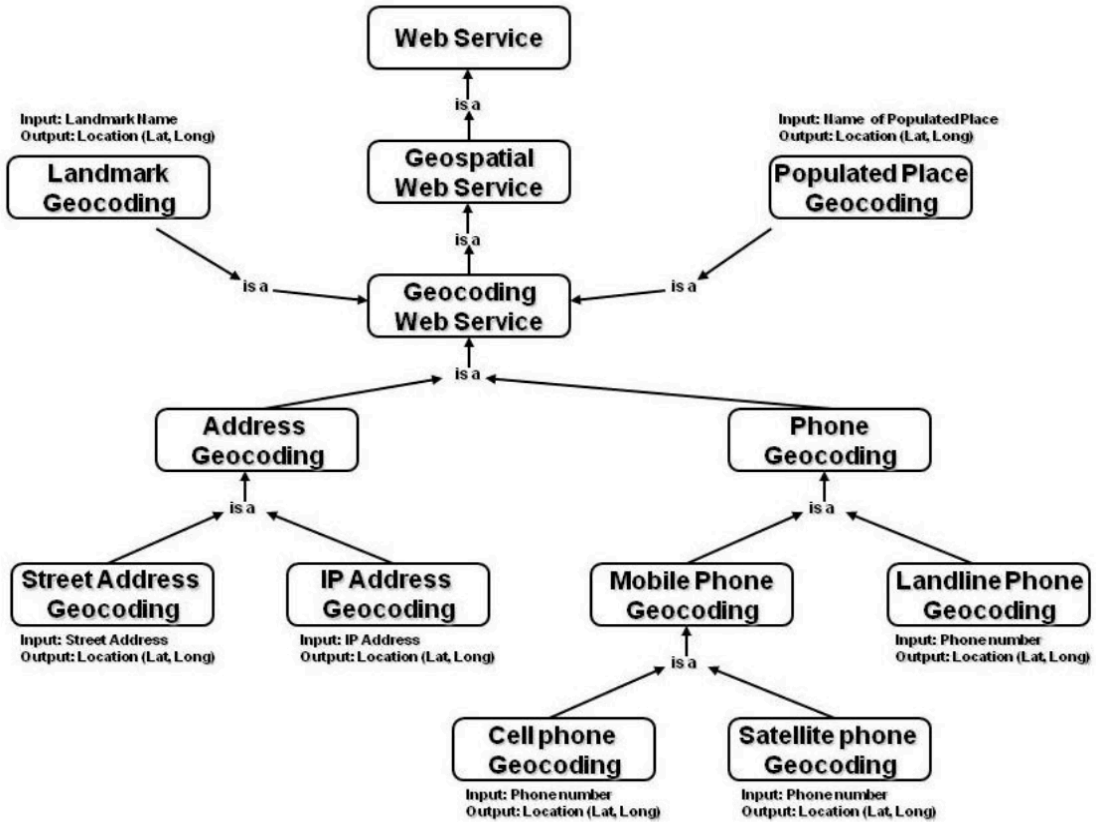


Figure 7.2: The taxonomic architecture of the semantics of geo-coding web services, adopted from [3].

possible variations of geospatial services are classified under abstract and primary classes, such as *location determining*, *data manipulation*, and *output generation*. The *Domain-specific analysis* class is defined to cover potential geospatial services that do not belong to the primary geospatial classes. In addition to the data types of services that are treated within the *Domain-specific data* class, in the type taxonomy, we also treat geospatial features (e.g. format, resolution and geo-referencing systems) as types. Thus, the proposed ontology model (service and type taxonomies) enables the user to apply a greater range of user objectives, perspectives, and input or output preferences during the synthesis process.

- *Extendable models* The abstract level of service and type taxonomies allows integrating or adding existing taxonomies (if available). For instance, Fig-

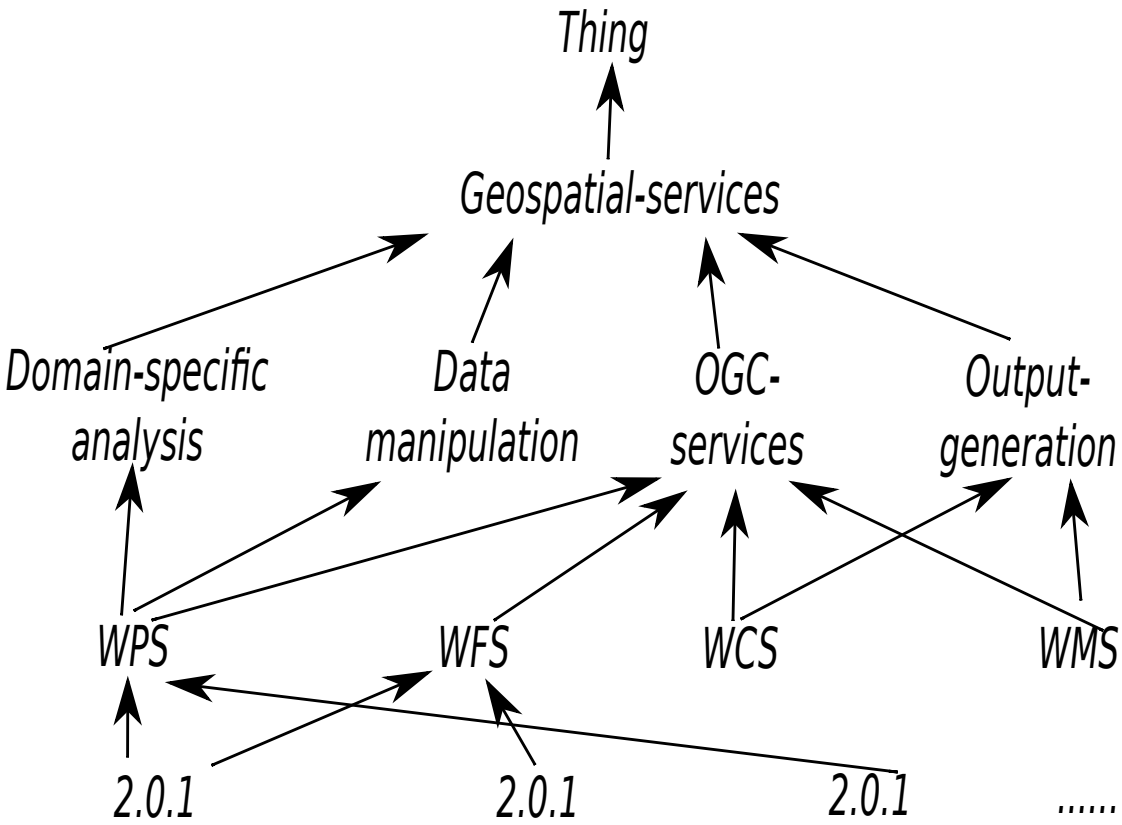


Figure 7.3: A proposed integrated architecture of the OGC services taxonomy.

ure 7.3 depicts how the OGC services can be added as a specific class of geospatial services. It shows that it is also possible for any instance of OGC services belonging to other classes of geospatial services, such as the *Data manipulation* and *Domain-specific analysis* classes. Besides that, additional taxonomies of geospatial services can be added. For example, the taxonomies presented by [3] can be integrated easily with our service taxonomy. That is, geo-coding Web services (see Figure 7.2) can be added under the geo-coding class. Data analysis services in Figure 7.4 can be assumed as a separate class of geospatial services.

Similarly in the type taxonomies, when a type taxonomy is available, it can be defined as a subclass of the *Domain-specific data* class or its subclasses. This allows users to benefit from existing related catalogues of geospatial and climate data, such as NASA's Global Change Master Directory (GCMD)[1].

[1] http://gcmd.nasa.gov/learn/keyword_list.html

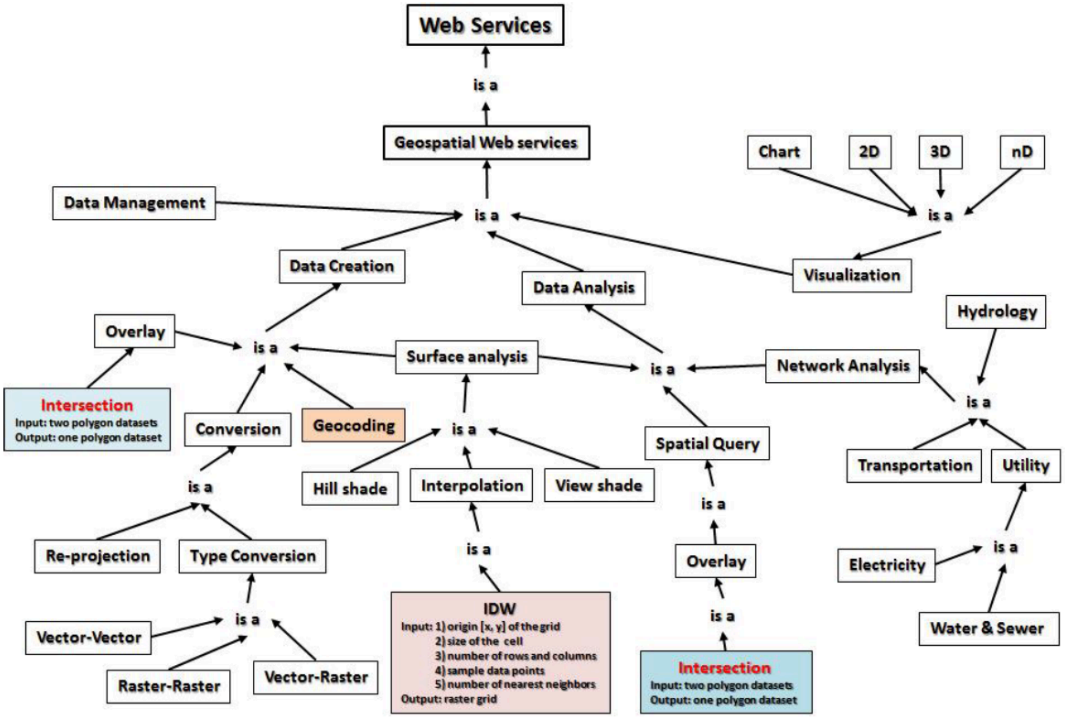


Figure 7.4: A sample geospatial services taxonomy, adopted from [3].

For instance, as shown in Figure 7.5, the GAEZ data classification can be easily integrated with the type taxonomy and added under the *Yield* type. Thus, users can apply a range of input preferences of yield data types.

Service Interface Descriptions

Compared to the existing approaches, our method of service interface description for geospatial services differs in the following ways:

- *Using symbolic names for services and data types*: Like other scientific applications, the data forms the core in the geospatial workflow process. Therefore, we are convinced that using symbolic names for describing geospatial services and data types provides the domain with precise vocabularies. Hence, an accurate description of the data types to ensure semantic service reuse and composition can be reached.

- *Decoupling semantic service descriptions*: Decoupling denotes that the service interface description and ontologies that describe services and types

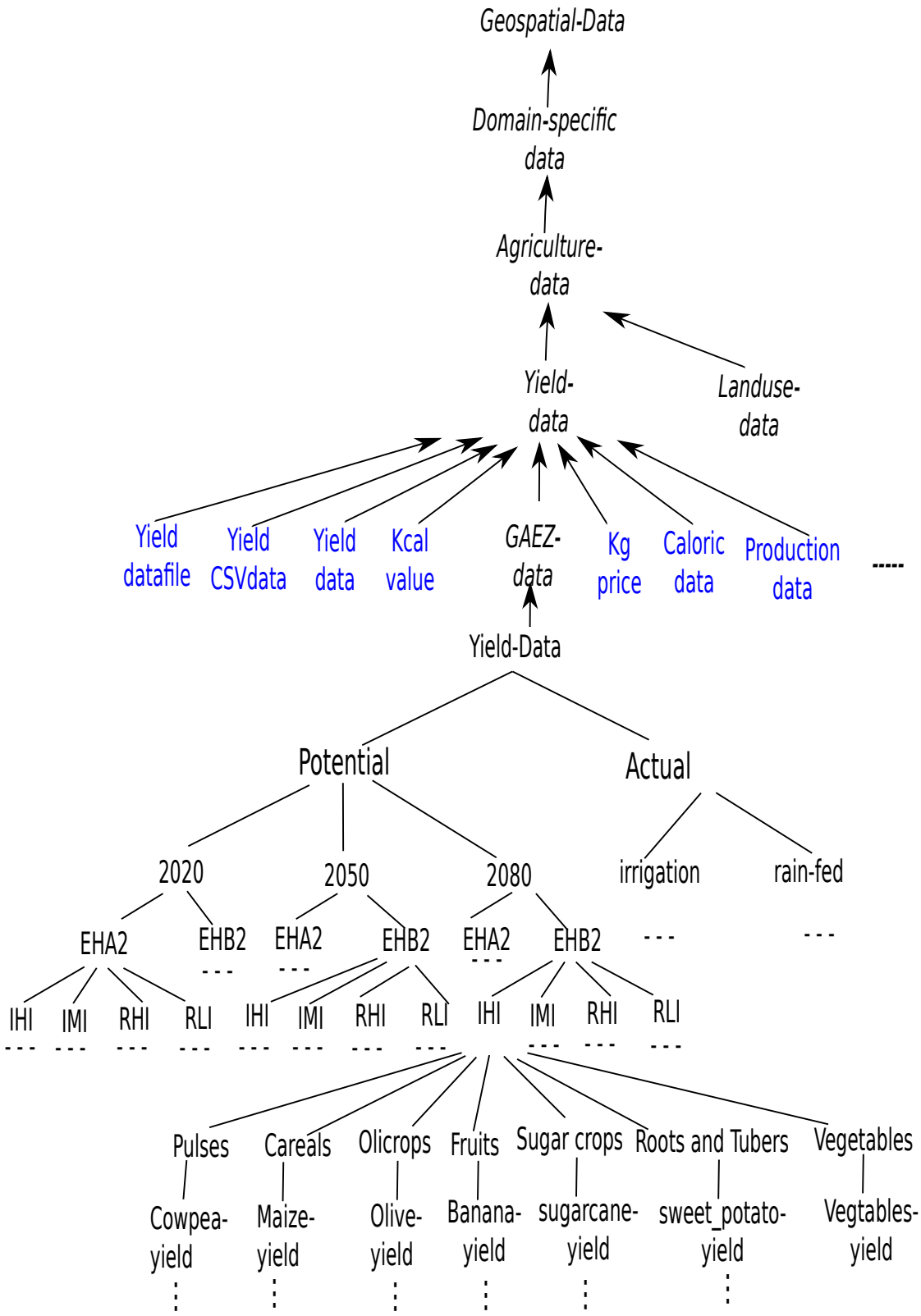


Figure 7.5: GAEZ-yield-ontology.

(semantic models) are defined in isolation, which means that each service is specified independently without regard to any problem-specific information. This enables users to employ their own terminology to describe services. Thus, several terminologies can be used to describe the same service. The experience with climate impact showed that most services that are used for SLR workflows are general-purpose services (e.g. data input and data manipulation services) that can be applied to different input data types. It also seems that some data manipulation services, such as re-sampling, format conversation, and transformation are often reused more frequently in geospatial applications. Hence, we define several modules to describe these services.

- *Behavioural aspects of service interface description*: As in PROPHETS, the synthesis algorithm relies on the behavioural service interface description to construct workflows. Therefore, we use input types (USE), output types (GEN), and kill types (KILL) to avoid or enforce particular behavioural aspects of potential SLR workflows.

- *User-centred environment*: In accordance with XMDD concepts, the domain modeller can use a sample XML editor (tag-based) to describe the service interface and use graphical interfaces to design service and type taxonomies as well as define domain constraints.

- *Domain expert involvement*: Modelling the application domain is highly dependent on domain experts. In addition to the power advantage of using XMDD technology that aims to overcome the semantic gap between domain experts and IT engineering, fortunately, we work with domain experts in climate impact analysis applications to design workflows. This allows us to define a precise vocabulary (terminology) to represent geospatial services and the data of SLR applications.

In addition to the characteristics of the ontology model and the service interface description, PROPHETS provides the dynamics of a semantic domain model. More precisely, with the PROPHETS plugin, workflow designers are able to change

the domain model according to their needs and knowledge can be obtained during the synthesis process. The change includes the editing of service interface descriptions, service and type taxonomies, and constraints. Moreover, the GNU R language provides free and very useful documentation, including examples of how to use the functions. Therefore, we benefit from the availability of R tools metadata to describe the semantic annotations of the service and data types of the domain.

7.3 Constraints-based Geospatial Workflows Synthesis

In Chapter 6, a method is presented to automatically construct geospatial service compositions (workflows) based on the preceding domain modelling of SLR applications. More specifically, in terms of workflow-based approaches, we present a constraint-driven workflow design to compose the geospatial services of SLR applications.

As discussed in Chapter 3, two main approaches have been proposed to handle the automatic composition of geospatial services: workflow-based approaches and synthesis- or planning-based approaches. In the workflow-based composition approaches, the workflow (process model) is usually designed manually to define the order of services, whereas services can be bound automatically with suitable real and concrete services. Chapter 6 shows how we use the jABC process modelling and execution to hide the complexity of the geospatial workflow development from end-users. Thus, we enable application experts who have no programming backgrounds to develop geospatial workflows. However, more and adequate knowledge is required to discover applicable services during workflow design and users face challenges in designing large workflows.

The current workflow-based approaches use semantics to support service discovery rather than the entire workflow construction. Synthesis- or planning-based approaches seem to be the most promising approach to handle automatic service composition and have already been embraced in the geospatial application domain [12; 155; 196; 197]. These approaches are almost the same; however, they use dif-

ferent domain model to describe workflow resources (e.g. services and data types). In our approach, we combine the advantages of three approaches (model-driven, workflow-based, and planning-based approaches) to achieve an advanced level of automatic service composition. More precisely, we combine an intuitive graphical formalism for the manual composition of services into workflows with additional functionality. We also shelter end-users from the complexity of defining semantic descriptions. This, through an intuitively usable plugin, helps to couple the domain model that includes semantic service descriptions with AI techniques to automatically synthesize SLR workflows.

The planning problem is defined for a loose specification branch between two abstract services of SLR applications. Therefore, an initial state and a goal state are identified. Given a domain application model and a planning problem, the synthesis determines possible solutions to achieve the goal. In essence, the synthesis algorithm naturally requires formal specifications of the synthesis problem. Nevertheless, in contrast to existing approaches, instead of specifying the planning problem in terms of some first-order or temporal logic, the workflow designer (in this case, the SLR application expert) works on intuitive and graphical interfaces. Moreover, to tailor the applied synthesis strategy to the considered application domain, the user can adjust and use many configuration options provided by PROPHETS.

In principle, in the current synthesis- or planning-based approaches, complete specifications of the target composition plan (workflow) are determined in advance. However, it will be a complicated task to realize the target service composition at an early stage. In particular, when considering the multiple models, this includes various and heterogeneous datasets as well as important characteristics of the geospatial domain. Therefore, in contrast to these approaches, in the sense of loose programming, our approach enables users through incremental formalization [198]. This means that the user can identify the domain requirements and constraints incrementally.

In the SLR applications, many solutions are technically possible and returned, but constraints are needed to exclude useless and unintended service combinations, and to guide the synthesis with regard to what is actually desired. In principle, we benefit from the incremental specification to perform the iterative adding of con-

straints. This helps the user to achieve several sorts of semantics-based geospatial service composition. For instance, in the example of SLR applications, as detailed in Chapter 6, we show how users define constraints to achieve a particular sort of semantics through several refinements:

- Refinement(a) is designed to find and compose services related to user objectives. Accordingly, we show how general-purpose services, such as manipulation services, are semantically reused in several SLR applications.
- Refinement(b) is used to identify services relevant to the user objective based on particular user perspectives. Without the need to describe its spatial semantic or geospatial topology, in this refinement, we illustrate how complex geo-processing services can be semantically reused in workflows.
- Refinement(c) focuses more on data taxonomies to define user input or output constraints and preferences. Therefore, our proposal approach allows the user to construct workflows involving complex constraints on the input or output at each composite service, such as data format, spatial resolution, and map-projection constraints. This supports the dynamic aspects of service composition by automatically composing re-sampling, transformation, and conversion services in the workflow.

Usually, defining constraints that exclude undesired solutions and/or include adequate solutions leads to a manageable set of solutions. During the workflow design phase of SLR applications, we have already demonstrated how the incremental specification strategy performed effectively to narrow the remaining solutions further. However, if the constraints demand more than the services in the domain model can provide (so-called over-specification), no solutions will be found. In this case, blocking constraints have to be removed or the domain model will have to be revised or extended by an additional functionality in order to obtain solutions again.

In addition to the incremental specification of constraints, the exploration of solutions before execution can bring several benefits to workflows. First, it provides a context to the interpretation of possible workflows before the intensive execution of the workflow. Second, it helps to discover the available quality of datasets

according to spatial resolution and map-projection constraints. Thus, the user can avoid or enforce specific data by defining additional constraints.

Often, searching only for the one shortest solution is not sufficient. In particular, as in SLR applications, most geospatial applications comprise several, but adequate workflow variants in practice. Therefore, it is feasible to increase the search depth gradually and let the user select a particular solution. For instance, in SLR applications, searching for the shortest solution returns only the solution related to SLR-landloss. However, users may be interested in other SLR applications, such as SLR-Yieldloss.

Considering the various types of geospatial data leads to the involvement of a large number of services in the workflow and more complex domain models. Accordingly, the state explosion is likely to occur to synthesize longer workflows. Consequently, an exponential runtime performance can be estimated for the synthesis. In fact, in the example of SLR applications, domain constraints have an effective impact on performance and, thus, the synthesis reaches maximum search depth in manageable time. For instance, instead of days in the unconstrained case to explore considered solutions, a couple of seconds are needed to explore all potential solutions of SLR applications. Nevertheless, it is not always true that the domain model can help reach the maximum search depth within the acceptance time. Thus, the exponential runtime performance of the synthesis can be reduced by using the divide-and-conquer-style workaround: perform two (or more) short syntheses instead of one long one [193].

Chapter 8

Conclusions and Future Work

The end-users working in scientific domains are scientists who have significant knowledge of their subject matter, but who typically lack knowledge and experience relating to principal software engineering practices. Also, from the perspective of IT experts, the design of domain-specific software and applications is a challenge, particularly in the scientific domain, which demands integrating substantial knowledge from scientists during software development. A further challenge, specifically in the geospatial domain, is considering the enormous heterogeneity of datasets and intensive computations, and addressing the multi-perspective requirements of several users.

Despite the rapid materialization of using SOA in the geospatial application domain, several barriers hinder the progress of using geospatial services. For instance, there is an obvious lack of a framework for facilitating geospatial service composition and the complexity of using standards. Thus, users face a great challenge when developing their own application based on geospatial service composition. As a result, most existing geospatial tools and data, such as climate impact analysis processes and data, are not available as services. Moreover, as the efficient communication of climate change insights to a broader audience, policy-makers, and decision-makers is both crucial and not trivial [158], developing appropriate climate information services is of high interest.

In principle, offering scientific analysis processes as services (components) at a proper level of granularity will increase the reuse of these scientific processes and can reduce the semantic gap between application experts in the scientific do-

main and IT experts. However, application experts face challenges in composing services. Based on the investigations carried out for the purpose of this thesis, the challenges stem from the complexity of existing technologies and the lack of a semantic domain model of geospatial resources. Therefore, utilizing a technology such as XMDD can resolve these challenges, as it enables domain experts to become involved in the service composition process and describe workflow resources (services and data). Nevertheless, they are facing complexity challenges in composing a correct and adequate workflow design that matches their objectives and preferences regarding data input and output.

In addition to the availability of XMDD technologies (e.g. jABC and jETI), the created services are based on scripts in the GNU R language. This allowed users to benefit from the open-source development model of the R language and from the add-on packages contributed by the R community. Thus, it shows promise for increased reuse in several geospatial application domains.

Supporting automatic services composition to design geospatial workflows has become a frequent demand among end-users (i.e. non-IT experts). As the successful application of such methodologies crucially depends on adequate domain modelling, a major part of this thesis focuses on the evaluation of domain-specific ontologies with the help of domain experts.

XMDD-based Geospatial Service Composition

Chapter 4 described how the functionality underlying the ci:grasp climate impact information platform was made available as easily reusable services, and how this enables the flexible design of climate impact analysis workflows with the jABC process design and execution framework. It also elaborated how a large unit of functionality of climate impact analysis is decomposed into the adequate level of granularity services (mostly fine-grained components). Accordingly, the scientific processes of climate change impacts and related datasets have become available as climate information services, and have been assessed to have an adequate user-level granularity. This is a relevant contribution to the community of climate impact assessment, as the variety of SLR assessment tools is still limited [199]. Furthermore, several geospatial services of general purpose are created. Thus, it

becomes possible to reuse them in various geospatial applications.

The three examples of climate impact workflows (see Section 4.2.3) are presented to show how the XMDD paradigm is used to handle the complexity and inflexibility of composing geospatial services. The examples showed how agile workflows free end-users from the burdens of learning programming and required technologies to design and adapt several variation of climate impact workflow scenarios. This addresses one of the great challenges for researchers working in the geospatial application domain, especially those who are not IT experts or trained programmers. Furthermore, the result is a key example for the usage flexibility that can be gained by following the XMDD paradigm when turning a more or less static system into a collection of services and workflows.

Moreover, Section 4.3 showed how the reuse of created geospatial services in the climate impact analysis workflows is assisted by using the statistical method in jABC. The statistics show that general-purpose services, such as data manipulation and output services, are reused more frequently. We envision that by constructing more variations in workflows, the frequency of reuse of these services will increase.

Semantic Domain Model of SLR Applications

Despite the promising results of using XMDD technologies to support the agile workflow design of geospatial services, users face challenges in constructing workflows given the multi-scale and multi-objective nature of environmental modelling for climate change impact assessment. In principle, a large number of possible workflows can be designed for SLR impacts. As a consequence, users cannot manually inspect all the services or data available and design the most suitable workflow. Therefore, the main objective of this thesis is to enable automatic geospatial service composition by annotating the application domain (in this case, SLR applications) with proper semantic characterizations (metadata) of workflow resources (e.g. services and types).

Although ontologies for supporting semantic service discoveries have seen much progress in recent years, there is still a significant need for further development. Investigations on how to arrive at such ontologies will provide us with further challenges in future. To the best of our knowledge, there is currently no such ontology

available. In fact, the Semantic Web Service Challenge [200], taking place from 2006 to 2009, basically suffered from this lack of information, and the world does not seem to have changed significantly since. Hence, a major contribution of this thesis is enabling the evaluation of new domain-specific ontologies in collaboration with experts from the SLR application domain.

As elaborated in Chapter 5, the complexity problem of the semantic description of geospatial services and data for non-IT users with no knowledge in the field of the semantic Web is addressed. This was performed through PROPHETS, which enables users to describe semantics by using (1) tag-based approach to describe the service interface, (2) graphical modelling concepts instead of formal complexity, and (3) a natural language template to specify constraints. Nevertheless, designing a domain model that comprises an appropriate semantic description of service and type is not a trivial task when employing the user language. For instance, for a specific domain application, all required resources (e.g. services and data) from which and the application experts from whom the manifold ingredients should come must be available.

Therefore, alongside the XMDD paradigm that enables users to get involved in the process of domain modelling, this thesis focuses on the more specific geospatial application domain (e.g. SLR applications). The domain knowledge is built and the service library is prepared by realizing requirements that originate from the application experts who define the system functions. In other words, we conclude that our way of modelling the domain of SLR applications introduced a promising approach to semantics-based automatic geospatial service composition.

The core components of this domain model are ontology models that are designed to realize possible variations in geospatial services and data. Thus, they address the many semantic challenges of geospatial service composition. The ontology models (service and type taxonomies) capture and represent geospatial domain knowledge, as well as classify geospatial data and services in order to achieve automation through semantic inference and constraints. To address several sorts of semantics, we define different classifications of geospatial services and data. For example, the general-purpose services, such as converting, transformation, and resampling, are classified as manipulation services and the services that are created for specific analysis are classified as domain-specific services.

Constraint-based Workflow Synthesis

Chapter 6 demonstrated how the PROPHETS synthesis plugin of the jABC workflow modelling framework can facilitate the automatic composition of workflows for analysing the impacts of SLR. In this setup, workflow designers are no longer required to implement the entire workflow manually. Instead, they can just provide a sketch of the intended workflow, together with a set of constraints that further specify the analysis objectives. PROPHETS then applies a synthesis algorithm to this abstract specification and returns a set of possible implementations to the user. The method showed how, in a user-level language and through the successive application of more and more constraints, the set of solutions that is returned by the synthesis algorithm can be reduced from an unmanageably large set of initial solutions to a manageable set of adequate solutions that match the user's intents.

In terms of loose programming, the results showed how incrementally identifying the domain requirements and constraints can successfully benefit from the ontologies for handling several semantic challenges of geospatial service composition, thus reaching an adequate solution of workflows. Moreover, rather than use standardization to handle the semantics of data compatibility, the results showed how the dynamic adaptation of conversion, re-sampling, and transformation services is performed to achieve compatibility among various formats, units, scales, and geo-referencing systems of geospatial data.

Performance of Workflow Synthesis

Increasing the number of services and data in the domain increases the execution time of workflow synthesis, which reflects the system performance. In fact, there is a performance issue relating to the workflow synthesis due to the state exploration problem. However, as elaborated in 6.4, the constraints that are used to attain adequate solutions also reduced the size of the search space, thus improving the synthesis performance.

In essence, improving performance largely depends on the domain model, specifically the ontology model and the behavioural characterization of the service interface. As in SLR applications, it showed that considering performance early in the stage of domain model design helps to add proper constraints for reducing both

the number of solutions and the size of the search space. To sum up, the presented domain model of the SLR application has a significant impact on improving the workflow synthesis performance. For instance, instead of days, we need only a couple of minutes to synthesize potential workflows.

Outlook

The international standards of OGC and ISO form the basis of most existing geospatial service compositions. However, to the best of our knowledge, there is currently no such ontology of OGC services for a particular application domain. Therefore, as a future plan, we intend to define a domain model for the example of a bomb threat scenario [100], which was discussed in Section 7.1.

Given the design principles of the domain model of SLR applications, we intend to extend the current domain model by integrating further climate impact applications, such as the city module and water resource adequacy applications described in Chapter 4. Thus, we can improve our inference about the semantic match of geospatial service composition by considering additional user analysis objectives and preferences. More information about the spatial and temporal characteristics of geospatial data can also be added to the current ontology. As a result, the number of services and types will increase in the domain. Therefore, we intend to evaluate the performance of the workflow synthesis.

Furthermore, if there were rich, multifaceted ontologies describing alternative implementations of the same service (comprising information on computation platforms, speed, resources, availability, etc.), then we could also take into account compatibility issues and preferences among the alternatives.

References

[1] Sven Jörges. Extreme model-driven development and jabc. In *Construction and Evolution of Code Generators*, pages 39–71. Springer, 2013. xi, 22, 23, 24

[2] Anders Friis-Christensen, Nicole Ostländer, Michael Lutz, and Lars Bernard. Designing service architectures for distributed geoprocessing: Challenges and future directions. *Transactions in GIS*, 11(6):799–818, 2007. xi, 2, 40, 42, 43

[3] Xuan Shi. The semantics of web services: An examination in giscience applications. *ISPRS International Journal of Geo-Information*, 2(3):888–907, 2013. xiii, 49, 125, 126, 127

[4] Ivica Crnkovic. Component-based software engineering—new challenges in software development. *Software Focus*, 2(4):127–133, 2001. 1, 12

[5] Michael J. Mineter, CH Jarvis, and Steve Dowers. From stand-alone programs towards grid-aware services and components: a case study in agricultural modelling with interpolated climate data. *Environmental Modelling & Software*, 18(4):379–391, 2003. 1, 40, 41

[6] Ron Lake and Jim Farley. Infrastructure for the geospatial web. In *The Geospatial Web*, pages 15–26. Springer, 2007. 1, 40, 41

[7] Lars Bernard and Nicole Ostländer. Assessing climate change vulnerability in the arctic using geographic information services in spatial data infrastructures. *Climatic Change*, 87(1-2):263–281, 2008. 1, 40, 41, 42, 52

[8] Carlos Granell, Laura Díaz, and Michael Gould. Service-oriented applications for environmental models: Reusable geospatial services. *Environmental Modelling & Software*, 25(2):182–198, 2010. 2, 4, 40, 43

[9] Laura Díaz, Carlos Granell, and Michael Gould. Case study: Geospatial processing services for web based hydrological applications., 2008. 2

[10] Werner Kuhn. Geospatial semantics: why, of what, and how? In *Journal on data semantics III*, pages 1–24. Springer, 2005. 2, 5, 39, 46, 47

[11] Gobe Hobona, David Fairbairn, and Philip James. Semantically-assisted geospatial workflow design. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, page 26. ACM, 2007. 2, 48

[12] Mahdi Farnaghi and Ali Mansourian. Automatic composition of wsmo based geospatial semantic web services using artificial intelligence planning. *Journal of Spatial Science*, 58(2):235–250, 2013. 2, 48, 130

[13] Jiangtao Liu, Qingyun Du, and Yi Liu. A dependable and adaptive method of geo-service composition. *Advances in Information Sciences & Service Sciences*, 4(14), 2012. 2

[14] Chuanrong Zhang, Tian Zhao, and Weidong Li. Current and future challenges of geospatial semantic web. In *Geospatial Semantic Web*, pages 167–189. Springer, 2015. 2, 43, 46, 47

[15] Sheila A McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *IEEE intelligent systems*, 3(2):46–53, 2001. 3

[16] Evren Sirin, James Hendler, and Bijan Parsia. Semi-automatic composition of web services using semantic descriptions. In *1st Workshop on Web Services: Modeling, Architecture and Infrastructure*, pages 17–24, 2003. 3

[17] Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubén Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Cristoph Bussler, and Dieter Fensel. Web service modeling ontology. *Applied ontology*, 1(1):77–106, 2005. 4, 20

[18] I Budak Arpinar, Ruoyan Zhang, Boanerges Aleman-Meza, and Angela Maduko. Ontology-driven web services composition platform. *Information Systems and E-Business Management*, 3(2):175–199, 2005. 4

[19] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, and Dana Nau. Htn planning for web service composition using shop2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396, 2004. 4

[20] Anna-Lena Lamprecht. *User-Level Workflow Design - A Bioinformatics Perspective*, volume 8311. Springer, 2013. 4, 27, 29, 30, 32, 38, 121

[21] Carlos Granell, Rob Lemmens, Michael Gould, Andreas Wytzisk, Rolf De By, and Peter Van Oosterom. Integrating semantic and syntactic descriptions to chain geographic services. *Internet Computing, IEEE*, 10(5):42–52, 2006. 4, 41

[22] C Zhang, W Li, and T Zhao. Geospatial data sharing based on geospatial semantic web technologies. *Journal of Spatial Science*, 52(2):35–49, 2007. 4

[23] Greg Wilson, DA Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, et al. Best practices for scientific computing. *PLoS biology*, 12(1):e1001745, 2014. 5, 52

[24] Samih Al-areqi, S Kriewald, AL Lamprecht, D Reusser, M Wrobel, and T Margaria. Agile Workflows for Climate Impact Risk Assessment based on the ci: grasp Platform and the jABC Modeling Framework. In *International Environmental Modelling and Software Society (iEMSs) 7th Intl. Congress on Env. Modelling and Software (published, 2014)*, 2014. 7, 53, 58

[25] Christian Kubczak, Sven Jörges, Tiziana Margaria, and Bernhard Steffen. eXtreme Model-Driven Design with jABC. In *CTIT Proc. of the Tools and Consultancy Track of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA)*, volume WP09-12, pages 78–99, 2009. 7

[26] Tiziana Margaria and Bernhard Steffen. Service-Orientation: Conquering Complexity with XMDD. In Mike Hinchey and Lorcan Coyle, editors, *Conquering Complexity*, pages 217–236. Springer London, 2012. 7, 21

[27] Bernhard Steffen, Tiziana Margaria, Ralf Nagel, Sven Jörges, and Christian Kubczak. Model-Driven Development with the jABC. In Eyal Bin, Avi Ziv, and Shmuel Ur, editors, *Hardware and Software, Verification and Testing*, volume 4383. Springer Berlin / Heidelberg, 2007. 7, 24, 60

[28] Samih Al-Areqi, Steffen Kriewald, Anna-Lena Lamprecht, Dominik Reusser, Markus Wrobel, and Tiziana Margaria. Towards a flexible assessment of climate impacts: The example of agile workflows for the ci: grasp platform. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, pages 420–435. Springer, 2014. 8, 61

[29] Samih Al-Areqi, Anna-Lena Lamprecht, and Tiziana Margaria. Improving the reuse of services in geospatial applications with xmdd technology, 2014. 8, 121

[30] Anna-Lena Lamprecht, Stefan Naujokat, Tiziana Margaria, and Bernhard Steffen. Synthesis-based loose programming. In *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, pages 262–267. IEEE, 2010. 8, 27, 34

[31] Stefan Naujokat, Anna-Lena Lamprecht, and Bernhard Steffen. Loose Programming with PROPHETS. In Juan de Lara and Andrea Zisman, editors, *Proc. of the 15th Int. Conf. on Fundamental Approaches to Software Engineering (FASE 2012), Tallinn, Estonia*, volume 7212 of *LNCS*, pages 94–98. Springer Heidelberg, 2012. 8

[32] Samih Al-Areqi, Anna-Lena Lamprecht, and Tiziana Margaria. Automatic workflow composition in the geospatial domain: an application on sea-level rise impacts analysis, 2016. 8

[33] Samih Al-Areqi, Anna-Lena Lamprecht, and Tiziana Margaria. Constraints-driven automatic geospatial service composition: workflows for the analysis

of sea-level rise impacts. In *International Conference on Computational Science and Its Applications*, pages 134–150. Springer, 2016. 8

[34] Bran Selic. The pragmatics of model-driven development. *IEEE software*, 20(5):19, 2003. 11, 21

[35] Charles W Krueger. Software reuse. *ACM Computing Surveys (CSUR)*, 24(2):131–183, 1992. 11

[36] Gary Haines, David Carney, and John Foreman. Component-based software development/cots integration. *SEI Technology Description. Pittsburgh PA: Software Engineering Institute, Carnegie Mellon University*, 1997. 11

[37] Jeffrey S Poulin. Measuring software reusability. In *Software Reuse: Advances in Software Reusability, 1994. Proceedings., Third International Conference on*, pages 126–138. IEEE, 1994. 12

[38] Jim Q Ning. A component-based software development model. In *Proceedings of the 20th Conference on Computer Software and Applications*, page 389. IEEE Computer Society, 1996. 12

[39] Peter Freeman. Reusable software engineering: Concepts and research directions. In *ITT Proceedings of the Workshop on Reusability in Programming*, volume 129, page 137, 1983. 12

[40] Nasib S Gill. Reusability issues in component-based development. *ACM SIGSOFT Software Engineering Notes*, 28(4):4–4, 2003. 12, 13

[41] Kenn Gardels. The open gis approach to distributed geodata and geoprocessing. In *Third International Conference/Workshop on Integrating GIS and Environmental Modeling, Santa Fe, NM*, pages 21–25, 1996. 13

[42] Hongyu Pei Breivold and Magnus Larsson. Component-based and service-oriented software engineering: Key concepts and principles. In *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on*, pages 13–20. IEEE, 2007. 13, 15

[43] Padmal Vitharana. Risks and challenges of component-based software development. *Communications of the ACM*, 46(8):67–72, 2003. 13

[44] Jack Greenfield and Keith Short. Software factories: assembling applications with patterns, models, frameworks and tools. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 16–27. ACM, 2003. 13

[45] I. Sommerville. *Software Engineering*. International computer science series. Addison-Wesley, 2007. 14, 16

[46] Thomas Erl. *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice Hall PTR, 2004. 14

[47] Wei-Tek Tsai. Service-oriented system engineering: a new paradigm. In *IEEE International Workshop on Service-Oriented System Engineering (SOSE'05)*, pages 3–6. IEEE, 2005. 14

[48] Michael P Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: a research roadmap. *International Journal of Cooperative Information Systems*, 17(02):223–255, 2008. 14

[49] Asit Dan, Robert D Johnson, and Tony Carrato. Soa service reuse by design. In *Proceedings of the 2nd international workshop on Systems development in SOA environments*, pages 25–28. ACM, 2008. 14

[50] Tiziana Margaria and Bernhard Steffen. Agile it: thinking in user-centric models. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, pages 490–502. Springer, 2008. 15, 21, 24

[51] Leonard Richardson and Sam Ruby. *RESTful web services*. " O'Reilly Media, Inc.", 2008. 16

[52] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. 16

[53] Fabio Casati, Mehmet Sayal, and Ming-Chien Shan. Developing e-services for composing e-services. In *International Conference on Advanced Information Systems Engineering*, pages 171–186. Springer, 2001. 16

[54] German Shegalov, Michael Gillmann, and Gerhard Weikum. Xml-enabled workflow management for e-services across heterogeneous platforms. *The VLDB Journal—The International Journal on Very Large Data Bases*, 10(1):91–103, 2001. 16

[55] Nikola Milanovic and Miroslaw Malek. Current solutions for web service composition. *IEEE Internet Computing*, 8(6):51, 2004. 18

[56] Anne Immonen and Daniel Pakkala. A survey of methods and approaches for reliable dynamic service compositions. *Service Oriented Computing and Applications*, 8(2):129–158, 2014. 18

[57] Jinghai Rao and Xiaomeng Su. A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer, 2004. 18

[58] Quan Z Sheng, Xiaoqiang Qiao, Athanasios V Vasilakos, Claudia Szabo, Scott Bourne, and Xiaofei Xu. Web services composition: A decade's overview. *Information Sciences*, 280:218–238, 2014. 18

[59] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5):907–928, 1995. 19

[60] Nicola Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*, pages 139–170. Springer, 1997. 19

[61] NF Roy and CD Hafner. The state of the art in ontology design. *AI Magazine*, 18(3):53–74, 1997. 19

[62] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001. 19

[63] Sean Bechhofer. Owl: Web ontology language. In *Encyclopedia of Database Systems*, pages 2008–2009. Springer, 2009. 19

[64] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, et al. Owl-s: Semantic markup for web services. *W3C member submission*, 22:2007–04, 2004. 19

[65] Holger Lausen, Jos de Bruijn, Axel Polleres, and Dieter Fensel. Wsml-a language framework for semantic web services. In *Rule Languages for Interoperability*, 2005. 20

[66] Ingo Weber, Hye-Young Paik, and Boualem Benatallah. Form-based web service composition for domain experts. *ACM Transactions on the Web (TWEB)*, 8(1):2, 2013. 20

[67] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for Agile Software Development. http://agilemanifesto.org, 2001. [Online; last accessed 4-March-2014]. 20

[68] A Elssamadisy. Soa and agile: Friends or foes? *Retrieved February*, 20:2009, 2007. 20

[69] Tiziana Margaria and Bernhard Steffen. Continuous model-driven engineering. *Computer*, 42(10):106–109, 2009. 20

[70] Apostolos Zarras, Panos Vassiliadis, and Valérie Issarny. Model-driven dependability analysis of webservices. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1608–1625. Springer, 2004. 21

[71] Bart Orriëns, Jian Yang, and Mike P Papazoglou. Model driven service composition. In *International Conference on Service-Oriented Computing*, pages 75–90. Springer, 2003. 21

[72] Tiziana Margaria and Bernhard Steffen. Business process modelling in the jabc: the one-thing-approach. *Handbook of research on business process modeling*, pages 1–26, 2009. 21, 23, 24

[73] Tiziana Margaria and Bernhard Steffen. Service-orientation: conquering complexity with xmdd. In *Conquering Complexity*, pages 217–236. Springer, 2012. 21

[74] Bernhard Steffen, Tiziana Margaria, Ralf Nagel, Sven Jörges, and Christian Kubczak. Model-driven development with the jABC. In *Hardware and Software, Verification and Testing*, pages 92–108. Springer, 2007. 24

[75] Anna-Lena Lamprecht and Tiziana Margaria. *Process Design for Natural Scientists - An Agile Model-Driven Approach*, volume 500. Springer Berlin Heidelberg, 2014. 24

[76] Georg Jung, Tiziana Margaria, Ralf Nagel, Wolfgang Schubert, Bernhard Steffen, and Horst Voigt. SCA and jABC: Bringing a service-oriented paradigm to web-service construction. In *Leveraging Applications of Formal Methods, Verification and Validation*, pages 139–154. Springer, 2009. 24

[77] Anna-Lena Lamprecht, Bernhard Steffen, and Tiziana Margaria. Scientific workflows with the jabc framework. *International Journal on Software Tools for Technology Transfer*, pages 1–23, 2016. 24

[78] Tiziana Margaria and Bernhard Steffen. Lightweight coarse-grained coordination: a scalable system-level approach. *International Journal on Software Tools for Technology Transfer*, 5(2-3):107–123, 2004. 24

[79] Tiziana Margaria, Ralf Nagel, and Bernhard Steffen. jETI: A tool for remote tool integration. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 557–562. Springer, 2005. 26

[80] Tiziana Margaria, Ralf Nagel, and Bernhard Steffen. Remote integration and coordination of verification tools in jeti. In *12th IEEE International*

Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), pages 431–436. IEEE, 2005. 26

[81] Bernhard Steffen, Tiziana Margaria, and Volker Braun. The Electronic Tool Integration platform: concepts and design. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):9–30, 1997. 26

[82] Stefan Naujokat, Anna-Lena Lamprecht, and Bernhard Steffen. Loose programming with prophets. In *International Conference on Fundamental Approaches to Software Engineering*, pages 94–98. Springer, 2012. 27

[83] Bernhard Steffen, Tiziana Margaria, and Burkhard Freitag. Module Configuration by Minimal Model Construction. Technical report, Fakultät für Mathematik und Informatik, Universität Passau, 1993. 28

[84] Zohar Manna and Pierre Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 6(1):68–93, 1984. 33

[85] Bernhard Steffen, Tiziana Margaria, and Burkhard Freitag. Module configuration by minimal model construction, 1993. 33, 34

[86] Ian Masser. *GIS worlds: creating spatial data infrastructures*, volume 338. ESRI press Redlands, CA, 2005. 39

[87] Christian Kiehle, Klaus Greve, and Christian Heier. Standardized geoprocessing–taking spatial data infrastructures one step further. In *Proceedings of the 9th AGILE International Conference on Geographic Information Science. Visegrád, Hungary*, 2006. 39, 42

[88] Chuanrong Zhang, Tian Zhao, and Weidong Li. Geospatial data interoperability, geography markup language (gml), scalable vector graphics (svg), and geospatial web services. In *Geospatial Semantic Web*, pages 1–33. Springer, 2015. 40, 44

[89] Weiguo Han, Liping Di, Genong Yu, Yuanzheng Shao, and Lingjun Kang. Investigating metrics of geospatial web services: The case of a ceos federated

catalog service for earth observation data. *Computers & Geosciences*, 92:1–8, 2016. 40

[90] Nadine Alameh. Chaining geographic information web services. *IEEE Internet Computing*, 7(5):22, 2003. 40, 42

[91] Theodor Foerster, Bastian Schaeffer, Johannes Brauner, and Simon Jirka. Integrating OGC web processing services into geospatial mass-market applications. In *Advanced Geographic Information Systems & Web Services, 2009. GEOWS'09. International Conference on*, pages 98–103. IEEE, 2009. 40, 42

[92] Victoria Rautenbach, Serena Coetzee, and Adam Iwaniak. Orchestrating OGC web services to produce thematic maps in a spatial information infrastructure. *Computers, Environment and Urban Systems*, 37:107–120, 2013. 40, 44

[93] Simon Cox, Paul Daisey, Ron Lake, Clemens Portele, and Arliss Whiteside. Geography markup language (gml) implementation specification version 3.0. *OpenGIS Consortium*, 2003. 41

[94] Jeff de La Beaujardiere. Opengis® web map server implementation specification. *Open Geospatial Consortium Inc., OGC*, pages 06–042, 2006. 41

[95] OGC. Opengis® web map server implementation specification (version: 1.3. 0), 2010. 41

[96] Panagiotis A Vretanos. Opengis web feature service 2.0 interface standard. *Open Geospatial Consortium Inc, Version*, 2(0), 2010. 41

[97] Peter Baumann. Ogc wcs 2.0 interface standard—core. *Open Geospatial Consortium Inc., Wayland, MA, USA, OpenGIS® Interface Standard OGC*, 2010. 41

[98] Kristian Senkler, Uwe Voges, and Albert Remke. An iso 19115/19119 profile for ogc catalogue services csw 2.0. In *workshop paper presented at 10th EC-GI & GIS workshop, Warsaw, Poland*, 2004. 41

[99] Peter Schut and A Whiteside. Opengis web processing service. *OGC project document*, 2007. 41

[100] Beate Stollberg and Alexander Zipf. Ogc web processing service interface for web service orchestration aggregating geo-processing services in a bomb threat scenario. In *Web and wireless geographical information systems*, pages 239–251. Springer, 2007. 41, 42, 122, 140

[101] Anthony M Castronova, Jonathan L Goodall, and Mostafa M Elag. Models as web services using the open geospatial consortium (OGC) web processing service (WPS) standard. *Environmental Modelling & Software*, 41:72–83, 2013. 42

[102] Jonathan L Goodall, Bella F Robinson, and Anthony M Castronova. Modeling water resource systems using a service-oriented computing paradigm. *Environmental Modelling & Software*, 26(5):573–582, 2011. 42

[103] Fei Xiao, Geoffrey YK Shea, Man Sing Wong, and James Campbell. An automated and integrated framework for dust storm detection based on ogc web processing services. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(2):151, 2014. 42

[104] C Kiehle and T Foerster. Ows-7 web processing service profiling engineering report. *OGC Document*, 2010. 42

[105] Alain Tamayo, Carlos Granell, and Joaquín Huerta. Measuring complexity in ogc web services xml schemas: pragmatic use and solutions. *International Journal of Geographical Information Science*, 26(6):1109–1130, 2012. 42

[106] Francisco J Lopez-Pellicer, Walter RenteríA-Agualimpia, Rubén Béjar, Pedro R Muro-Medrano, and F Javier Zarazaga-Soria. Availability of the ogc geoprocessing standard: March 2011 reality check. *Computers & Geosciences*, 47:13–19, 2012. 42

[107] INSPIRE Directive. Directive 2007/2/ec of the european parliament and of the council of 14 march 2007 establishing an infrastructure for spatial

information in the european community (inspire). *Published in the official Journal on the 25th April*, 2007. 43

[108] Bertram Ludäscher, Kai Lin, Shawn Bowers, Efrat Jaeger-Frank, Boyan Brodaric, and Chaitan Baru. Managing scientific data: From data integration to scientific workflows*. *Geological Society of America Special Papers*, 397:109–129, 2006. 43, 44

[109] Jia Yu and Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *ACM Sigmod Record*, 34(3):44–49, 2005. 44

[110] Ian J Taylor, Ewa Deelman, Dennis Gannon, Matthew Shields, et al. *Workflows for e-Science*. Springer-Verlag London Limited, 2007. 44

[111] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009. 44

[112] Yan Liu, Ian Gorton, and Adam Wynne. Architecture-Based Adaptivity Support for Service Oriented Scientific Workflows. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pages 309–314. IEEE, 2013. 44, 45

[113] Bertram Ludäscher, Ilkay Altintas, Shawn Bowers, Julian Cummings, Terence Critchlow, Ewa Deelman, David D Roure, Juliana Freire, Carole Goble, Matthew Jones, et al. Scientific process automation and workflow management. *Scientific Data Management: Challenges, Existing Technology, and Deployment, Computational Science Series*, pages 476–508, 2009. 44, 122

[114] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *ACM Sigmod Record*, 34(3):31–36, 2005. 44

[115] Ji Liu, Esther Pacitti, Patrick Valduriez, and Marta Mattoso. A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, 13(4):457–493, 2015. 44

[116] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific Workflow Management and the Kepler Systems. *Concurrency and Computation: Practice & Experience*, 18:1039–1065, 2006. 44

[117] S.P. Callahan, J. Freire, J. Freire, E. Santos, C.E. Scheidegger, C.T. Silva, and H.T. Vo. Managing the evolution of dataflows with vistrails. In *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pages 71–71, 2006. 44

[118] Gustavo Alonso and Claus Hagen. Geo-Opera: Workflow concepts for spatial processes. In *Advances in Spatial Databases*, pages 238–258. Springer, 1997. 44

[119] Tino Fleuren and Philipp Muller. Bpel workflows combining standard ogc web services and grid-enabled ogc web services. In *Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference*, pages 337–344. IEEE, 2008. 44

[120] Genong Yu, Liping Di, Bei Zhang, and Huilin Wang. Coordination through geospatial web service workflow in the sensor web environment. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 3(4):433–441, 2010. 44

[121] Nengcheng Chen, Liping Di, Genong Yu, and Jianya Gong. Geo-processing workflow driven wildfire hot pixel detection under sensor web environment. *Computers & Geosciences*, 36(3):362–372, 2010. 44, 45

[122] Gobe Hobona, David Fairbairn, Hugo Hiden, and Philip James. Orchestration of grid-enabled geospatial web services in geoscientific workflows. *Automation Science and Engineering, IEEE Transactions on*, 7(2):407–411, 2010. 44

[123] Genong Eugene Yu, Peisheng Zhao, Liping Di, Aijun Chen, Meixia Deng, and Yuqi Bai. Bpelpower—a bpel execution engine for geospatial web services. *Computers & Geosciences*, 47:87–101, 2012. 44

[124] Jorge de Jesus, Peter Walker, Michael Grant, and Steve Groom. Wps orchestration using the taverna workbench: The escience approach. *Computers & Geosciences*, 47:75–86, 2012. 44

[125] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R Pocock, Anil Wipat, et al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004. 44

[126] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006. 45

[127] Efrat Jaeger, Ilkay Altintas, Jianting Zhang, Bertram Ludäscher, Deana Pennington, and William Michener. A Scientific Workflow Approach to Distributed Geospatial Data Processing using Web Services. In *SSDBM*, pages 87–90. Citeseer, 2005. 45

[128] Jianting Zhang, Deana D Pennington, and William K Michener. Validating compositions of geospatial processing web services in a scientific workflow environment. In *null*, pages 821–822. IEEE, 2005. 45

[129] Jianting Zhang. Tracking dynamics of geospatial phenomena in distributed and heterogeneous environments using scientific workflow and web services technologies. In *Grid and Cooperative Computing, 2006. GCC 2006. Fifth International Conference*, pages 474–481. IEEE, 2006. 45

[130] Ufuk Utku Turuncoglu, Nuzhet Dalfes, Sylvia Murphy, and Cecelia DeLuca. Toward self-describing and workflow integrated earth system models: A coupled atmosphere-ocean modeling system application. *Environmental modelling & software*, 39:247–262, 2013. 45

[131] Andrew Pratt, Chris Peters, G Siddeswara, Brad Lee, and A Terhorst. Exposing the Kepler scientific workflow system as an OGC web processing service. *Proceedings of iEMSs (International Environmental Modelling and Software Society)*, 2010, 2010. 45

[132] David A Swayne, Wanhong Yang, AA Voinov, A Rizzoli, and T Filatova. End-to-end workflows for coupled climate and hydrological modeling, 2010. 45

[133] Emanuele Santos, David Koop, Thomas Maxwell, Charles Doutriaux, Tommy Ellqvist, Gerald Potter, Juliana Freire, Dean Williams, and Cláudio T Silva. Designing a provenance-based climate data analysis application. In *Provenance and Annotation of Data and Processes*, pages 214–219. Springer, 2012. 45

[134] Jianting Zhang. A practical approach to developing a web-based geospatial workflow composition and execution system. In *Proceedings of the 3rd International Conference on Computing for Geospatial Research and Applications*, page 21. ACM, 2012. 45

[135] Peng Yue, Mingda Zhang, and Zhenyu Tan. A geoprocessing workflow system for environmental monitoring and integrated modelling. *Environmental Modelling & Software*, 69:128–140, 2015. 45

[136] Sam Meek, Mike Jackson, and Didier G Leibovici. A bpmn solution for chaining ogc services to quality assure location-based crowdsourced data. *Computers & Geosciences*, 87:76–83, 2016. 45

[137] Amit P Sheth. Semantic web process lifecycle: role of semantics in annotation, discovery, composition and orchestration, 2003. 46

[138] U Voges and K Senkler. Opengis catalogue services specification 2.0–iso19115/iso19119 application profile for csw 2.0 opengis project document 04-038r2, 2004. 46

[139] Peng Yue. Semantics-enabled geospatial data and services discovery. In *Semantic Web-based Intelligent Geospatial Web Services*, pages 49–63. Springer, 2013. 46, 47

[140] Mihai Andrei, Arne Berre, Luis Costa, Philippe Duchesne, Daniel Fitzner, Miha Grcar, Jörg Hoffmann, Eva Klien, Joel Langlois, Andreas Limyr, et al.

Swing: An integrated environment for geospatial semantic web services. In *European Semantic Web Conference*, pages 767–771. Springer, 2008. 47

[141] Krzysztof Janowicz, Sven Schade, Arne Bröring, Carsten Keßler, Patrick Maué, and Christoph Stasch. Semantic enablement for spatial data infrastructures. *Transactions in GIS*, 14(2):111–129, 2010. 47

[142] Clifford A Kottman. The open gis consortium and progress toward interoperability in gis. In *Interoperating geographic information systems*, pages 39–54. Springer, 1999. 47

[143] Holger Wache, Thomas Voegele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner. Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, pages 108–117. Citeseer, 2001. 47

[144] Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, and Thomas Vögele. Ontologies for geographic information processing. *Computers & Geosciences*, 28(1):103–117, 2002. 47

[145] Eva Klien, Michael Lutz, and Werner Kuhn. Ontology-based discovery of geographic information services—an application in disaster management. *Computers, environment and urban systems*, 30(1):102–123, 2006. 47

[146] Tom Broens, Stanislav Pokraev, Marten Van Sinderen, Johan Koolwaaij, and Patricia Dockhorn Costa. Context-aware, ontology-based service discovery. In *Ambient Intelligence*, pages 72–83. Springer, 2004. 47

[147] Michael Lutz. Ontology-based descriptions for semantic discovery and composition of geoprocessing services. *Geoinformatica*, 11(1):1–36, 2007. 47

[148] Chuanrong Zhang, Tian Zhao, and Weidong Li. The framework of a geospatial semantic web-based spatial decision support system for digital earth. *International Journal of Digital Earth*, 3(2):111–134, 2010. 47

[149] Yuqi Bai, Liping Di, and Yaxing Wei. A taxonomy of geospatial services for global service discovery and interoperability. *Computers & Geosciences*, 35(4):783–790, 2009. 47

[150] Hao Li, Yandong Wang, and Penggen Cheng. Semantic description for the taxonomy of the geospatial services. *Boletim de Ciências Geodésicas*, 21(3), 2015. 47

[151] Shawn Bowers and Bertram Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In *International Workshop on Data Integration in the Life Sciences*, pages 1–16. Springer, 2004. 47

[152] Peng Yue, Liping Di, Wenli Yang, Genong Yu, and Peisheng Zhao. Semantics-based automatic composition of geospatial web service chains. *Computers & Geosciences*, 33(5):649–665, 2007. 47

[153] Rob Lemmens, Andreas Wytzisk, R d By, Carlos Granell, Michael Gould, and Peter van Oosterom. Integrating semantic and syntactic descriptions to chain geographic services. *IEEE Internet Computing*, 10(5):42–52, 2006. 47

[154] Peng Yue, Liping Di, Wenli Yang, Genong Yu, Peisheng Zhao, and Jianya Gong. Semantic web services-based process planning for earth science applications. *International Journal of Geographical Information Science*, 23(9):1139–1163, 2009. 48

[155] Sérgio AB Cruz, Antonio MV Monteiro, and Rafael Santos. Automated geospatial web services composition based on geodata quality requirements. *Computers & Geosciences*, 47:60–74, 2012. 48, 130

[156] Peng Yue. *Semantic web-based intelligent geospatial web services*. Springer, 2013. 48

[157] Jochen Hinkel and Richard J.T. Klein. Integrating knowledge to assess coastal vulnerability to sea-level rise: The development of the DIVA tool. *Global Environmental Change*, 19(3):384–395, August 2009. 52

[158] Anthony Patt. Communicating uncertainty to policy makers. In *Uncertainties in Environmental Modelling and Consequences for Policy Making*, pages 231–251. Springer, 2009. 52, 135

[159] Tabea K. Lissner, Dominik E Reusser, Jacob Schewe, and Tobia Lakes. Linking human well-being and livelihoods with climate change impacts: contextualizing uncertainty in projections of water availability. *HESSD*, page inPrep, 2014. 52

[160] Markus Wrobel, Alexander Bisaro, Dominik Reusser, and JürgenP. Kropp. Novel approaches for web-based access to climate change adaptation information – mediation adaptation platform and ci:grasp-2. In Jiří Hřebíček, Gerald Schimak, Miroslav Kubásek, and AndreaE. Rizzoli, editors, *Environmental Software Systems. Fostering Information Sharing*, volume 413 of *IFIP Advances in Information and Communication Technology*, pages 489–499. Springer Berlin Heidelberg, 2013. 52

[161] Markus Wrobel and Dominik Reusser. Towards an Interactive Visual Understanding of Climate Change Findings on the Net: Promises and Challenges. In Birgit Schneider and Thomas Nocke, editors, *Image Politics of Climate Change*, pages p187–210. Transcript, London, 2014. 52

[162] Géza Tóth, Bartosz Kozlowski, Sylvia Prieler, and David Wiberg. GAEZ Data Portal - Users's Guide. Technical report, IIASA, FAO, Rome, Italy, 2012. 54, 57

[163] Prajal Pradhan, M.K.B. Lüdeke, Dominik E Reusser, and Jürgen P. Kropp. Food Self-Sufficiency across scales: How local can we go? *Environmental Science and Technology*, page under review, 2014. 55

[164] M. Boettle, D. Rybski, and J. P. Kropp. How changing sea level extremes and protection measures alter coastal flood damages. *Water Resour. Res.*, 49(3):1199–1210, 2013. 55

[165] A. Jarvis, H.I. Reuter, A. Nelson, and E. Guevara. Hole-filled srtm for the globe version 4, 2008. 56, 61

[166] DECRG. Gross domestic product 2010. http://preview.grid.unep.ch/index.php?preview=data&events=socec&evcat=1, 2010. [Online; last accessed 25-June-2015]. 56

[167] The World Bank; Center for International Earth Science Information Network (CIESIN), Columbia University; International Food Policy Research Institute (IFPRI) and Centro Internacional de Agricultura Tropical (CIAT). Gridded population of the world, version 3 (gpwv3): Population count grid. http://dx.doi.org/10.7927/H4639MPP, 2005. [Online; last accessed 25-June-2015]. 56

[168] UN. World urbanization prospects: The 2014 revision. Technical report, United Nations, Department of Economic and Social Affairs, Population Division, 2014. 56

[169] The World Bank; Center for International Earth Science Information Network (CIESIN), Columbia University; International Food Policy Research Institute (IFPRI) and Centro Internacional de Agricultura Tropical (CIAT). Global rural-urban mapping project, version 1 (grumpv1): Settlement points. http://sedac.ciesin.columbia.edu/data/set/grump-v1-settlement-points, 2011. [Online; last accessed 30-April-2014]. 56

[170] IIASA / FAO. Global agro-ecological zones (gaezv3.0). http://www.gaez.iiasa.ac.at/, 2012. [Online; last accessed 7-March-2014]. 56

[171] LP-DAAC. Land cover type yearly l3 global 500 m sin grid, mcd12q1 v051. https://lpdaac.usgs.gov/dataset_discovery/modis/modis_products_table/mcd12q1, 2012. [Online; last accessed 25-June-2015]. 56

[172] Lila Warszawski, Katja Frieler, Veronika Huber, Franziska Piontek, Olivia Serdeczny, and Jacob Schewe. The inter-sectoral impact model intercomparison project (isi–mip): Project framework. *Proceedings of the National Academy of Sciences*, 111(9):3228–3232, 2014. 55, 56, 68

[173] Vladimir Smakhtin, Carmen Revenga, and Petra Döll. A pilot global assessment of environmental water requirements and scarcity. *Water International*, 29(3):307–317, 2004. 55, 56, 68

[174] Charles J Vörösmarty, PB McIntyre, Mark O Gessner, David Dudgeon, Alexander Prusevich, Pamela Green, S Glidden, Stuart E Bunn, Caroline A Sullivan, C Reidy Liermann, et al. Global threats to human water security and river biodiversity. *Nature*, 467(7315):555–561, 2010. 55, 56, 68

[175] FAO. Fao aquastat database. http://www.fao.org/nr/water/aquastat/main/index.stm, 2004. [Online; last accessed 25-June-2015]. 55, 56, 68

[176] Alberte Bondeau, Pascalle C Smith, Sönke Zaehle, Sibyll Schaphoff, Wolfgang Lucht, Wolfgang Cramer, Dieter Gerten, Hermann Lotze-Campen, Christoph Müller, Markus Reichstein, et al. Modelling the role of agriculture for the 20th century global terrestrial carbon balance. *Global Change Biology*, 13(3):679–706, 2007. 55

[177] ICF. Measure dhs statcompiler. http://www.statcompiler.com/, 2005. [Online; last accessed 25-June-2015]. 55, 68

[178] FAO. Fao statistical database", howpublished = "http://faostat.fao.org/, 2004. [Online; last accessed 25-June-2015]. 55

[179] Tiziana Margaria. Service is in the Eyes of the Beholder. *IEEE Computer*, 2007. 57

[180] Steffen Kriewald. *srtmtools: SRTM tools*, 2013. R package version 2013-00.0.1. 58

[181] Robert J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2015. R package version 2.3-40. 58

[182] Roger Bivand, Tim Keitt, and Barry Rowlingson. *rgdal: Bindings for the Geospatial Data Abstraction Library*, 2015. R package version 0.9-3. 58

[183] Simon Urbanek. *png: Read and write PNG images*, 2013. R package version 0.1-7. 58

[184] Milan Kilibarda. A plotgooglemaps tutorial, 2013. 58

[185] Steffen Kriewald, Till Fluschnik, Dominik Reusser, and Diego Rybski. *cca: City Clustering Algorithm (CCA)*, 2015. R package version 0.98. 58

[186] Hernán D Rozenfeld, Diego Rybski, José S Andrade, Michael Batty, H Eugene Stanley, and Hernán a Makse. Laws of population growth. *Proceedings of the National Academy of Sciences of the United States of America*, 105(48):18702–7, December 2008. 58, 61

[187] Tiziana Margaria and Bernhard Steffen. Agile it: thinking in user-centric models. In *Leveraging Applications of Formal Methods, Verification and Validation*, pages 490–502. Springer, 2009. 60

[188] TK Lissner, CA Sullivan, DE Reusser, and JP Kropp. Determining regional limits and sectoral constraints for water use under climate change. *Hydrology and Earth System Sciences Discussions*, 11:4695–4727, 2014. 68

[189] Robert Hijmans, Julian Kapoor, John Wieczorek, Nel Garcia, Aileen Maunahan, Arnel Rala, and Alex Mandel. Global administrative areas (2012). gadm database of global administrative areas, version 2.0. `http://www.gadm.org/`, 2012. [Online; last accessed 25-June-2015]. 68

[190] Alexander Wickert and Anna-Lena Lamprecht. jabcstats: An extensible process library for the empirical analysis of jabc workflows. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, pages 449–463. Springer, 2014. 70

[191] Brahim Medjahed, Athman Bouguettaya, and Ahmed K Elmagarmid. Composing web services on the semantic web. *The VLDB Journal—The International Journal on Very Large Data Bases*, 12(4):333–351, 2003. 105

[192] Antti Valmari. The state explosion problem. In *Lectures on Petri nets I: Basic models*, pages 429–528. Springer, 1998. 110

[193] Anna-Lena Lamprecht. *User-Level Workflow Design: A Bioinformatics Perspective*, volume 8311. Springer, 2013. 110, 133

[194] Anna-Lena Lamprecht and Tiziana Margaria. Scientific workflows and xmdd. In *Process Design for Natural Scientists*, pages 1–13. Springer, 2014. 121

[195] Sven Lehmann. Konzept zur generierung einer web-basierten benutzerschnittstelle zur ausführung wissenschaftlicher workflows - ein generativer software-ansatz. Master's thesis, University of Potsdam, 2014. 122

[196] M FARNAGHI and A MANSOURIAN. Automatic composition of geospatial web services in order to generate landslide susceptibility map, 2015. 130

[197] Peng Yue. Automatic composition of geospatial web service. In *Semantic Web-based Intelligent Geospatial Web Services*, pages 65–85. Springer, 2013. 130

[198] Bernhard Steffen, Tiziana Margaria, Andreas Claßen, and Volker Braun. Incremental formalization: a key to industrial success, 1995. 131

[199] Jochen Hinkel, Detlef P. Vuuren, Robert J. Nicholls, and Richard J. T. Klein. The effects of adaptation and mitigation on coastal flood impacts during the 21st century. An application of the DIVA and IMAGE models. *Climatic Change*, 117(4):783–794, August 2012. 136

[200] Charles Petrie, Tiziana Margaria, Holger Lausen, and Michal Zaremba, editors. *Semantic Web Services Challenge. Results from the First Year*. Springer US, 2009. 138