Semantic interoperability of distributed geo-services

Semantic interoperability of distributed geo-services

Rob Lemmens

Publications on Geodesy 63

NCG Nederlandse Commissie voor Geodesie Netherlands Geodetic Commission

Delft, September 2006

Semantic interoperability of distributed geo-services
Rob Lemmens
Publications on Geodesy 63
ISBN-10: 90 6132 298 7
ISBN-13: 978 90 6132 298 6
ISSN 0165 1706

Published by: NCG, Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission, Delft, The Netherlands.
Printed by: Optima Grafische Communicatie, Optima Graphic Communication, Rotterdam, The Netherlands.
Cover illustration: Rob Lemmens

NCG, Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission
P.O. Box 5058, 2600 GB Delft, The Netherlands
T: +31 (0)15 278 28 19
F: +31 (0)15 278 17 75
E: info@ncg.knaw.nl
W: www.ncg.knaw.nl

The NCG, Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission is part of the Royal Netherlands Academy of Arts and Sciences (KNAW)

Abstract

The last two decades have shown a major shift from stand-alone software systems to networked ones. As with all information system domains, Geographic Information Systems (GISs) have been influenced to a large extent by recent internet developments, resulting in an increasing availability of client/server applications using distributed geo-(web-)services, such as interactive maps, route planners and gazetteers. There is an increasing need for organisations to perform on demand geo-processing tasks by integrating and reusing geo-information and geo-services from within and outside the organisation. These activities are typically performed in the context of so called Geo-information Infrastructures (GIIs).

The process of integrating services is commonly referred to as service chaining. This requires that services can be easily found, and that they are executable and interoperable. Interoperability means that the services 'understand' each other's messages. A major impediment is formed by the semantic heterogeneity (the differences in meaning) of geo-information and of the functionality of geo-services. Making services semantically interoperable is an important prerequisite for information sharing in today's networked society. This involves services that rely on different knowledge domains, one of which is the geo-information domain.

Within this context, the research presented in this thesis provides solutions for the computer-aided integration of distributed heterogeneous geo-information and geo-services, based on their semantics (the meaning of their content).

Geo-information distinguishes from other information by its spatial relevance. Geo-services often have to deal integrally with multiple-representations of features in a spatial, temporal and thematic dimension. Geo-services are also *implicitly* connected by the geographic location of the features they process. This has implications for the interoperability of geo-services. For example, the validity of a service (e.g., a routeplanner) may be bound to a specific geographic area, which could imply it cannot be used in combination with services involving another validity area. On the contrary, services that seem to be incompatible due to differences in feature representation (e.g., geometry, coordinate reference system), may turn out to be useful in combination, because they contain information on the same locations.

On demand geo-processing requires services and the meta-information that describes the services to be available at the time a task is being executed. Moreover, the service descriptions should be based on commonly agreed rules for service characterisation. Inter-service contracts that contain such rules may result in service interoperability and this can be achieved at three levels: syntax, structure and semantics. The influential specifications of the Open Geospatial Consortium (OGC) and the ISO 19100 series of standards, implement formal contracts on the syntactical and structural level, but they prescribe only informal contracting at the semantic level. Despite their rigid conceptualisation, they lack a machine-accessible formalisation that supports the specification of semantics for geo-information and geo-services. This research has developed such a formalisation, which is specified in a so called *semantic interoperability framework*. In this framework a key role is played by machine ontologies, which are machine-accessible representations of knowledge that are used for inferring intra- and inter-resource relationships. Recent research efforts in the field of the Semantic Web have contributed considerably to the deployment of ontology-based applications by providing a theoretical foundation (Description Logics), ontology languages (e.g., the Web Ontology Language (OWL)), and tools for ontology creation, access and reasoning with web-based (machine) ontologies. The power of web-based ontologies lies in their interoperable (XML based) representation, the use of unique namespaces and the fact that they allow for automated reasoning.

The semantic interoperability framework developed in this research, contains (1) geo-information modelling ontologies which are based on the ISO General Feature Model, (2) domain specific ontologies (amongst others, one which is based on a data model used by the Dutch Topographic Service), and (3) a geo-operation modelling ontology. The latter is based on a geo-operation taxonomy, an input/output parameter characterisation and a workflow model. The taxonomy and parameter characterisation have been developed as part of this research, the workflow model is based on OWL-S, an OWL-based upper ontology for web services.

Ontology-based service descriptions have been created in the context of four use cases in the following areas: (1) information model integration for risk mapping, (2) ad hoc data integration in a disaster emergency situation, (3) reuse of geodata and geo-services in scientific research, and (4) ad hoc integration of travel services. The ontology-based descriptions are used as representations of service requests and advertisements in a matchmaking process. The matchmaking is performed by an ontology reasoner which can infer implicit relationships that exist in a knowledge base containing service descriptions as sets of concepts. The reasoner is implemented together with the ontologies in a prototype environment. Except for the reasoner, this has been carried out with open source software. Within this environment, basic matchmaking has been successfully performed to support data set integration and service chaining. This has been demonstrated by tests implementing the aforementioned use cases.

The offered solution is flexible and extensible. With respect to flexibility, the research demonstrates the use of incomplete service descriptions. With respect to extensibility, the research shows how service descriptions can be extended with new concepts. It is also demonstrated how existing application domains can be linked

through ontology mappings. In the process of service chaining, four steps have been identified, i.e., discovery, abstract composition, concrete composition and execution. The link between the abstract and concrete composition of services is realised by annotation, which connects ontology elements with parameters of executable code. For one of the use cases, this code has been deployed in a prototype software application (the latter being part of an external research effort).

There are also limitations to the approach followed, which are partly due to the limitations of OWL and reasoning with it, i.e., with respect to spatial reasoning and the use of metaclasses. In addition, the current prototype environment has several shortcomings: (1) constraints of the user-interfaces (entering service descriptions in Description Logics is still rather complex), (2) the inflexibility of the reasoning implementation and (3) the incompleteness of mappings between domain ontologies, all of which are thought to be surmountable.

A number of recommendations are made for the improvement of the current design and implementation of the interoperability framework, such as the incorporation of: meta-information propagation, concept similarity quantifiers and result ranking in the matchmaking process. The deployment of the approach requires key organisations such as OGC to develop and maintain domain independent parts of a semantic interoperability framework and organisations with a GII mandate to manage its domain dependent parts.

Application fields that are thought to benefit from the presented approach in the short term are, amongst others: service discovery and chaining in GII, harmonisation of geo-information models, multiple-representation of geo-information, profile matching of geo-service users, documentation of geo-processing history (lineage), and quality assessment of meta-information. The target groups of this research are firstly geo-information engineers who are confronted with information integration issues and service interoperability issues, and secondly, information engineers in general confronted with distributed information and with end users that need to access distributed services as one virtual application.

Samenvatting

De laatste twee decennia hebben een verschuiving laten zien van autonome softwaresystemen naar genetwerkte systemen. Zoals alle domeinen in de informatietechnologie zijn Geografische Informatie Systemen (GIS) sterk beïnvloed door de recente ontwikkelingen van het internet. Deze hebben geresulteerd in een toenemende beschikbaarheid van client/server-applicaties die gebruik maken van gedistribueerde geo-(web-)services, zoals interactieve kaarten, routeplanners en gazetteers. Er is een groeiende behoefte binnen organisaties aan het verwerken van geo-informatie "op verzoek" door middel van het integreren en hergebruiken van geo-informatie en geo-services zowel van binnen als van buiten de organisatie. Deze activiteiten worden doorgaans uitgevoerd in de context van geo-informatie infrastructuren (GII).

Het proces van integreren van services is ook wel bekend als *service chaining*. Dit vereist dat services vindbaar, uitvoerbaar en interoperabel zijn. Interoperabiliteit betekent dat services elkaars berichten 'begrijpen'. Een grote hindernis wordt gevormd door de semantische heterogeniteit (verschillen in de betekenis) van geo-informatie en van de functionaliteit van geo-services. Het creëren van semantisch interoperabele services is een belangrijke voorwaarde voor het uitwisselen van informatie in de huidige genetwerkte maatschappij. Dit betreft services die betrekking kunnen hebben op verschillende kennisdomeinen, waarvan het geo-informatie domein er één is.

Binnen deze context draagt het onderzoek, beschreven in deze dissertatie, oplossingen aan voor de computer-ondersteunde integratie van gedistribueerde heterogene geo-informatie en geo-services, gebaseerd op hun semantiek (betekenis van hun inhoud).

Geo-informatie onderscheidt zich van andere informatie doordat het een ruimtelijke relevantie heeft. Geo-services worden gekenmerkt door hun vaak integrale verwerking van meervoudige representaties van geo-objecten (representaties van geografische fenomenen) in een ruimtelijke, temporele en thematische dimensie. Geo-services zijn ook onderling *impliciet* verbonden door de geografische locaties van de geo-objecten die ze verwerken. Dit heeft implicaties voor de interoperabiliteit van geo-services. Bijvoorbeeld, de geldigheid van een service (bijv. een routeplanner) kan verbonden zijn aan een specifiek geografisch gebied, hetgeen kan inhouden dat deze service niet kan worden gebruikt in combinatie met andere

services met een afwijkend geldigheidsgebied. Andersom, services die incompatibel lijken door hun verschil in geo-object representatie (bijv. geometrie, coördinaat-referentiesysteem) kunnen in combinatie bruikbaar blijken te zijn, omdat ze informatie bevatten over dezelfde locaties.

Geo-informatie verwerking op verzoek vereist dat services en meta-informatie die deze services beschrijft, beschikbaar zijn op het moment dat een taak moet worden uitgevoerd. Bovendien moeten de service-beschrijvingen gebaseerd zijn op overeengekomen regels voor de karakterisering van services. Inter-service contracten die zulke regels bevatten, kunnen bijdragen aan service-interoperabiliteit op drie niveaus, namelijk: syntax, structuur en semantiek. De invloedrijke specificaties van het Open Geospatial Consortium (OGC) en de ISO 19110 standaarden implementeren formele contracten op het syntactisch en structurele niveau, maar op het semantisch niveau schrijven zij alleen informele contractering voor. Ondanks hun rigide conceptualisering missen ze een machine-toegankelijke formalisatie die ondersteuning zou kunnen bieden voor de specificatie van semantiek voor geo-informatie en geo-services. In dit onderzoek is zo'n formalisatie ontwikkeld. Deze is gespecificeerd in een *raamwerk voor semantische interoperabiliteit*. In dit raamwerk is een sleutelrol weggelegd voor machine-ontologieën. Dit zijn machine-toegankelijke representaties van kennis, die kunnen worden gebruikt voor de afleiding van relaties tussen informatie-elementen. Recent onderzoek op het gebied van het Semantic Web heeft bijgedragen tot de ontwikkeling van ontologie-gebaseerde applicaties door het beschikbaar maken van een theoretische basis (Description Logics), ontologietalen (bijv. de Web Ontology Language (OWL)), en gereedschappen voor ontologiebouw, -toegang en redeneren met web-gebaseerde (machine) ontologieën. De kracht van web-gebaseerde ontologieën ligt in de interoperabele (XML-gebaseerde) representatie, het gebruik van unieke *namespaces* en het feit dat ze automatisch redeneren ondersteunen. Het raamwerk voor semantische interoperabiliteit, dat is ontwikkeld in dit onderzoek, bestaat uit (1) geo-informatie modelleringsontologieën die zijn gebaseerd op het ISO *General Feature Model*, (2) domein-specifieke ontologieën (o.a. één gebaseerd op het datamodel dat wordt gehanteerd door de Nederlandse Topografische Dienst Kadaster), en (3) een geo-operatie modelleringsontologie. De laatste is gebaseerd op een geo-operatie taxonomie, een karakterisering van de invoer/uitvoer parameters en een workflow-model. De taxonomie en parameterkarakterisering zijn ontwikkeld als onderdeel van dit onderzoek, het workflow-model is gebaseerd op OWL-S, een OWL-gebaseerde generieke ontologie voor web services.

De op ontologie gebaseerde service-beschrijvingen zijn gecreëerd in de context van vier gebruikersscenario's in de volgende toepassingen: (1) informatiemodel-integratie for risicokartering, (2) ad hoc gegevensintegratie in een rampsituatie, (3) hergebruik van geodata en geo-services in wetenschappelijk onderzoek, en (4) ad hoc integratie van services in een reisscenario. De service-beschrijvingen worden gebruikt als representaties van service-aanvragen en aanbiedingen in een *matchmaking* proces. Deze matchmaking wordt uitgevoerd door een ontologie-redeneer mechanisme (Eng: *reasoner*), dat impliciete relaties kan afleiden uit een kennisbank waarin services zijn beschreven als concepten. De reasoner is geïmplementeerd samen met de bovengenoemde ontologieën in een prototype omgeving. Met uitzondering van de reasoner, is de implementatie uitgevoerd met open source software. In deze omgeving zijn op succesvolle wijze basale vormen van matchmaking uitgevoerd voor de ondersteuning van gegevensintegratie en service chaining. Dit is gedemonstreerd aan de hand van tests, die implementaties vormen van de bovengenoemde gebruikerscenario's.

De aangereikte oplossing is flexibel en uitbreidbaar. Wat betreft flexibiliteit wordt dit gedemonstreerd aan de hand van het gebruik van incomplete servicebeschrijvingen. Wat betreft uitbreidbaarheid laat het onderzoek zien hoe servicebeschrijvingen kunnen worden uitgebreid met nieuwe concepten. Tevens wordt getoond hoe applicatiedomeinen kunnen worden gekoppeld middels *ontology mappings*. In het service chaining proces worden vier stappen onderscheiden, namelijk: het vinden (Eng: *discovery*), abstracte compositie, concrete compositie en uitvoering (Eng: *execution*). Het verband tussen de abstracte en concrete compositie van services wordt gerealiseerd door middel van *annotatie*, die een verbinding maakt tussen ontologie-elementen en parameters in de softwarecode. In één van de gebruikerscenario's is deze softwarecode daadwerkelijk gebruikt in een computertoepassing (de laatste maakt deel uit van een extern onderzoek).

Er zijn tevens beperkingen in de gevolgde benadering. Deze zijn gedeeltelijk te wijten aan de beperkingen van OWL en het redeneren met haar elementen, bijvoorbeeld met betrekking tot het ruimtelijk redeneren en het gebruik van metaklassen. Bovendien heeft de huidige prototype-omgeving de volgende beperkingen: (1) m.b.t. de gebruikerinterface (de invoer van service-beschrijvingen in Description Logics is nogal gecompliceerd), (2) de inflexibiliteit van de redeneringsimplementatie, en (3) de incomplete set van mappings tussen de domein-ontologieën. Desalniettemin worden deze prototypebeperkingen als tijdelijk gezien en niet als principieel.

Een aantal aanbevelingen wordt gegeven voor het verbeteren van het huidige ontwerp en de implementatie van het interoperabiliteitsraamwerk, zoals het meenemen van: meta-informatie propagatie, de rangschikking van resultaten in het matchmaking proces, en het kwantificeren van concept-gelijksoortigheid. Het operationeel inzetten van het raamwerk vereist dat sleutelorganisaties, zoals OGC, de domein-onafhankelijke delen van het raamwerk zullen ontwikkelen, onderhouden en beschikbaar maken, en dat organisaties met een GII-mandaat de domeinafhankelijke onderdelen beheren.

Toepassingsgebieden die op korte termijn worden geacht te profiteren van de gepresenteerde aanpak zijn o.a. service chaining in GII, harmonisatie van geo-informatie modellen, meervoudige representaties van geo-informatie, profielmatching van geo-service gebruikers, documentatie van de proceshistorie van geo-informatie (Eng: *lineage*), en kwaliteitsbeoordeling van meta-informatie. De doelgroepen van dit onderzoek zijn als eerste de groep van geo-informatie-technici, die worden geconfronteerd met problemen van informatie-integratie en interoperabiliteitsproblemen met services, en als tweede, informatie-technici, die in het algemeen geconfronteerd worden met gedistribueerde informatie en met eindgebruikers die gedistribueerde services moeten kunnen benaderen als zijnde één virtuele applicatie.

Acknowledgements

The long period of research work that has resulted in this thesis has been valuable and enjoyable. There are many people that have contributed with their professional advise, inspiring discussions, encouragement and friendship.

First, I would like to thank Prof. Wolfgang Kainz for giving me the opportunity to start the research and the freedom to work during the first years of my research.

I am very grateful to Prof. Peter van Oosterom, who took up the promotorship in 2003. Peter gave me much insight in both theoretical and practical issues and gave me ample support by reviewing the thesis carefully. Peter is a strong advocate of the use of formal semantics in GIS and this strengthened many of my ideas. He was able to motivate me during the tough periods of thesis writing.

I was also privileged to have Rolf de By as my supervisor. His professionalism has helped me in striving for scientific quality and consistency in the thesis. He also jumpstarted my Latex word processing, which paid off in the end. Rolf often helped out with several other issues as a great colleague.

During the research work I often teamed up with Marian de Vries, my 'PhD-mate' in Delft. We had long discussions on semantics and OpenGIS, which were very useful for me to put my topic into perspective.

I would like to express my sincere thanks to my colleagues at the ITC Department of Geo-information Processing (GIP). They have often supported my work by releasing me from other duties. Teaching the Internet GIS course modules together with Barend Köbben was (and is) very enjoyable. His enthusiasm and flexibility has helped me a lot during the teaching periods. My thanks goes to ITC in general and specifically to Prof. Martien Molenaar as rector, Prof. Menno-Jan Kraak as head of the GIP department and Prof. Martin Hale as head of Research who enabled me to continue my research work over the years. To Andreas Wytzisk for sharing his OpenGIS expertise. To Loes Colenbrander for providing logistical support. To Marga Koelen, Carla Gerritsen and Petry Maas-Prijs for lending me many books for extensive periods. To Ard Blenke for providing the necessary computer support.

Wim Koolhoven has assisted me with implementing parts of the GeoMatch-Maker prototype. His programming skills are much appreciated. Wim Feringa assisted with polishing some of the figures in the thesis. The cover page was evaluated by my colleagues with cartographic scrutiny during one of the coffee breaks.

Contributions have also been made by the students that I supervised during their MSc thesis work: Julian Gomez, Wim Bakker, Trias Aditya, Helbert Arenas, Lydia Marleny Prieto, Gustavo Zarrate, and Aster Denekew Yilma. Our work was often of mutual interest. Many other people have also contributed indirectly through the discussions I had with them over the years: Heiner Stuckenschmidt, Michael Lutz, Florian Probst, Eva Klien, Carlos Granell. The cooperation with Carlos during his stay at ITC in 2005 was really fruitful.

Jantien Stoter, Connie Blok, Corné van Elzakker, Arbind Tuladhar, Javier Morales and Arta Dilo have shared their PhD experiences with me, which was especially valuable during the last period of logistical arrangements. Thanks also to Yuxian Sun and Martin Salzmann for their moral support.

RacerPro Systems, Volker Haarslev and Ralf Möller are kindly acknowledged for making available their RacerPro reasoner software for this research. In addition, several others provided data support and professional advise: Topografische Dienst Kadaster, Provincie Overijssel and Wilko Quak.

I would like to thank especially Jantien Stoter, Marian de Vries, John Horn, Dick Beekman, Sanjeev Sarpal and Theodor Förster for reading, correcting and commenting on parts of the manuscript.

In the end, much of the academic work was not conducted between 9 and 5 and has sacrificed parts of social life. Fortunately, there were always people around that gave me inspiration and friendship.

I wish to express my love and affection to my family and specifically my parents who have given a lifetime support to my academic career.

Finally, Bernadette, you have given me more support than I could have wished for. Your understanding, encouragement and love have made this result possible. Even in a practical sense you have given me great help. During tough times in the study you cared about Nikki and me. And Nikki, your smiles, questions, humor and 'Papaaaa' are unrivalled in putting life into perspective. I am so happy to be with both of you.

Contents

Abstract i

Samenvatting v

Acknowledgements ix

1 Why interoperability is important 1
- 1.1 Research context and motivation . . . 2
- 1.2 Research objectives . . . 7
- 1.3 Research approach . . . 10
- 1.4 Related work . . . 11
- 1.5 Thesis outline . . . 16

2 Interoperable distributed services 19
- 2.1 Distributed processing paradigms . . . 20
  - 2.1.1 Services as units of data processing . . . 20
  - 2.1.2 Classification of services . . . 25
  - 2.1.3 Tightly- and loosely-coupled systems . . . 28
- 2.2 Interoperability and heterogeneity . . . 29
- 2.3 Overcoming heterogeneity by contract . . . 31
- 2.4 Interoperability models . . . 34
  - 2.4.1 Distributed computing platforms . . . 34
  - 2.4.2 ISO RM-ODP . . . 35
  - 2.4.3 Web services . . . 35
  - 2.4.4 Semantic Web . . . 36
  - 2.4.5 Spatial data interchange standards . . . 37
  - 2.4.6 Open Geospatial Consortium (OGC) specifications . . . 38
  - 2.4.7 ISO 19100 standards . . . 40
- 2.5 Geo-services . . . 41
- 2.6 Geo-service use cases . . . 43
  - 2.6.1 Riskmap NL . . . 44
  - 2.6.2 Emergency 112 . . . 45

2.6.3 Research Net . . . 46
2.6.4 Travel Google . . . 47
2.7 Summary and reflection . . . 47

3 Service models for discovery, composition and execution 49
3.1 Information modelling . . . 49
3.1.1 A layered approach . . . 50
3.1.2 Abstraction hierarchies . . . 53
3.1.3 Information abstraction . . . 54
3.2 Process modelling . . . 55
3.2.1 Modelling dynamic systems . . . 57
3.2.2 Formalised process models and languages . . . 58
3.2.3 Process modelling in GIS . . . 61
3.3 Service chaining . . . 62
3.3.1 User context . . . 62
3.3.2 Modes of discovery, composition and execution . . . 64
3.4 Summary and reflection . . . 66

4 Semantic modelling 69
4.1 What is an ontology? . . . 70
4.2 Foundations for machine ontology . . . 74
4.2.1 Description Logics . . . 74
4.2.2 Resource Description Framework (RDF) . . . 78
4.2.3 Web Ontology Language (OWL) . . . 79
4.3 Ontology design and creation . . . 82
4.4 Ontology representations and notation . . . 84
4.5 Reasoning with a knowledge base . . . 85
4.5.1 TBox and ABox reasoning . . . 86
4.5.2 Application of reasoning: matchmaking . . . 89
4.6 Semantic interoperability frameworks . . . 90
4.6.1 Ontology integration . . . 91
4.6.2 Ontology classification . . . 92
4.6.3 Semantic annotation and markup . . . 93
4.6.4 Ontology meta-information . . . 94
4.7 Semantic web services . . . 94
4.7.1 OWL-S . . . 95
4.7.2 Semantic annotation of web services: WSDL-S . . . 98
4.8 Geo-semantic modelling and spatial relevance . . . 99
4.9 Summary and reflection . . . 100

5 Semantic interoperability framework for geo-services 101
5.1 Semantic framework overview 101
5.1.1 Information and operation model 102
5.1.2 Framework ontologies 105
5.1.3 Relations with ISO specifications 106
5.2 Feature symbol ontology 107
5.2.1 Feature overview 108
5.2.2 Feature geometry 110
5.2.3 Thematic attributes 111
5.2.4 Coverages 112
5.2.5 Example usages of feature symbols in feature concepts . . . 112
5.3 Feature concept ontologies 114
5.3.1 NEN3610 114
5.3.2 TOP10NL 117
5.3.3 Riskmap 119
5.3.4 Travel 121
5.4 Geo-operation characterisations — OPERA 122
5.4.1 Atomic geo-operations 122
5.4.2 Ontology design — OPERA-R and OPERA-D 123
5.5 OPERA-R — Feature processing operations 125
5.5.1 Atomicity 125
5.5.2 Classification of operations 126
5.5.3 Operation type descriptions 127
5.6 Geo-service descriptions 130
5.6.1 Semantic descriptions based on OWL 131
5.6.2 Creating hybrid descriptions: grounding 135
5.7 Summary and reflection 136

6 Geo-information matching and service chaining 139
6.1 Example: Riskmap chain 139
6.2 Semantic modelling of geo-service chains 140
6.3 Derived operations and ontology mappings 144
6.3.1 ISO 19119 mapping 145
6.3.2 Multiple ontology mapping 146
6.3.3 Definitions of actual operation implementations 148
6.4 Matchmaking 148
6.4.1 The service descriptor, a shorthand notation 149
6.4.2 Matchmaking modes 150
6.4.3 TBox reasoning 153
6.4.4 ABox reasoning 154
6.4.5 Reasoning with control flow 155
6.4.6 Exploitation of ontology mappings 158
6.5 Summary and reflection 159

7 Use case implementations **161**
7.1 Riskmap NL 161
7.1.1 Extension of the Riskmap chain 161
7.1.2 Exploitation of ontology mappings 163
7.2 Emergency 112 164
7.2.1 Annotation 165
7.2.2 Information retrieval 167
7.3 Research Net 169
7.3.1 Phase 1: Service provision 169
7.3.2 Phase 2: Service consumption 173
7.4 Travel Google 173
7.4.1 Ontology inspection 175
7.4.2 Creation of service request 176
7.4.3 Exploitation of ontology mappings 177
7.4.4 Service description analysis and service selection 178
7.5 Summary and reflection 181

8 Implementation of prototypes: OnToGeo and GeoMatchMaker **183**
8.1 Practical design and implementation issues of OnToGeo 184
8.1.1 Implementation of OWL-S 184
8.1.2 Distributed ontologies and namespaces 186
8.2 Workbench tools 187
8.2.1 Protégé core and OWL plug-in 187
8.2.2 Ontoviz plug-in 188
8.2.3 OWL-S editor plug-in 188
8.2.4 OWLDoc plug-in 189
8.2.5 Reasoning through DIG interface 189
8.2.6 RacerPro 191
8.2.7 JRacer 192
8.2.8 RICE 192
8.2.9 AutoFocus 193
8.3 GeoMatchMaker, an integrated prototype 193
8.3.1 Functional parts 193
8.3.2 Compilation of Java code 197
8.3.3 Usage 197
8.3.4 Exploitation and embedding 200
8.3.5 Limitations 200
8.4 Creating service descriptions 201
8.4.1 Individual-based descriptions 201
8.4.2 Class-based descriptions 201
8.5 Summary and reflection 202

9 Conclusions and recommendations 205
9.1 Summary and reflection 205
9.2 Conclusions 207
9.2.1 Main conclusion 207
9.2.2 Answers to research questions 207
9.2.3 Specific issues 212
9.3 Main contributions 216
9.4 Deployment 218
9.5 Recommendations for further work 220

A UML notation 223

B ISO 19100 overview 225

C OPERA-R geo-operation types 227
C.1 Human interaction operations 227
C.2 Feature modelling 228
C.3 Feature access 228
C.4 Feature processing 229
C.4.1 Feature instantiation 229
C.4.2 Across attribute types 229
C.4.3 Among thematic attributes 230
C.4.4 Feature selection 231
C.4.5 Get feature property values 232
C.4.6 Geometric object change 232
C.4.7 Topology 233
C.4.8 Metric measurement 236
C.4.9 Overlay 236
C.4.10 Distance-based 238
C.4.11 Neighbourhood 239
C.4.12 Connectivity 240
C.4.13 Interpolation 240
C.4.14 Geometric transformation 241
C.4.15 Temporal 242
C.5 Feature presentation manipulation 242
C.6 Service creation and management 242
C.7 Service execution 242
C.8 MetaInfo creation and storage management 243
C.9 MetaInfo processing 243
C.10 MetaInfo presentation manipulation 243

D OPERA-R
I/O parameters for feature processing operations 245

E ADL Gazetteer OWL service description **251**

F ADL Gazetteer WSDL service description **253**

G ISO 19119 mapping **255**

Bibliography **263**

Curriculum Vitae **291**

ITC Dissertations **293**

Chapter 1

Why interoperability is important

The thesis that lies in front of you describes a research effort that lies at the crossroads of Geographic Information Systems (GISs) and the Semantic Web.

GIS GISs are specialised in the handling of georeferenced data (data containing features with an earth-related location). The fact that the term GIS is increasingly used in conjunction with Geo Information *Services*, demarcates the change of software applications and more importantly, the way of thinking about geospatial data handling, i.e., a paradigm shift is taking place from the data viewpoint to the functional viewpoint. As with all information system domains, GIS has been recently influenced to a large extent by internet developments, resulting in an increasing availability of client/server applications using distributed geo-web-services. Web services are software systems that provide specific functionality to a group of clients over a computer network. New challenges lie ahead in order to integrate these services into meaningful *service chains* (e.g., by using a shop locator of provider X together with a route planner of provider Y) and make them instantly available to the clients.

Semantic Web The Semantic Web can be seen as the next step in the evolution of the World Wide Web as originally envisaged by its founder, Tim Berners-Lee. In the Semantic Web, documents and services are provided with well-defined meaning (semantics), so that computers can more sophistically process and integrate the content of these documents and services, in order to improve the access to information. In the Semantic Web, the semantics of information are laid down in formal descriptions (meta-information), based on machine-processable knowledge structures (ontologies). The Semantic Web draws upon theories developed in several existing disciplines such as mathematical logic, computer science and knowledge representation. With the advent of new web-oriented languages based on the Extensible Markup Language (XML), the Semantic Web is slowly making its way into current information systems. Its predominant application fields are found in medical and multimedia information and service integration.

Regarding the limitations of the Semantic Web, Berners-Lee said the following in 1998 [25]:

> "A Semantic Web is not Artificial Intelligence. The concept of machine-understandable documents does not imply some magical artificial intelligence which allows machines to comprehend human mumblings. It only indicates a machine's ability to solve a well-defined problem by performing well-defined operations on existing well-defined data. Instead of asking machines to understand people's language, it involves asking people to make the extra effort."

The potential synergy between GIS and the Semantic Web forms an important basis of this research. The motivation for the work reported on in this thesis is provided in Section 1.1. The research objectives and research approach are presented in respectively Sections 1.2 and Section 1.3. An overview on related work is provided in Section 1.4. The chapter concludes with an outline of the thesis in Section 1.5.

1.1 Research context and motivation

Lost in the information clouds

The last two decades have shown an observable shift from stand-alone software systems to distributed systems. This has been triggered by societal and business incentives for collaboration as well as a great technology push in developments of hardware, software and communication networks. The internet and the World Wide Web have been important contributors to this process, providing a scalable architecture with approachable technology for businesses and the general public. As a result, the wide range of communication opportunities has led to an information push, that produces countless electronic documents and on-line services. Regarding the latter, geo-information technology is making its contribution with services such as interactive maps, route-planners and gazetteers. Despite the fact that this offers added value to our information society, many valuable information sources are lost in the information clouds, because (1) their usage/functionality is not properly made public and (2) they are only connected through low-level hyperlinks and (3) their usage/functionality is still left as stand-alone. As a consequence, documents and services are often (if at all) found in isolation and the combination of their content is left to the end user. This leads to redundant sources of data and makes information handling inefficient, especially when it comes to non-routine tasks. Such tasks are typically ones that do not have a known predefined solution and require *on demand information processing*, which involves ad hoc application building. In service-oriented environments, this can be achieved by *service chaining*, which entails the discovery, composition and integral execution of (potentially distributed) services to support a specific task.

Interoperability for all

Organisations have only recently started to work on more advanced forms of integrated information provision to support specific tasks on demand. Efforts in the field of integrated geo-information provision are being run in initiatives to build geographic information infrastructures (GIIs). This is done at local, regional level and international levels [48, 70, 102, 286]. An example of a recently started international GII is the European Union's INSPIRE (INfrastructure for SPatial InfoRmation in Europe) project. Its main objective and principles have been described as follows [69]:

> "INSPIRE aims at making available relevant, harmonised and quality geographic information for the purpose of formulation, implementation, monitoring and evaluation of Community policy-making.
>
> INSPIRE principles
>
> - Data should be collected once and maintained at the level where this can be done most effectively.
> - It should be possible to combine seamlessly spatial data from different sources and share it between many users and applications.
> - Spatial data should be collected at one level of government and shared between all levels.
> - Spatial data needed for good governance should be available on conditions that are not restricting its extensive use.
> - It should be easy to discover which spatial data is available, to evaluate its fitness for purpose and to know which conditions apply for its use."

In general, a GII provides the bridges between information providers and consumers. Apart from the political, legal, organisational and financial aspects that are involved, a key issue in this development is the advancement of *interoperability*. Interoperability plays a role for hardware as well as for software. The problem of interoperability and its possible solutions in the domain of GIS are illustrated with the help of a travel power plug metaphor. Figure 1.1 shows three hardware solutions to the problem a traveller faces when s/he wants to use electricity in a country with another electricity supply system. Solution (a) is the most basic; it adapts one socket shape to the other. Solution (b) provides multiple socket shapes

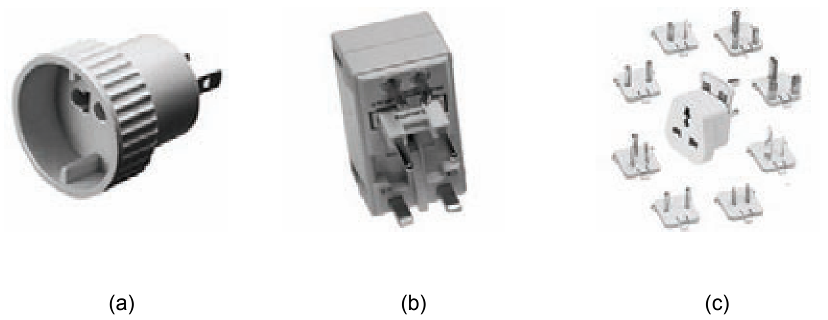
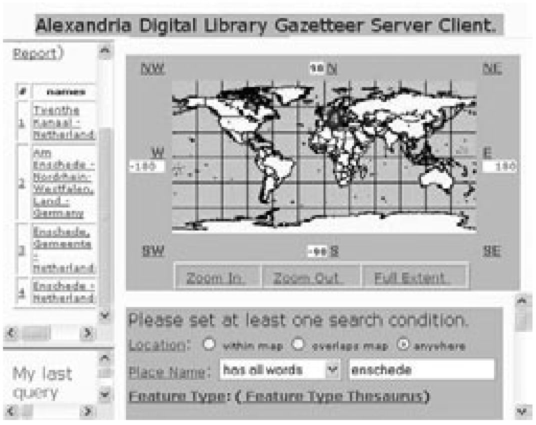


Figure 1.1: Three different solutions to a well-known interoperability problem.

(and is commonly called a 'universal adapter'). Although solution (c) is also 'universal', it is the most flexible of the three. It makes use of an intermediate socket shape in order to accommodate specific end-pieces; any odd electricity supply system just requires a different end-piece. In the context of information systems (in general and GIS specifically), solution (a) is earmarked as a bi-lateral contract and solution (b) as a multi-lateral contract. Solution (c) is representative of the ideas of the interoperability program of the Open Geospatial Consortium (OGC), which actually addresses the 'front-piece' of the contract as well (in contrast to many travel plugs, which have no universal socket shape at the front). The members of OGC are building up a set of interface specifications for generic geodata access and geo-services. OGC specifications can be seen as the intermediate socket shape of travel plug (c); the proprietary software products are the end-pieces which can remain independent with their own implementations. OGC's specifications build on top of ICT-mainstream standards, such as the Unified Modelling Language (UML) and the Extensible Markup Language (XML). The aforementioned approach has gained momentum over the last couple of years since important specifications, such as OGC's Web Map Service (WMS) and Web Feature Service (WFS) have proved to offer a practical solution to heterogeneity problems in the increasing demand for geo-information integration.

OGC and beyond

But the interoperability efforts do not end here. The attentive reader has noted that the travel plugs of Figure 1.1 do not solve all electricity problems of the traveller. Countries may differ in the electrical voltage they use and this is not apparent from the socket shape. So it is with geographic information systems. Users of geo-information need to know the content of a Web Map Service before they can do meaningful things with it. Here, metadata standards help out. They provide schemas with data fields that should be filled by the service provider with,


Alexandria Digital Library Gazetteer Server Client.
Report)
# names
1 Twenthe Kanaal - Netherland
2 Am Enschede - Nordrhein-Westfalen, Land - Germany
3 Enschede, Gemeente - Netherland
4 Enschede - Netherland
My last query
NW
10 N
NE
W
-180
E
180
SW
-90 S
SE
Zoom In
Zoom Out
Full Extent
Please set at least one search condition.
Location: ○ within map ○ overlaps map ⦿ anywhere
Place Name: has all words enschede
Feature Type: (Feature Type Thesaurus)

Meta-information

Scope and purpose: A gazetteer associates geographic names (place names and feature names) with geographic locations and other descriptive information. The Alexandria Digital Library (ADL) Gazetteer combines the U.S. place names from the U.S. Geological Survey's GNIS database, the non-U.S. place names from the National Imagery and Mapping Agency's GNS database, and other gazetteer data sets.

Figure 1.2: The Alexandria Digital Library gazetteer web service. Left: output in a web browser. Right: Part of the meta-information, describing the service; its content representation is meant for humans, not for machines.

among others, the geodata acquisition method, its coordinate reference system, displayed feature types (e.g. 'parcels'), etc. And this is where it often ends. The meaning of 'parcels' is left to the interpretation of the user. In addition, many metadata elements are only required in informal text (see the description of the ADL Gazetteer service[1] in Figure 1.2)—which is reasonable given the human effort required for metadata entry— and metadata is not cross-linked between information sources[2]. Therefore, the challenge that lies in front of us, is to solve this so called *semantic heterogeneity* problem.

To link back to our travel plug metaphor, what we are aiming for, is an 'intelligent' plug that recognises the voltage and adapts accordingly. At this point it is informative to switch to a closer metaphor, namely PC motherboard expansion slots and cards (e.g., sound card, video card, etc.), which commonly applies a so called plug-and-play mechanism (the PC recognises the card and it installs its drivers automatically). The plug-and-play paradigm lies close to the semantic interoperability issues that we are facing in current distributed geo-services. The units/services to be connected have to be recognised by a mediator. This so called *inferencing* is done with the help of descriptions that are built into each service (the services are called *self-descriptive*), see Figure 1.3.

Mediation between descriptions may be performed by a third service or by the services themselves. If the mediation is done in-service then it is said they

[1]http://middleware.alexandria.ucsb.edu/client/gaz/adl/index.jsp

[2]Note that these arguments are similar to the remark above, that '...many valuable information sources are lost in the information cloud...'

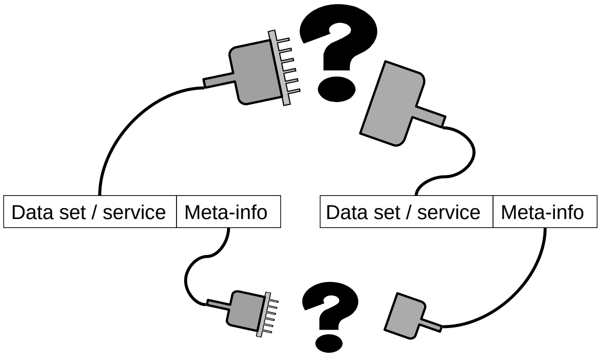


Figure 1.3: Inferencing resolves the connectivity question between information sources and is done with the meta-information that describes those information sources. We should be able to answer the question on top by answering the question at the bottom.

'recognise each other'. The fact that plug-and-play is often referred to as *plug-and-pray* is due to the malfunctioning of the inference engine that is used to resolve the connectivity between the descriptions of the plugged units. Part of the challenge is to get this process to an acceptable level of quality.

Although the idea of making geo-services plug-and-play is an ambitious goal, the fundament is on the horizon. One of the first steps is to facilitate semantically enriched descriptions of geo-services and to apply inferencing to discover relationships between them, in order to semi-automate the process of information integration and service chaining. This is considered to be one of the major challenges in this research.

What distinguishes geo-information services from other services?

This can be best understood by analyzing the kind of information they act on: geo-information (a similar elaboration can be found in [164]):

- Geo-information contains representations of phenomena that are related to the earth surface. Those representations model the properties of these phenomena and the relationships between them, with respect to three aspects: space, time and theme. Properties may range from a coordinate pair to a cadastral transaction date. Relationships may involve an intersection between roads, an interpolated height value of two bore hole points, proximity expressions, such as 'the nearest bus station', etc. Geo-services are a special kind of services that support information of this type and enable the user to derive new geo-information, based on spatial, temporal and thematic relationships.

- Data collection by multiple organisations results in multiple versions of the same geographic feature with respect to accuracy, geometry, conceptual model, etc.

- Geo-services are implicitly connected by the geographic location of the features they process. This has implications for the interoperability of geo-services. For example, the validity of a service (e.g., a routeplanner) may be bound to a specific geographic area, which could imply it cannot be used in combination with services with another validity area. On the contrary, services that seem to be incompatible due to differences in representation (e.g., geometry, coordinate reference system), may turn out to be useful in combination, because they contain information on the same locations.

- Many geo-services combine information based on spatial relationships, and as such they derive new information, by adding new spatial attributes to the features' existing spatial, temporal and thematic attributes. Due to this multiple-dimensionality, geo-information is of a complex nature. The intended use of its contents in geodata sets and in geo-services needs a thorough description of its semantics (meaning). Therefore, geo-specific concepts, such as geometry types and spatial relationship types need to be defined formally and unambiguously.

- Geo-information can be massive. Therefor, care has to be taken in shipping only relevant parts of a dataset or as alternative, shipping software code.

- Geo-services can make use of existing software libraries of mathematical functions due to the geometric nature of its input and output.

- As geo-information often needs to be reused in different and dynamic application domains, geo-services are required to be flexible and interoperable with each other so that new applications can be built on-demand. Moreover, the repeatedly reuse of processed information necessitates the availability of unambiguous meta-information on the processing history (lineage) of geo-information.

- Finally, end-user applications implementing geo-services often use a map as an interface. Natural user-interactions, like zooming and panning, are functions that many geo-services are expected to support.

1.2 Research objectives

The general objective of this research is to

> *Provide solutions for the computer-aided integration of distributed heterogeneous geo-information and geo-services, based on their semantics, to support on demand geo-processing.*

More specifically, the research aims at:

- Providing a realistic and applicative framework in which the semantics of geodata sets and geo-services can be described, which allows for automated reasoning between these descriptions.
- The applicability of the framework to a variety of application fields in GIS.
- Extensibility. The framework should be extensible with extra descriptions of its semantics, in order to accommodate the step-wise implementation of an application domain.
- The use of influential information technology standards and geo-information standards that support interoperability.
- The use of open source software. The licensing scheme of open source software eases the use, modification and distribution of prototype software, related to the research.

Research questions

Specific research questions that are taken as guidelines in this research are found below. The chapter numbers in brackets behind each question indicate the chapters that address the relevant issues related to this question. All research questions are answered specifically in Section 9.2.2.

1. *What are the requirements of on demand geo-processing?* (Chapter 2)
2. *What are the key problems in making geo-services interoperable and what solutions are currently available?* (Chapters 2,3,4)
3. *Up to what extent can and should semantics be modelled to support semantic interoperability (both for geo-information and functionality of geo-services)?* (Chapters 5,6,7)
4. *How can we semantically enrich meta-information of geo-information and of geo-services to support service chaining?* (Chapters 4,7)
5. *What are the capabilities and limitations of Semantic Web tools to support geo-semantic interoperability?* (Chapters 7,8)
6. *What are minimum requirements for a semantic interoperability framework for geo-services?* (Chapters 4,5,6,7)

Research scope

The scope of this research can be characterised with the following demarcations:

- This thesis concentrates on the modelling of the functionality of geo-services and their inputs and outputs. Reasoning is performed with spatial *concepts*, not with the spatial characteristics of *geographic instances* of real-world objects. In other words, the reasoning applied in this research, can infer that a service is capable of, for example, topological containment, but it cannot infer the containment relationship between 'Louvre' and 'Paris'. Reasoning about the actual real-world objects is only relevant if these objects are made part of the ontology. This research assumes they are part of a database. However, in some parts of the thesis, methods are presented that are 'close' to incorporating such spatial reasoning. These will be indicated wherever applicable.

- This thesis focuses on spatial and thematic aspects of information, not on temporal aspects.

- The geo-information ontologies, developed in this research, are based on work by OGC and the ISO 19100 series of standards, in particular (the short names appear between brackets and can be found in Appendix B): 19101 (Reference), 19103 (Concept), 19107 (Spatial), 19109 (GFM), 19110 (Catalog), 19111 (RefCoord), 19112 (RefIdent), 19119 (Services), 19125 (SFeature), 19126 (FACC), 19131 (Product), 19133 (LBSNav) and 19136 (GML). For an overview of the contents of these and other 19100 standards, see Appendix B. Three-dimensional objects, as part of these standards, are not supported.

- The geo-operation ontology (OPERA), developed in this research (see Section 5.4), provides a general structure for all types of geo-operations. Its detailed structure focuses on feature processing operations, such as buffer, overlay, interpolation and transformation. Its design has concentrated on vector-based feature operations, but raster coverages are supported as well.

- With respect to service chaining, this thesis elaborates upon the aspects of discovery and abstract service composition, but not on the concrete composition and actual execution of a service chain. Languages such as WSBPEL (briefly introduced in Section 3.2.2) and SOAP (briefly introduced in Section 2.4.3) are therefore not considered to be at the core of this research.

- This thesis does not embark upon linguistic issues in the context of interoperability.

Fields of application

The main target group of this research is the group of geo-information engineers who are confronted with information integration issues and service infrastructural issues that cannot be resolved due to the lack of a common understanding of the meaning of the source content that is to be found or integrated. This research aims at providing structural web-based solutions for dealing with such issues. For an important part, these solutions draw upon existing state-of-the-art methods, developed in generic computer science, specifically from the field of the Semantic Web, and apply them to the field of geo-services. However, some of the presented solutions are not geo-specific, such as the structure of the proposed interoperability framework and annotation method, and may be also applied to other application fields.

1.3 Research approach

To achieve the research objectives and answer the research questions of Section 1.2, a *semantic interoperability framework* has been designed and implemented according to the demands set in four use cases, using, as much as possible, existing interoperability standards. The design and implementation choices are as follows. The starting point is the use of principal models of geo-information under development by the Open Geospatial Consortium (OGC) and ISO/TC211 (International Organization for Standardization, Technical Committee 211, Geographic information/ Geomatics). These models contain knowledge of many international specialists and follow a state-of-the-art approach. In addition, the specifications of OGC are supported by industrial implementations. The Object Management Group's (OMG) Universal Modelling Language (UML) [202] is used in all ISO/TC211 documents and in order to maintain uniformity, UML is also deployed in all other model documentation in this thesis. The ISO models are used as a basis for three basic types of geo-ontologies, i.e., (1) *concept* ontology type, containing conceptualisations of real-world phenomena, (2) *symbol* ontology type, containing symbolic representations of geographic features and (3) *operation* ontology type, containing a classification of geo-operations. This separation is made in order to acknowledge the different levels of abstraction in geo-information. In all models, a *feature* is considered to be the fundamental unit of geographic information. One of the starting points for the classification of geo-operations has been to consider the elements of the feature concept ontology and the feature symbol ontology to be representatives for the input and output parameter types of the operation classes. The symbol and operation ontologies are implemented fully. Concept ontologies represent the conceptual world and a complete implementation is only feasible at a generic level. As this thesis aims at providing an applicative framework, a decision was made to implement a limited number of *specific* concept ontologies. These represent the Dutch geo-information model NEN3610 [195], a data model of the Dutch topographic service (TOP10NL) [17] and two domain specific models ('Riskmap' and 'Travel'). Their content is chosen in overlap so as to serve as testing materials. The ontologies are implemented in the World Wide Web Consortium's (W3C) Web Ontology Language (OWL) [184], which is chosen because it is currently the most versatile ontology language that supports web based applications. OWL is supported by most ontology editors and other tools. Data sets and services are described as *advertisements* with OWL constructs in order to allow integral analysis in one ontology environment. Data set and service descriptions are also queried with OWL constructs for the same reason. Such queries are formulated as *requests* and are handled by a reasoner. The RacerPro reasoner [233] has been chosen, because it is renowned for its *ABox* reasoning capabilities. An ABox is the part of the knowledge base that contains assertions (hence the term 'ABox'). ABox reasoning allows to infer ontological relationships between concepts *and* instances of concepts. To allow a user to perform reasoning and service chaining in one environment, an integrated prototype ('GeoMatchMaker') has been built by adapting and integrating existing software modules. GeoMatchMaker implements the OWL ontology for web services, OWL-S (S stands for services)[181]. OWL-S was selected for its clear structure and conciseness.

Four use cases, each of which addresses different aspects of semantic interoperability, demonstrate the actual use of the framework in a practical environment. Figure 1.4 shows the typical context of on demand processing in which these use cases are thought to take place. A user who wants to perform a specific task is confronted with a distributed data and service repository. Depending on his expertise, he can decompose his task and sends dedicated requests to a discovery mechanism (the GeoMatchMaker prototype). The right combination of data and services is found using an iterative process. Once finished, the composite is executed on a computer network.

To allow others to use the developed framework, documentation and references are available in this thesis detailing all of the ontologies and tools used. All ontology constructs have been documented in Description Logics notation, in order to remain implementation independent (the theory of Description Logics is the mathematical foundation of OWL; see Section 4.2.1).

A note on terms Whenever the term *information source* is used in this thesis, it refers to data sets as well as services. The term *meta-information* is used to indicate descriptive information about data sets or services.

1.4 Related work

This section describes related work, close to the research objective of this thesis. It elaborates on previous and current work in the field of the formal modelling of *geo-semantics.* Geo-semantics is termed herein as the semantics of spatio-temporal entities in geo-information and geo-information processing. For references to a

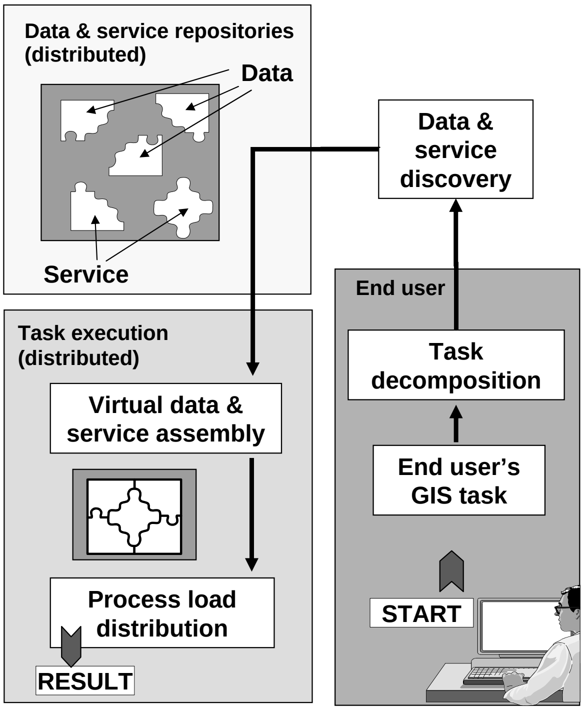

Data & service repositories (distributed)
Data
Service
Data & service discovery
End user
Task decomposition
End user's GIS task
START
Task execution (distributed)
Virtual data & service assembly
Process load distribution
RESULT

Figure 1.4: On demand processing: context of the use cases.

wider embedding of this research work, the reader is referred to Chapters 2, 3 and 4. Some recent papers (late 2005 and onwards), have been included in this section, but as they were not published at the time of undertaking the major part of this research, their results could not be used in the actual research.

The modelling of geo-semantics has been part of GIS from day one. Also, the field of geo-information has a long tradition of metadata handling and standards development. However, making the semantics explicit for the purpose of model integration and reuse, has only come to fruition recently. In this respect, many of the semantics of geo-information have been brought closer to the formal level of ontologies by the models provided by OGC specifications and ISO 19100 standards. This section elaborates upon in particular the research efforts that have embarked upon the use of formal ontologies in geo-information handling.

Major developments in the area of geo-semantics can be observed at several research platforms, in particular the conference series of COSIT[3], GIScience[4] and AGILE[5] and the recent GeoS 2005 workshop [242]. A list of geo-semantics research groups and individuals is being maintained at http://www.geosemantics.org.

A selection of prominent developments is highlighted in the remainder of this section by distinguishing between the following main areas of application:

1. Foundations of geo-semantics
2. Spatial reasoning
3. Integration and multiple-representation of spatial data and schema
4. Geo-semantic frameworks and ontology building
5. Geo-service interoperability
6. Geo-service functionality and workflow modelling

This section ends with the positioning of the author's work.

Foundations of geo-semantics

Several key research efforts have elaborated on the application of ontologies in GIS. Foundations on the essence of ontologies in GIS have been laid down by Frank in the Chorochronos project [83] and by Mark et al., reporting on issues in cognitive models of geo-information [180]. GI-related interoperability issues that occur in federated databases have been highlighted by Bishr [29] in an ontological setting. Proposals for integrating ontologies in future GIS have been put forward by Fonseca and Egenhofer [64, 78] and in [77]. More recently, the term *semantic reference systems* was coined by Kuhn [153, 154]. It refers to the foundation

[3]http://www.cosit.info
[4]http://www.giscience.org/
[5]http://www.agile-online.org/

of formal methods for computational mappings between ontological contexts in a semantic framework. Kuhn speaks of 'transformation' and 'projection', analogously to their counterpart methods in spatial reference systems. Bittner et al. [31] and Agarwal [3] both give a comprehensive overview of the interoperability issues involved in ontology-based geo-information integration.

Spatial reasoning

Spatial reasoning involves the inferencing of spatial relationships from spatial data. In this area, fundamental work has been carried out in the field of qualitative spatial reasoning [53, 248] and Description Logics [106]. [257] provides an overview of current approaches. The Bremen University Semantic Translator for Enhanced Retrieval (BUSTER) project [280] uses a place name structure to define spatial footprints of objects as a basis for spatial reasoning between place name regions. The recently conducted OGC Geospatial Semantic Web Interoperability Experiment (GSW IE) has embarked on the processing of geographic queries based on their ontology-based content, and has aimed at change proposals for OGC Web Feature Service (WFS) / Filter Encoding specifications [169]. The involvement of temporal aspects in reasoning in a spatio-temporal approach is presented by Reitsma and Bittner [238], which draws upon a model of enduring objects and perduring processes, as presented earlier in [101]. The inclusion of linguistic semantics for performing spatial queries using imprecise spatial and temporal references such as 'far' and 'near' is argued by I. Budak Arpinar et al. [13]. An approach for automating semantic annotation for spatial relationships has been reported by Klien and Lutz [145].

Integration and multiple-representation of spatial data and schema

The application of ontologies for the integration of geodata sets and geodata models has been fostered by mapping agencies and organisations tasked with the management of national spatial data infrastructures. Their motivation is, amongst others, (1) to streamline the data acquisition and provision with the queries of end-users which are often expressed in different world views [100], (2) to allow integration with other data sources [94, 95], and the facilitation of *update propagation* (the reuse of updates from one data set into another) [266].

One aspect of data set integration is semantic similarity between geo-spatial concepts, of which measures have been proposed by Rodríguez and Egenhofer [241] and Brodeur and Bédard [40]. Schwering and Raubal follow an approach that accounts for spatial relations between concepts [251]. Duckham and Worboys [63] propose a method that includes extensional knowledge (instance-based information) for schema integration. Examples of specific fields that exploit geo-semantics modelling for information integration, are cross border mapping [139] and cadastral domain modelling [113]. Methods to integrate cadastral domain models by means of translating UML into machine ontologies have been investigated in [114].

The work of De Vries [281] is at several points related to the research presented in this thesis as these two research efforts were carried out in the same time frame and with occasional cooperation. The work of De Vries can be distinguished by its focus on standards-based (mainly OGC) integration of actual geodata, covering topics such as client-server architecture, OGC Web Services, Geography Markup Language (GML), geodata visualisation and model harmonisation.

Geo-semantic frameworks and ontology building

The creation of geo-semantic frameworks is discussed in [142] by providing a general approach for integrating ontologies, and in [75] in terms of formal mappings between ontologies. Several research efforts have resulted in OWL ontologies and tools, such as information brokers, see [280] for example. A set of OWL ontologies, based on ISO 19100, has been developed by Drexel University [124] and is available at http://loki.cae.drexel.edu/∼wbs/ontology/list.htm. Several other initiatives have resulted in geo-spatial application ontologies (e.g., Mindswap[6][116], SWEET[7]). The ACE-GIS project has produced domain ontologies based on the ISO standards 19107 (Spatial), 19111 (RefCoord)and 19112 (RefIdent) [227, 228] and several application ontologies for an emergency scenario[8]. With respect to geo-ontology design, Lutz and Klien [173] and Klien and Probst [147] give excellent guidelines for the integrated creation of domain and application ontologies.

Geo-service interoperability

The application of semantic modelling to geo-services involves characterising service functionality and their input and output, in order to achieve semantic interoperability. It has gained interest with the increasing popularity of OGC services. Approaches for ontology-based geo-service discovery are currently being researched by the Münster Semantic Interoperability Lab (MUSIL). In a number of publications, approaches are reported for query formulation [173], a query client interface [146] and an integrated architecture, based on a semantic catalog [172]. Methods for ontology-based annotation and registration of geo-information sources have been reported in [146, 227, 277]. The exploitation of geo-semantics modelling to specific service application fields can be particularly found in the area of disaster management [229], sensor network services [247] and Location Based Services [250, 288].

Geo-service functionality and workflow modelling

Approaches of formal modelling of geo-service workflow have been presented in [6, 190]. The need for ontology-based approaches has been identified in [66], but

[6]Ontologies available at: http://www.mindswap.org/2004/geo/geoOntologies.shtml

[7]Ontologies available at: http://sweet.jpl.nasa.gov/ontology/

[8]Ontologies available at: http://musil.uni-muenster.de/onto/ACE/

the actual implementation of ontology-based workflow models as present in OWL-S to geo-service functionality is clearly an open field for further research.

Relations with this research

There appears to be a research niche, especially in the last topic (geo-service functionality and workflow modelling). Current research has only shallowly embarked upon the following issues:

- The development of strategies for the design and use of ontologies that support the description of geo-services in terms of their input, output, and functionality.
- The application of reasoning with the above descriptions to support the discovery of, and interoperability between geo-services.

The above issues are the main topics to which this thesis work contributes. It does so, by proposing a semantic interoperability framework, embedded in existing geo-information standards, and applying it in a prototype environment. The thesis reports on theoretic and practical aspects of this framework and covers issues on all six of the above topics, but specifically focuses on the last three topics.

Previous research contributions by the author have covered topic 3 (Integration and multiple-representation of spatial data and schema)[256], topic 4 (Geo-semantic frameworks and ontology building) [157, 162], topic 5 (Geo-service interoperability) [160, 161, 163] and topic 6 (Geo-service functionality and workflow modelling) [158, 159, 164, 165].

1.5 Thesis outline

This section gives a brief overview of the contents of the chapters in the thesis. The general structure of the thesis is as follows. Chapter 1 provides an introduction to the thesis work. Chapters 2-4 provide theoretical foundations for the semantic interoperability framework developed in this thesis. They contain relevant views from existing literature as well as theory, developed as part of this research. Chapters 5 and 6 describe the development of the semantic interoperability framework and form the core development of this research. Chapters 7 and 8 describe the thesis work related to the prototypes that implement this framework and which are used to verify the presented concepts. Chapter 9 contains conclusions and recommendations. An overview of the contents of each chapter is given below.

Chapter 1 provides a general introduction and characterisation of the thesis, including research objectives, research questions, research methodology, related work and thesis outline.

Chapter 2 addresses relevant issues that are involved in on demand information processing. It introduces the reader to common concepts of interoperability and distributed services, that are reported in literature. In addition, it contains an introduction to four use cases, in order to set a practical scene for the rest of the thesis. The use cases are expanded upon in Chapter 7.

Chapter 3 describes the characteristics of services that are relevant for on demand processing. It provides models for abstracting information and processes, based on existing work and presents the principles of service chaining. The presented models are used as a basis for the semantic models in Chapters 5 and 6.

Chapter 4 provides an introduction to existing semantic modelling principles and ways to access these models. Description Logics and Semantic Web techniques play a central role in creating ontologies and reasoning with ontology-based information sources. A major part of this chapter consists of generic information technology, such as the Web Ontology Language (OWL) and OWL-S (OWL for web services). Towards the end, it focuses on geo-information applications.

Chapter 5 presents an integrated framework of ontologies as a basis for the description of, and the reasoning with, geo-information and geo-services. It applies the modelling techniques, discussed in Chapter 4.

Chapter 6 demonstrates how OWL, OWL-S and reasoning (as described in Chapter 4) can be applied to the framework of Chapter 5 for geo-information matching and service chaining (discovery and abstract composition of services).

Chapter 7 describes the implementation of the use cases, introduced in Chapter 2. These four scenarios concentrate on different aspects of the practical applicability of the framework of Chapter 5 and techniques of Chapter 6. They are demonstrated through performing several tests with data set/service advertisements and requests. The implementation uses the prototype ontology, named*OnToGeo* and the prototype application, named *GeoMatchMaker*. They are described in Chapter 8.

Chapter 8 describes the software implementation of the prototypes developed in this thesis work and the workbench tools that were used for performing the tests in the use cases, as described in Chapter 7.

Chapter 9 provides a summary of the thesis, its final conclusions, its main contributions and ends with recommendations for further work.

Appendix A provides the UML elements used in this thesis that are represented with a notation that deviates from the standard UML 2 notation.

Appendix B lists all current ISO 19100 standards with their numbers and short name.

Appendix C lists the atomic reference types for geo-operations in the OPERA-R ontology and describes their functional semantics (the *R* in OPERA-R stands for 'reference operations'). This appendix is linked to Section 5.4. The list can be considered to be the base taxonomy for OPERA-R. The input/output parameters of each operation are given in Appendix D.

Appendix D lists the input/output parameters for each class of feature processing operation in OPERA-R, as described in Sections 5.4, 5.5 and Appendix C. The input/output parameters are expressed in terms of the classes in the SYMBOL ontology.

Appendix E contains an OWL code example. It shows the code of the Alexandria Library (ADL) Gazetteer service, which is used as example geo-service in several parts of the thesis.

Appendix F contains a WSDL code example for the ADL Gazetteer service.

Appendix G provides all ontology mappings between ISO 19119 (Services) classes and OPERA classes. These mappings are explained in Section 6.3.1.

Chapter 2

Interoperable distributed services

The needs for organisations to integrate intraorganisational and interorganisational processes have directed these organisations to make use of well-defined and interoperable processes and avoid duplication of data. Over the past two decades, computer software systems have shifted from single purpose monolithic environments to modular ones. This has enabled organisations (1) to distribute software processes over geographically distinct systems and at the same time keep them synchronised, and (2) to reuse software modules. An example of the first is the use of web services for purchasing books on-line at Amazon.com. An example of the second is a routeplanner module that can be integrated in a variety of third party web pages. The process of combining services to achieve larger tasks is called *service chaining*.

Interoperable distributed services can play an important role in on demand information processing, because they can provide specific (additional) functionality in combination with each other or in combination with existing information systems. In the latter case, this may also involve the creation of service chains that are not executed fully by machines, but require a human effort (e.g., manual start of a service, parameter provision) at certain places in the chain.

This chapter discusses the elements of distributed services and models that support their interoperability. This is done by raising issues and report on approaches in the light of on demand processing. Section 2.1 provides the basic characteristics of services as separate units and as units in a distributed service composition. Sections 2.2 and 2.3 highlight the heterogeneity issues that hamper the effectiveness of such composition and reports on how this can be overcome by interoperability measures. Existing interoperability models are provided in Section 2.4. Section 2.5 elaborates upon geo-services as a special case of services. It contains examples and a coarse-grained classification of functionality. Section 2.6 introduces a set of four use cases which have been developed as part of this research. They are considered to be representative for the application domain of distributed services and on demand geo-processing. Section 2.7 contains a summary and reflection on the current chapter.

2.1 Distributed processing paradigms

The ISO Reference Model on Open Distributed Processing defines distributed processing as follows:

Term 2.1 *(Adapted from [125])* Distributed processing *is defined as information processing in which discrete components may be located in different (geographic and/or organisational) places, and where communication between components may suffer delay or may fail.*

Key characteristics of system distribution are [126] the spreading of components across space and/or across management regime and their parallel and asynchronous execution. Due to the independence of the system's component providers, heterogeneity appears in hardware and software. The components have a certain control autonomy and may be physically mobile.

To enable the creation of flexible intraorganisational and interorganisational distributed systems, they have to be open (supporting internetworking) and modular. In a modular software system, the functionality of the system is split-up into well-defined modules. Modules are assigned areas of functional responsibility [20]. Modules are design artefacts and may have associated products such as interface specifications and code. Note that modular systems are not necessarily distributed.

2.1.1 Services as units of data processing

Reasons for splitting-up functionality in modular software systems are the needs for:

- Flexibility to swap one module with another with the same specifications
- Reuse of modules
- Possibility to distribute a group of modules over multiple hardware systems
- Ability to connect distinct business processes between organisations

Good modular architectures make dependencies explicit and help to reduce and control these dependencies [260]. Developments in information technology have brought numerous paradigms, such as object-orientation and service-oriented architectures and brought new concepts such as objects, components, agents, etc.

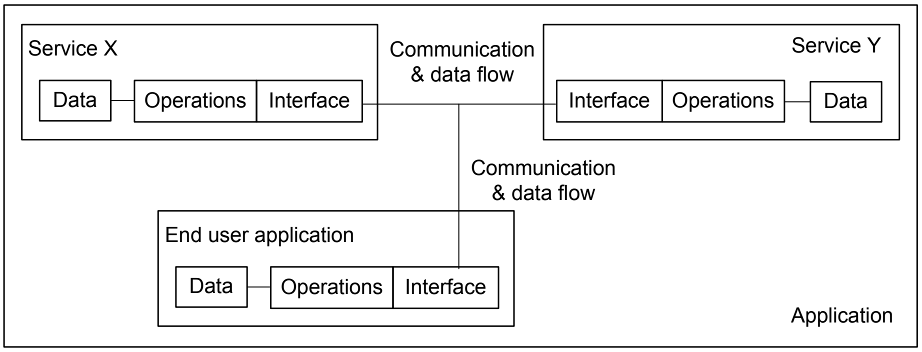


Figure 2.1: Basic elements of interaction in a service environment.

The often unclear differentiation between some of these concepts has caused confusion in literature. In the following section the principles of modular processes and distributed processing are clarified and argumentation is provided for the terminology used throughout this thesis.

Services Services as basic units of processing are considered to make available operations and one or more interfaces that give access to them. They may be tightly-coupled with data instances, needed by the operations (so called tightly-coupled data, see definition of Term 2.8). Modular software systems can normally be broken down into such basic units. We distinguish between client side software and server-side software. The latter can encompass one or more services. Figure 2.1 portrays a typical service environment in which services communicate with each other and with an end-user application. In the context of the client/server paradigm, an end-user application always has the role of a client. In between-service-communication a service X may become a client of another service Y and service Y may be a client of service Z. The whole of services and end-user application is called an *application.*

In summary, a service is defined as follows:

Term 2.2 *(After [109]) A* service *is an abstract resource that represents, through an interface, a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities. Entities may be persons or organisations. To be used, a service must be realised by a concrete* software component *and may be tightly-coupled to a data set. Services may be composed into composite services.*

The terms software component and web service are often used in the same context, but refer to different concepts:

Term 2.3 *(Taken from [109]) A* software component *is a software object, meant to interact with other components, encapsulating certain functionality or a set of functionalities. A component has a clearly defined interface and conforms to a prescribed behaviour common to all components within an architecture.*

Term 2.4 *(Taken from [109]) A* web service *is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialisation in conjunction with other web-related standards.*

The terms WSDL and SOAP are explained in 2.4.2

Web services can be seen as deployed components paired with a service provider. Unlike traditional web servers that serve web pages for consumption by people, web services provide computational services to other systems [260].

Any service consists of interfaces, operations and may be tightly-coupled with data to provide its functionality to clients or other services. An operation is defined as follows:

Term 2.5 *An* operation *is a function that a service makes available. An operation has input and output parameters. Operations may be combined into composite operations.*

The interface of a service defines the service's access points for a client or other services:

Term 2.6 *(Taken from [109]) A* service interface *is the abstract boundary that a service exposes. It defines the types of messages and the message exchange patterns that are involved in interacting with the service, together with any conditions implied by those messages.*

A service is accessible once it is *bound*. Binding takes place as part of the negotiation between a service requestor and a service provider and is defined as follows:

Term 2.7 *(Taken from [109])* Binding *is the mapping of an interface and its associated operations to a particular concrete message format and transmission protocol.*

In addition to the input data that a service may accept and the output data it may produce, a service instance may be tightly-coupled with a data set instance [132], referred here as *tightly-coupled data.*

Term 2.8 Tightly-coupled data *is data, that is tightly-coupled to the service (and its operations) and is used by the service to provide meaningful responses to client requests.*

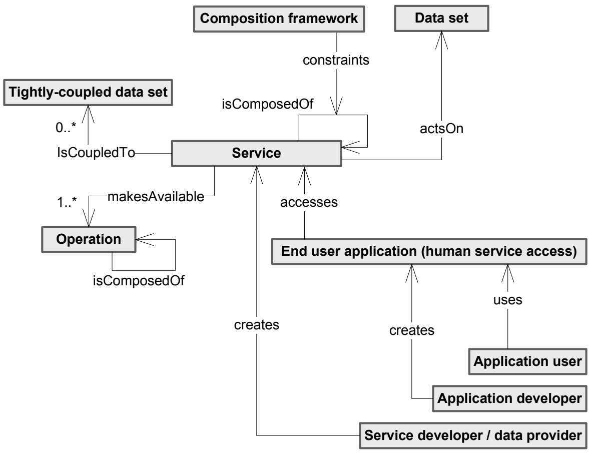


Figure 2.2: UML class diagram showing the user context of service access.

An example of tightly-coupled data is a road data set of a country's national mapping agency that is tightly coupled to a routeplanner web service.

Several properties of services become more visible when we place them in a user perspective of a modular software framework (see Figure 2.2). Central in this figure is the service. A service makes available one or more operations. The reflexive relation 'is composed of' indicates that a service may be composed of other services (making it a composite service). Such composition may be constrained by a *composition framework*, which is defined as follows:

Term 2.9 *A* composition framework *prescribes for a set of services the specifications to which they and their potential service composition should comply.*

For example, web services may be constrained by SOAP messaging (see also Section 2.4.3.

The 'acts on' [1] relationship in Figure 2.2 between service and data implies that the service may accept input data and produce output data. In fact, the service acts on *information* contained in the data.

In practice a number of services does not handle input data from data sets, because they create data from user-actions (e.g., in digitising).

A service may also contain a workflow, which is defined as follows:

Term 2.10 *The* workflow *of a service defines the execution order (control flow) and data bindings between its internal operations (or sub-services, in case it is a composite service).*

[1]the ISO Geographic information - Reference model [128] refers to 'operates on'

A stand-alone workflow, e.g., in the form of a description, is not executable. It must be embedded in a service, even if the workflow is described in the form of a script.

An end user application takes care of the communication between a human actor and a service or a combination of services. The application developer provides the application with an appropriate human user-interface. The functionality of access to a particular service may be offered in one case directly through an end user client, in another case through another service. The end user application as 'first' access to the service chain may be as thin as a web browser. The application developer is involved with service selection as well as the initial development and the maintenance of the end user application.

Services may be also invoked by other services, such as software agents. In this case the invoking service is seen as a composite service that internally acts as a client to the other services.

An important notion in the composition of services is the *service chain*, which is defined as follows:

Term 2.11 *(Adapted from [132]) A* service chain *is a pattern of services, which can be represented by a directed graph, and where, for each adjacent pair of services, occurrence of the first action is necessary for the occurrence of the second action.*

Figure 2.3 shows the comparison between single services and a service chain. (a) shows a single service making available a single operation. In (b), a single service makes available multiple operations, but the interface hides from the outside of the service the internal input/output parameters between its operations. In (c), operations are made available in separate services, forming a service chain. Service chains may also occur as a combination of (b) and (c).

Other terminology

In this thesis the combination of services and a client application is referred to as *client/service application* or *application* in short.

The term *metadata* is often defined as *data about data.* However, in this thesis, the term metadata has a wider scope and means *data about something.* This includes data descriptions, as well as operation descriptions and service descriptions. The term *meta-information* is used to emphasise the semantic properties of the metadata. In the context of GIS, the term *feature data* is used to indicate the actual *data* that is described by the metadata. Feature data represents features (abstractions of real world phenomena).

The term *service discovery* refers to the process of finding services that (partly or completely) fulfil a user task.

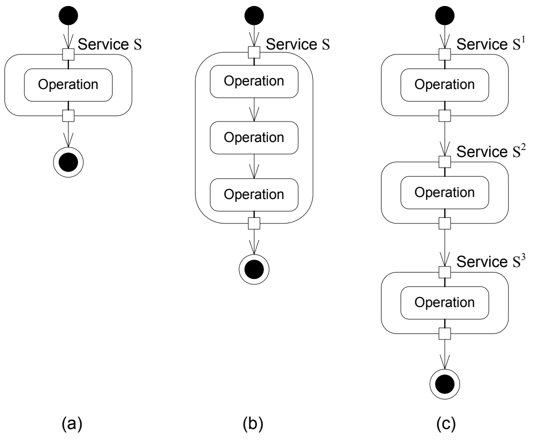


Figure 2.3: UML activity diagrams of (a) a single service with one operation, (b) a single service with multiple operations and (c) a service chain. A service's input and output parameters are the publicly available parameters of its operations (sent through its interface, indicated by the small square boxes). Although in (b) and (c) only sequence patterns are shown, other patterns, such as *split* and *iterate* are also possible.

2.1.2 Classification of services

Services can be classified along different classification axes (adapted list from [274]). We distinguish:

- Application domain. An application domain refers to the concepts and operations that are common to a particular work field, such as land use planning or disaster management. In case the service is bound to an application domain, it is called *task-specific*. Classification in application domains is subject to levels of detail; Within each application domain, more fine-grained classifications are normally possible.

- Scalability: The ease with which a service can handle larger inputs, higher number of inputs of the same type, or more requests per time interval.

- Intended type of coupling. Services are deployed in environments ranging from tightly-coupled to loosely-coupled (see also section 2.1.3). This is reflected in the design of the service's data, operations and interface.

- Data dependency. A service's data characteristics involve (1) input (2) output and (3) the data that is internally used for its operations, the so-called tightly-coupled data. The degree to which the service relies on the tightly-coupled data (in other words: its data dependency or strong data coupling) has a strong effect on its reusability. Consider the comparison between a generic multiplier service and a more specific currency converter service. In the generic form, the service needs the same data as in the specific form, but in the generic form the data is not tightly-coupled data, but data provided by the client.

A similar example can be given in a geographic analysis setting: An algorithm that determines the shortest route between two points over a network can be implemented by a generic service as well as a service that specifically calculates the shortest route over the road network, represented by a road data set (for example of the Dutch national mapping agency). Within the same route finder context we can distinguish between a service that allows for a destination input, specified by the application user and one that has a fixed destination (the latter is often used in business web sites to provide the client with a route to the business premises).

In application domains where we need frequent updates of data (in the two examples respectively updates of currency rates and updates of road availability), typically such data sets can be offered through services. Data dependency then exists through tightly-coupled services, rather than tightly-coupled data.

- Interface. Services often perform their tasks within specific composition frameworks by prescribed standardised interfaces. Services may be self-descriptive of their interfaces. This is a common practice in loosely-coupled architectures, such as web services.

- Reusability: The ability of a single service to be deployed multiple times in combination with different (combinations of) services. The reusability of a service depends —amongst other aspects— on its type of coupling and the degree to which its interface, tightly-coupled data and application domain matches with the other services. The demand of reusability has an influence on the 'weight' of a service. In this respect there is a dualism in making a service more reusable. It may be stripped to offer only its prime functionality or it may be provided with more functionality to make it more robust to different application domains. According to Szyperski et al.[260] there is no universal rule to determine the optimal weight in a cost perspective. It depends on factors of the service providers and on the target markets.

- Performance. Performance is concerned with how long it takes the service to respond when an event occurs [20].

- Availability. Availability of a service is the probability that it will be operational when it is needed [20].

- Security. Security is a measure of the service's ability to resist unauthorised usage while still providing its services to legitimate users [20].
- Cost. Services can be offered through different payment schemes, such as pay-per-use or flat fees for a license period. Pay-per-use schemes may implement payment units of computing time, processed data size, number of user interactions, number of software component calls, etc. Schemes that involve calls to composite services charge for the components in a so-called billing hierarchy [260]

Modular system architecture design The context of service *combinations* can be classified along the following classification axes:

- Scope of distribution. Distribution may relate to:
  - Hardware, i.e., the way in which the services are distributed on a number of nodes, which is determined by that number, the distance between them and the network topology.
  - Organisational level, i.e., whether the services are distributed within the boundaries of the office, the enterprise or in a wider geographic range.
- Composition framework. Services are designed to operate in compliance with specifications that prescribe the way they are deployed and the way in which they communicate once deployed. This also influences the type of coupling between services.

Dynamics of service composition The following aspects play a role in the way services are composed:

- User demands:
  - How frequently end users access the services.
  - How diverse the tasks are, that are generated by the environment in terms of
    - Functionality.
    - Requested application interface.
  - Urgency. The composition process may differ in speed, depending on the time span between the initiation of a processing task and the requested composition.
- Availability of services. The degree of instant availability of services determines how many new services have to be acquired (or created) to build a new application or composite service.

- Dynamics of service provision. The dynamics of an environment may cause frequent changes in the provision of services.

In an environment with many similar tasks, applications may have to be rebuilt many times. An application developer may want his application to be generic, i.e., to respond to a variety of input parameters. He can achieve this either by (1) using a set of generic services without the need to regularly acquire new services for different user requests or (2) by using many specific services which are composed on-the-fly to solve the task at hand. Option 1 would be typically performed using *design-time coupling* and option 2 using *run-time coupling*. For an explanation of these terms, see Section 2.1.3.

2.1.3 Tightly- and loosely-coupled systems

The deployment of services results in the execution of an application that makes use of these services through coupling.

Coupling is defined as follows:

Term 2.12 *(adapted from [156])* Coupling *is the degree of how strongly one system element is connected to, has knowledge of, or relies on other elements. Elements can be software components, services, data, etc.*

We distinguish between loose coupling and tight coupling. An element with loose coupling is not dependent on too many other elements. 'Too many' is context-dependent and refers to the consequences that are faced with tight coupling. These consequences are [156]:

- Changes in dependent elements enforce local changes.
- Element properties are hard to understand in isolation.
- Reuse of an element needs the presence of dependent elements.

There is no absolute measure for the degree of coupling. The terms loosely-coupled and tightly-coupled are used to compare the coupling characteristics of alternative system implementations or architectures.

In system deployment another related distinction is made between *design-time coupling* and *run-time coupling*. Design-time coupling typically is the starting point for building tightly-coupled systems and run-time coupling for loosely-coupled systems. However, this has to be put in perspective: tightly-coupled systems can still be built with run-time coupling and vice versa.

Run-time coupling only needs dependent services to be present at run-time. This allows for flexibility and independency in the implementation of these services. Web services are prominent examples of services intended for run-time coupling. Design-time coupling necessitates the services that are functionally dependent to be present in the same compiler environment at design-time. An example of a GIS architecture using design-time coupling is ESRI's ArcObjects architecture.

The Service-Oriented Architecture (SOA) facilitates the creation of loosely-coupled services, such as web services (for a description of SOA, see Section 2.4.3). Typically, loosely-coupled services can be easily interchanged with others of similar functionality and an equivalent interface. This offers a great potential for service integration. However, a set of loosely-coupled services by definition does not necessarily offer integrated functionality. Loosely coupled therefore does not mean interoperable. On the contrary, because of their independency, loosely-coupled services are often heterogeneous and they are not a priori known to interoperate at other levels than the syntactic communication agreements assigned by the composition framework [2]. Other consequences of loosely-coupled architectures are that:

- services may need other services, which may not be available at run-time, and
- loose-coupling has a run-time overhead, reducing application performance.

2.2 Interoperability and heterogeneity

There is an important distinction to make between web services as a paradigm and actual instances of web services. The web services paradigm provides a framework for the creation of web services as building blocks of one or more modular process environments. This framework provides a common syntax on which web services base their communication. Nevertheless, the individuals of an arbitrary set S of web service instances do not necessarily behave as interoperable services within S, because they may not have been intended to interoperate within this set. This is due to the fact that their way of communication may not have a common structure and semantics.

Interoperability is one of the key conditions for system integration and service chaining, i.e., matching of service requests and service advertisements. The following definition is based on [41]:

Term 2.13 Interoperability *is the ability of two software artefacts (e.g., services) to interact effectively at run-time to achieve shared goals, e.g., a joint activity. The interoperation of two software artefacts X and Y requires that X can send requests R to Y based on a mutual understanding of R by X and Y, and Y can return responses S to X based on a mutual understanding of S as (respectively) responses to R by X and Y.*

Interoperability problems emerge when there exists heterogeneity in information (represented by data sets or services). Heterogeneity can be characterised

[2]Levels of interoperability are defined in Section 3.1

by the conflicts that occur when two resources (data sets and/or services) are combined. These conflicts play a role on different abstraction levels. Information heterogeneity can occur on three levels: syntax, structure and semantics [257]. Interoperability can be achieved at each of these three levels when it resolves the heterogeneity issues at that level.

Some of the potential conflicts between heterogeneous resources are highlighted with help of an example, provided in Table 2.1.

Name	Lusaka
Country	Zambia
Location	28.2 ,-15.3
Population	2 million
Capital	yes
Access code	+260 1

Table 2.1: A representation of the city of Lusaka.

At the syntax level, two resources may have conflicts on data encodings. In the example of Table 2.1, a representation of the city of Lusaka may use a data serialisation for the attribute *location* as below:

```
(Lusaka,28.2,-15.3)
```

Another may take a form, which is perfectly valid, but incompatible with the previous:

```
(Lusaka -15.3 28.2)
```

The identification of the structure level is important in helping to understand the essence of a heterogeneity issue and to create a separation of concerns in geo-information and geo-service descriptions. At the structure level, the following conflicts can occur:

- Different data types are used for the same attribute of a representation. For example, the 'capital' attribute may be represented by a string or a boolean type. The 'location' attribute may be represented by a point datatype or a polygon datatype.
- A characteristic may take a single attribute in representation R and multiple attributes in representation S. For example, an access code may be one value in R and may be split into a country and area code in S.
- Attributes that appear in one resource are simply missing in the other.

Heterogeneity on the semantic level involves conflicts on the intended meaning of different resources. Semantic heterogeneity refers to the differences in interpretation of real-world phenomena and is believed to cause serious problems in communication processes [31]. Semantic heterogeneity can involve cognitive heterogeneity and naming heterogeneity [30, 174]. These terms are also called homonymy and synonymy respectively [257].

Here a distinction is made between a real-world phenomenon, its conceptualisation (i.e., in the mind of a human) and the name that is attached to the phenomenon by a human.

Cognitive heterogeneity involves situations where the same name refers to different conceptualisations of a real-world phenomenon (e.g., a 'house' in the mind of a postman versus a 'house' in the mind of a house agent). Naming heterogeneity exists when different names refer to a single conceptualisation (e.g., 'railroad' and 'railway'). These types of conflicts may often be resolved by one-to-one concept mappings. More complex are heterogeneities where these simple mappings cannot be used and where semantic structures have to be taken into account [257]. A problem of this kind is relating the location of the Lusaka Holiday-Inn hotel (28.195,-15.303) to the representation of Lusaka as in Table 2.1. This conflict of accuracy has to be resolved with a model that makes the semantics of geometries and their relation types explicit. Semantic structures may be divided into information structures and process structures. The first allows to compare the information concepts represented by data sets and process input/outputs. The latter facilitates the comparison of process models, i.e., how the input information is used and how the output information is produced.

Semantic interoperability The modelling of semantic structures may solve semantic heterogeneity issues and consequently contribute to the achievement of semantic interoperability between information and processes. Semantic interoperability involves the mutual understanding of the *context* of the requests and responses as mentioned in Term 2.13.

Quantification of interoperability An intriguing question is whether interoperability can be quantified. Goodchild et al.[93] implies that interoperability might be evaluated by 'ease of use', represented by the amount of training needed to accomplish a certain task, or by the number of user actions required. Other suitable metrics might be based on the transferability of knowledge, measuring the effort required by someone trained on System A to achieve the same productivity on System B. More conceptual approaches make use of semantic distance similar to the methods mentioned under semantic matchmaking. Ke-Thia Yao et al.[291] reports on the implementation of interoperability gauges that use semantic distance based on ontological relations.

2.3 Overcoming heterogeneity by contract

Interoperability can only be achieved if heterogeneity issues of the interacting systems, are well understood. They can be overcome by system communication

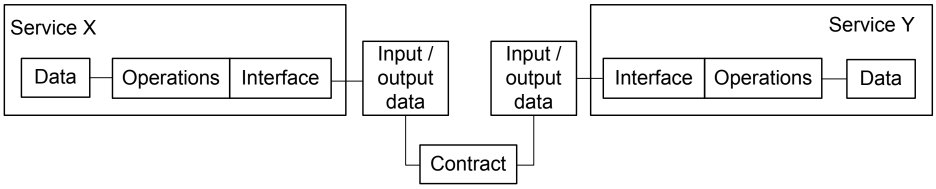


Figure 2.4: Data interchange by transfer.

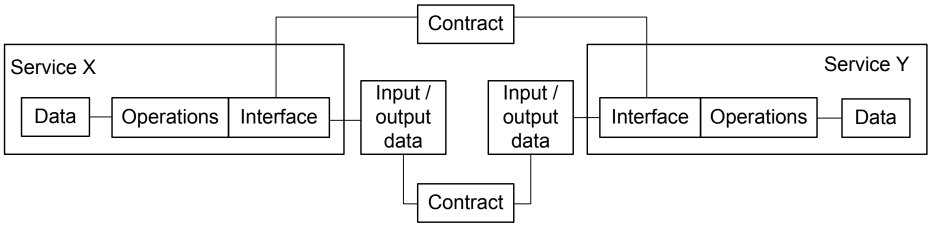


Figure 2.5: Data interchange by transaction.

agreements. The key to overcoming heterogeneity (and so achieving interoperability) is to establish contracts that specify a set of agreed communalities between systems [272].

Data interchange is basically done in two ways, by *transfer* and by *transaction* [131]. The first is the more traditional way of transferring data sets in which a contract is signed between systems, specifying communalities of their input/output data (see Figure 2.4).

Data interchange by transaction refers to data exchange between systems in which a contract is signed by the systems involved, specifying communalities of their input/output data *and* a common interface to be used by these systems (see Figure 2.5). This also requires the service to offer specific functionality. An example is the Web Map Service (WMS) specification by OGC [211], which describes the interface parameters of three operations that are made available in a WMS, i.e., *GetCapabilites*, *GetMap* and *GetFeatureInfo*.

Data can involve feature data as well as metadata. Metadata describes the characteristics of systems, such as their feature data, operations and/or interfaces. Those descriptions may contain references to contracts that the systems commit themselves to.

Contracts can be characterised by the following aspects:

- Abstraction level. Contracts about data may specify agreements at different abstraction levels. A distinction can be made between the levels at which heterogeneity is defined (see Section 2.2): syntactic, structural and semantic level.

- Contract formalisation and representation. A contract may be informally described (e.g. in loose terms in natural language) or more formally in a conceptual model with formalised relationships between concepts. Representations of contracts may take different forms, such as text documents or machine-accessible data structures, such as ontologies.

  Contracts may define new formalisations, but may also contain references to formalisations specified in other contracts.

- Contract scope

  - Bi-lateral (contracts between two systems)
  - Multi-lateral (contracts between more than two systems)

  Bi-lateral contracts require a pair of translators for each contract. An example is the conversion of ESRI's shapefiles to DXF, and vice versa. When more systems are involved, it requires $n(n-1)$ translators. Multi-lateral contracting involves compliancy with one common model. The number of required translators is reduced to $2n$ [272].

- Contract detail (the detail of the information models and process models, that systems must comply to; see Chapter 3).

An interoperability standard can be seen as a set of multi-lateral contracts. Standards that support geo-information interoperability have been developed by OGC and ISO/TC211 and involve specifications on feature data, metadata, services, etc. These standards may specify contracts of different abstraction, representation and detail. They build on top of already existing standards, such as those from ICT-mainstream technology. For example, the Open Geospatial Consortium Web Map Server specification (WMS) [206] defines contracts for (1) feature data syntax (an output map may be of the format JPEG, GIF or PNG) and structure (raster) and (2) the syntax, structure and semantics of interface request/response pairs (*GetCapabilities*, *GetMap* and *GetFeatureInfo* request/response). Syntax and structure are formally defined by means of XML application schema, semantics are informally represented. An overview on essential interoperability efforts is given in Section 2.4.

Today, we observe that an important number of services are not interoperable with services with which they could form valuable service chains. This is mainly caused by the fact that contracts are often only bi-lateral, if they exist at all.

In initiatives to establish geo-information infrastructures (GII), a cohesive set of standards is adopted in order to achieve a necessary and sufficient level of interoperability. A current trend is to aim at standards that support the paradigm of data exchange by transaction, because they provide a higher level of interoperability. Examples can be found at national scale [48, 102] and international scale [70, 286].

2.4 Interoperability models

Past and ongoing efforts to create interoperable software systems have led to models that support the resolving of syntactic, structural and semantic heterogeneity in data, service interfaces and metamodels describing them. The following subsections provide relevant models and model-providing initiatives for our objective of geo-service interoperability. The order of the sections is from basic ICT-generic building blocks to specific geo-information models. It is common practice that the specific ones are built on top of the generic ones. In some cases also cross-fertilisation takes place such as between web service models and the Semantic Web. For most of the described models, the World Wide Web (WWW) serves as a backbone. For a description of the web, the reader is referred to the numerous resources available, e.g., [26, 60].

UML Many interoperability models use the Unified Modelling Language (UML) for providing a conceptual view of the model. UML allows to visually characterise concepts and the relationships between them in an object-oriented manner. Although UML originally only offered informal modelling constructs, a new version UML 2 [202] has adopted formal semantics in its metamodel. In addition, UML provides the Object Constraint Language (OCL) with which extra information can be added to the elements of the UML model, in the form of constraints [11].

2.4.1 Distributed computing platforms

The need to distribute software applications between geographic locations within and between organisations and the need to perform integral computing over them has resulted in software technologies that support the building of distributed computing platforms. Prominent examples are OMG's CORBA [276], Microsoft's DCOM [186], and SUN's Enterprise Java Beans [220]. Several research efforts in this area have converged in the concept of the Grid. Grid technologies support flexible, secure, coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organisations [80]. The Grid was originally focussed on sharing computational power and resources for advanced science and engineering, but the Grid community is now actively developing fundamental mechanisms for the interaction of any kind of resources through a service oriented approach [90]. Requirements for the so called Grid services are beyond web services: they should support dynamic, volatile, ad-hoc, large and long-lived computing environments. As a consequence, issues on reliability, performance and security should be addressed. Recently the Grid and the Semantic Web communities have begun to cooperate. A prominent example of Grid implementation is the Earth System Grid (ESG) [79, 81]. ESG couples earth system models for the purpose of global climate research. ESG has established GRID connections between servers at seven computer centres across the USA. Services of ESG aim at data retrieval and on-line data processing and analysis.

Increasingly, the importance of web-based applications have stimulated the use of information carriers that work well with internet protocols. W3C's Extensible Markup Language (XML) [282] has acquired a key role in today's distributed applications, such as the ones that use web services. This has the following reasons:

- XML provides a way to encode both structure and content of data. Its tag naming allows for self-describing information.
- XML works well with internet applications.
- XML is flexible (its extensibility facilitates tailor-made languages).
- XML code is human-readable.
- XML namespaces facilitate unique global identifiers of information contained in the XML document.

XML is used for the encoding of domain specific data, service requests and responses, as well as for the description of services, workflow, etc.

2.4.2 ISO RM-ODP

The ISO/IEC 10746 Reference Model for Open Distributed Processing (RM-ODP) is a standard developed to support the interconnection of distributed information processing systems [126]. It is an architecture that provides a framework for the specification of ODP systems. Amongst others, methods are prescribed for (1) the specification of a complete system in terms of five viewpoints: enterprise viewpoint, information viewpoint, computational viewpoint, engineering viewpoint and technology viewpoint and (2) the definition of a set of common functions that are provided through services within the distributed processing system. One prominent function is service trading [269], which is often referred to as the *publish-find-bind* paradigm [204]. Figure 2.6 shows the actors and actions in a web service environment that uses this paradigm. The service provider publishes his service into a registry of service descriptions. The service requester searches (and finds if available) a service in the registry and uses it by binding to the service provider. Standard protocols such as WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery and Integration) are used to support these actions. During the use of the service, service requests and responses are communicated through SOAP (Simple Object Access Protocol).

2.4.3 Web services

The paradigm of web services facilitates the building of functional component chains based on XML. A definition of a web service is given in Section 2.1.1. Businesses and individuals show a growing tendency of interacting dynamically with data and services on the web and this explains the reason why the number of web

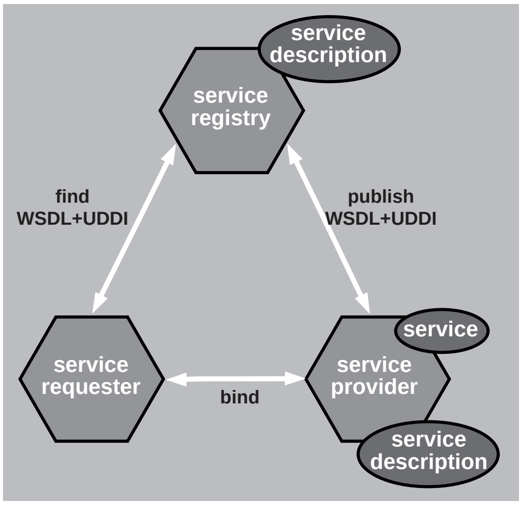


Figure 2.6: Publish-find-bind paradigm, taken from [96].

services is growing rapidly. The exploitation of web services follows the publish-find-bind paradigm [96] (also called Service-Oriented Architecture, SOA) specified in the Reference Model of Open Distributed Processing (RM-ODP, ISO/IEC 10746, see Section 2.4.2). Increasingly, effort is dedicated to the definition of web services workflow languages that support the execution of service chains. Interoperability will surely prove to be the critical success factor in web services proliferation [293].

Current research topics cover issues like modelling, development, deployment, publishing, as well as discovery, composition, collaboration, monitoring and analytical control.

Attempts to ensure semantic interoperability between web services embark upon semantic annotation and the development of Semantic Web-enabled Web Services (SWWS) [44] (see also Section 4.7).

2.4.4 Semantic Web

The Semantic Web [27] is the next step in the evolution of the World Wide Web into a network of documents and services that are provided with a well-defined meaning. This is established by documenting relationships between web resources (which could be referred to as meta-modelling), allowing for better resource discovery results and better cooperation between computer applications. Resources can be classic text documents, domain specific data (such as geo-data), software applications, etc. A key role is played by ontologies. In the Semantic Web context, these are machine accessible representations of knowledge that are used for inferring intra- and inter-resource relationships[3]. Recent work within the Semantic Web community has resulted in W3C recommendations of two knowledge representation languages, i.e., the Resource Description Framework (RDF) [23] and the Web Ontology Language (OWL) [184]. OWL draws upon the formal theory of Description Logics, which has roots in first-order predicate logic and provides highly expressive concept-forming constructs [14]. See Section 4.2 for OWL's backgrounds and principles. Currently, main activities of the Semantic Web initiative include:

- Development of ontology representation languages such as OWL
- Development of tools for ontology building and ontology mapping
- Semantic markup of resources, including web services
- Development of knowledge-based reasoners
- Application and demonstration development for information retrieval and reasoning

The most prominent application domains of the Semantic Web are currently bio-informatics as well as medical applications and storage and retrieval of document and multi-media resources. GIS is an application domain that is not (yet) at the forefront of Semantic Web applications, but geographic examples and earth science applications are already referred to in a few presentations at the leading Semantic Web conference [72]. Related to the Semantic Web ideas is the paradigm of topic maps. Topic maps constitute a modelling approach for the semantic annotation of WWW resources. Topic maps are very similar to RDF and have been developed more or less independently. While RDF was endorsed by the World Wide Web Consortium, the topic map standard was approved as ISO 13250 in the year 2000 [85]. Although mapping mechanisms are available between topic maps and Semantic Web sources, they are not expected to be integrated. In another arena, a cross-fertilisation initiative has been taken recently between the Semantic Web community and the Object management Group (OMG) on developing a specification for ontology engineering, the Ontology Definition Metamodel (ODM) [84]. Chapter 4 elaborates on the technical principles of the Semantic Web.

2.4.5 Spatial data interchange standards

Spatial data interchange standards define the agreements in communicating spatial data between users through software systems. This involves data format, data structure and data content (see Section 2.3). Data exchange standards are typically represented by conceptual schemas on a high abstraction level, together with

[3]In Chapter 4 a more precise definition of *ontology* is provided.

application schemas on lower levels of abstraction. They serve the purpose of creating interoperability at all levels (syntax, structure and semantics) between the resources that comply to these standards. Spatial data exchange standards typically build geo-domain specific elements on top of geo-domain independent models, adopted from ICT-mainstream technology. For example, today, XML is the defacto encoding standard for many spatial data interchange standards.

The schemas used in the ISO 19100 series of standards make use of UML schema representations for their precise definition and understanding. ISO 19109 (GFM) [131] formalises the rules for building spatial data application schemas. The standard distinguishes between the conventional data interchange by *transfer* and the increasingly popular data interchange by *transaction* (see Section 2.3). The first implies that the application schema holds information about data structure and data content of the data set. Examples are SDTS (USA), Interlis (Switzerland), ATLIS(Germany) and NEN1878/NEN3610 (The Netherlands). In the second case, application schemas also hold information on a common interface to be used by the interchanging applications. The development of GII, such as the European INSPIRE [70], is necessitating existing spatial data transfer standards to be harmonised or revised, based on modern international standards, developed by ISO. An example implementation of the ISO 19109 (GFM) standard is the new Dutch data exchange standard NEN3610 (version 2) which serves as a data model for geo-information exchange in The Netherlands [195]. It implements ISO 19109 (GFM) by making use of a general feature model, a feature catalogue model and adopts OGC's Geography Markup Language (GML) as interchange standard. A feature catalogue defines feature classes and their attributes. ISO has standardised the method of feature cataloguing in [133]. The feature catalogue of NEN3610 (version 2) defines a feature on a high abstraction level by describing properties such as location and geometry. In this way NEN3610 forms the common basis for more specific geo-information models used in topographic mapping, land-use planning, water management, etc. Feature catalogues are built from domain terminology classifications. A well known example of such a system is the CORINE land cover classification [36].

2.4.6 Open Geospatial Consortium (OGC) specifications

The Open Geospatial Consortium (OGC) is a consortium of commercial, governmental and academic organisations that builds GI related specifications on top of ICT mainstream technologies. OGC's mission is to foster the interoperability between geographic information systems. The work of OGC centres around the delivery of spatial interface and encoding specifications for processes and data. In this way the interfaces of services and client applications are standardised. The publicly available OGC specifications can be implemented by system developers in order to make their proprietary systems interoperable with others. OGC's standards development takes place in consensus based processes, commonly through interoperability testbeds. From its birth in 1994, OGC has gradually proceeded from building foundations such as feature and geometry models to building service models. OGC's data and process modelling is based on hierarchically related information modelling constructs. The generic basis is formed by abstract specifications with topics covering models for spatial features, services, metadata, etc. At the implementation level, modelling languages are used to represent the syntax, structure and semantics of geospatial and geoprocessing-related information resources. An example of such a language is the Geography Markup Language (GML) [210]. OGC's specifications cover a wide range of issues, basically involving services, data and metadata. This may involve also specific types of services, such as the OpenGIS Location Service (OpenLS) implementation for mobile applications. OGC uses wherever possible mainstream ICT of which UML and XML are clear examples. In addition, OGC's specifications are harmonised with the ISO 19100 standards series under development by ISO/TC211 (International Organization for Standardization, Technical Committee 211, Geographic information/Geomatics). OGC's specifications cover both generic and domain specific topics.

Service model RM-ODP has been applied in the OpenGIS Reference Model [207] and consequently OGC adopts a service oriented architecture. In this architecture, services act either on data (e.g., processing services, portrayal services), on metadata (e.g., catalog services) or on services (e.g., workflow services). Metadata describes either data or services. Well known OGC data service specifications are the Web Map Service (WMS) [211] and the Web Feature Service (WFS) [218]. These services have well defined interfaces that support discovery actions through a GetCapabilities operation and several standardised operations for data retrieval. Within the more recent OGC Web Services Common Specification (OWS) efforts [215], these services are aligned with the mainstream publish-find-bind paradigm represented with SOAP, WSDL and UDDI, however without service chaining support. A taxonomy of geo-services has been initiated under OGC abstract specification Topic 12 OpenGIS Service Architecture [205], which has adopted the ISO 19119 (Services) standard on *Geographic information - Services* [132]. ISO 19119 (Services) describes Geo services in the five viewpoints of RM-ODP and contains a specification for service metadata.

Spatial operations Spatial operations as part of spatial analysis services have been included as topology operators in the filter encoding specification [214]. Filter encoding appears in several OGC specifications, such as WFS and Simple Feature Specification (SFS) for SQL [127]. An extension to the filter encoding that allows for the creation of new geometries, has been proposed by the Web Spatial Analysis Service (WSAS) Implementation Specification [212].

In a more recent effort, the Web Processing Service (WPS) specification provides access to spatial operations, ranging from simple calculations to complex models. The WPS interface exposes the parameters for data input, operation initialisation and data output. This is done with three WPS operations: GetCapabilities, DescribeProcess and Execute. The final report of the experiment [219] concludes that WPS is able to handle different kind of data and processes, but also identifies problems with aligning to other standards and limitations of service discovery, which relies on text matching between keywords and service descriptions.

Service discovery In OGC context, service discovery is handled by a service registry that provides service metadata with details on service types, as defined in ISO 19119 (Services) [205]. Service metadata can be queried and managed with catalog services [209]. The OGC Web Services Common Specification (OWS) provides mechanisms within the publish-find-bind paradigm of RM-ODP. As today there is no OGC specification that deals with semantics in support of service (and data) discovery. An attempt is has been recently undertaken in the OGC Geo Semantic Web Interoperability Experiment (GSW IE) [169, 213].

2.4.7 ISO 19100 standards

In 1994, ISO Technical Committee 211 (ISO/TC211) commenced its work on establishing a set of standards for Geographic information / Geomatics. Since then, ISO/TC211 has integrated standardisation initiatives on geo-information worldwide. The ISO 19100 series of standards, being developed by ISO/TC211, entail the handling of geographic data, including management, acquisition, processing, analysis, access, presentation and transfer [151]. Due to a common interest, the paths of ISO/TC211 and OGC often meet. The work of ISO/TC211 and OGC has been streamlined over the years. The focus of ISO/TC211 has been directed towards generic specifications and OGC has focussed on implementation specifications and interoperability test beds and projects. ISO/TC211 and OGC have agreed on the adoption of certain OGC specifications as ISO standards (e.g., the OGC WMS specification as ISO 19128 (WMS) and OGC's GML as ISO 19136 (GML)). An important basis in the 19100 series of standards is formed by the General Feature Model (appearing in ISO 19109 (GFM)), a metamodel for geographic features and their properties. A number of standards are still under development. An overview of the work of ISO/TC211 and the ISO 19100 standards can be found in [151] and on-line at http://www.isotc211.org/. A list of standards that make up the 19100 family can be found with their identification numbers and their titles in Appendix B. Note: References to ISO standards in this thesis are indicated by their number and a short name between brackets, as listed in Appendix B. When a reference is made to all the standards of the ISO 19100 family, the number '19100' is used.

2.5 Geo-services

Geographic information systems are gradually shifting from monolithic systems to service-oriented ones. Geo-services are defined as services that involve data (input, output or tightly-coupled data) with a geographic reference. Today's GIS products can be classified as design-time or run-time. Table 2.2 gives an overview of some of these products. ESRI's ArcObjects, ESRI's Geoprocessing toolbox and Intergraph's Geomedia Objects are examples of commercial, design-time based, software suites. They contain object-oriented software constructs for spatial database creation, spatial data manipulation and the building of end user interfaces. The combination of ESRI's Geoprocessing toolbox and their Modelbuilder also allows users to create service chains, visualise and store them in a script.

Other design-time based products are developed in an open source environment such as GRASS [99], GDAL [86] and Geotools) [89]. Each provide a publicly available application programming interface (API) that allows application builders to use them within their programming environment.

Run-time based applications are typically implemented as client/server applications. There is an increasing number of products facilitating the building of server and client components that can be used in service oriented applications. Commercial examples are ESRI's ArcWeb services and Microsoft's Map point web service. They use proprietary interfaces.

An increasing number of products, both design-time and run-time-based, are implementing OGC interface specifications (e.g. for feature access, web mapping, etc.). Web applications built with these products are considered to be loosely-coupled, compared to the proprietary examples mentioned above.

The above products can be used to build repositories of geo-services and off-the-shelf geo-software components that can be used to build applications. The majority of such repositories are used within geo-information infrastructures (GII) and are typically made available through web portals [47, 68] or through registries that allow for direct interface access [89, 200, 253].

Geo-service classification The growing number of service-oriented implementations on the GIS market, necessitates the distinction of their purpose, in order to select the best for the intended application. Several efforts have proposed classifications of GIS operations in terms of their functionality [7, 92]. Such taxonomies can also be found in many GIS text books, of which good examples are [50] and [275]. They are useful for describing primitive actions of geo-services. At a higher abstraction level, ISO 19119 (Services) [132] classifies geodata-processing services and services that are needed to find and create other services, such as catalog services and services for managing service chains.

To the best of our knowledge, GIS operation taxonomies have not yet been used in a formalised framework that supports reasoning for integral analysis of the functionality of geo-services. A more elaborated section on geo-operation classifications, as a basis for such reasoning, can be found in Section 5.4. For now, we can

Software product	Open Source	OGC based	Coupling	Information exchange
ArcObjects	No	No	Design-time	Proprietary objects
Geomedia Objects	No	No	Design-time	Proprietary objects
ArcWeb Services	No	No	Run-time	SOAP
ArcIMS	No	No	Run-time	ArcXML
ArcIMS + WMS connector	No	Yes	Run-time	OGC WMS interface
Microsoft Map-Point web service	No	No	Run-time	SOAP
Minnesota Mapserver	Yes	Yes	Run-time	OGC WMS interface
GDAL	Yes	Partly	Design-time	GDAL/OGR API
GRASS	Yes	Partly	Design-time	GDAL/OGR API
GeoTools	Yes	Partly	Design-time	GeoAPI (OGC)
OGC OWS compliant products	Yes/No	Yes	Run-time	SOAP

Table 2.2: GI software product examples classified according to aspects of interoperability.

confine to a rough classification as in Table 2.3. In the light of the classification axes, presented in Section 2.1.2, this classification identifies *application domains* within geo-information handling.

On the top level we distinguish between services that support human interaction and services acting upon artefacts: feature model, feature, service and meta-information. Meta-information can involve information about data or services. In an end-user application, human interaction services are typically chained to services of the other categories. An example of a human interaction service is a map viewer service typically used for map displays, such as panning, zooming, etc., that acts between the feature access service and a thin client such as a web browser (a *three* tier client/server setup). Figure 2.7 shows an example of such an intermediate map viewer service. In a *two* tier setup, a thicker client that contains zooming and panning functionality is used to communicate directly to the feature access service.

The service classification that is proposed here, only partly follows the ISO 19119 (Services) classification of services. Sections 6.3 and 6.5 clarify that the proposed classification provides semantics that better support service discovery and service chaining. It also holds more disjoint service sub types and allows further disjoint sub typing in a taxonomic structure.

Service type		Service sub type	Examples
Human interaction	**(artefact it acts on)**		
	Feature model	Feature modelling	Services for modelling feature types and feature type properties (including associations between feature types)
	Feature	Feature access (acquisition, storage and exchange)	Sensor service, digitising service, positioning service, format conversion service
		Feature processing and analysis	Overlay service, buffer service, generalisation service, gazetteer service
		Feature presentation manipulation	Style and cartographic symbol editors
	Service	Service creation	Chain validation service, documentation service
		Service execution	Chain execution service, Grid service (e.g., server allocation)
	Meta-information	Meta-information creation and storage management	Annotation service, Publishing service as part of catalogue/registry service
		Meta-information processing	Discovery service as part of catalogue/registry service
		Meta-information presentation manipulation	Style editing service as part of catalogue/registry service

Table 2.3: Geo-service classification.

2.6 Geo-service use cases

In order to demonstrate semantic modelling and its research issues, a number of use cases have been defined. A short description is given below. The uses cases and related experiments are worked out in chapter 7 All use cases have the common goal of data/operation integration. The use cases are the following:

- *Riskmap NL* is project-oriented and focuses on the integration of information models for integrating data-providing services for risk mapping.
- *Emergency 112* focuses on ad-hoc integration of data sources in a disaster emergency situation.
- *Research Net* is project-oriented and aims at the creation and reuse of meta-information for data and operations in scientific research.
- *Travel Google* focuses on ad-hoc integration of travel data (e.g. hotel locations) and operations (e.g. route planners).

Table 2.4 provides a comparison of the different contexts for the use cases, referring to some of the service characteristics given in section 2.1.1. The fields of the table are interpreted as follows.

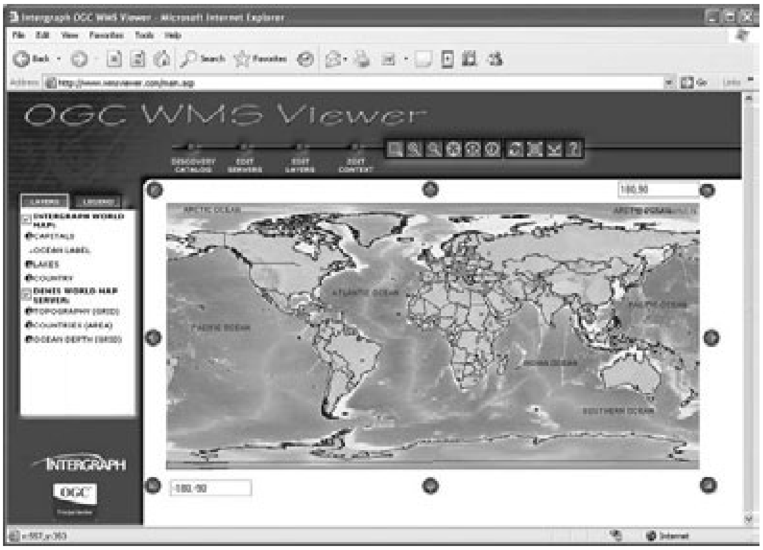


Figure 2.7: Map viewer service, running in a web browser. The service allows to display and interact with multiple OGC Web Map Services.

- The use cases focus either on pursuing interoperability of data only, or interoperability of both data and operations. Typically this involves data-providing services and data-processing services respectively.
- 'Composition dynamics' are expressed as *project-oriented* or *ad-hoc*, indicating the time for the interoperation effort is typically months, respectively a few hours.
- 'Service provision' gives an indication on the heterogeneity and the dynamics of the services that are to be composed.
- The 'focus of the interoperation effort' indicates on which aspect of service (data and/or operation) integration the use case concentrates on: service model integration, service description integration, service discovery, service composition or service chain execution.

2.6.1 Riskmap NL

In response to two severe incidents in The Netherlands (fireworks disaster in Enschede in 2000 and cafe fire in Volendam 2001) the Dutch government has decided to provide more insight into potential hazards to civilians and disseminate this information amongst governmental agencies. This has resulted in an initiative to

	Riskmap NL	Emergency 112	Research Net	Travel Google
Data / operation focus	Data / operations	Data	Data / operations	Data / operations
Composition dynamics	Project-oriented	Ad-hoc, urgent	Project-oriented	Ad-hoc
Service provision	Few sources, infrequent changes	Many sources, frequent changes	Few sources, infrequent changes	Many sources, frequent changes
Focus of interoperation effort	Integration of information models of different domains	Fast integration of data descriptions	Reuse of remote services / service chains	Discovery of remote services, not necessarily to form a chain

Table 2.4: Context comparison of use cases.

produce interactive web based provincial risk maps. So far, nine out of twelve provincial maps are on-line. Despite the template that is used, they differ significantly in accuracy, categorisation and visualisation. Up to this stage, the use case is based on facts. From hereon a fictive scenario is followed. A team of GI engineers is instructed to integrate geo-information for risk maps and make these risk maps interoperable with other data/services. The Riskmap team decides to do this through the integration of ontologies, representing different data models such as NEN3610 (see Section 2.4.5), TOP10NL (the object oriented model of the Dutch Topographic Service) [148], and a hazard domain model (called 'Riskmap'). This integrated model is to be used by several future applications, such as map viewing, planning and emergency response activities. Oscar is a member of the Riskmap team, responsible for a feasibility study. He proceeds as follows. With Protégé (an ontology editor) he builds the ontologies, based on documentation of NEN3610, TOP10NL and the current provincial risk maps. He creates a mapping between the ontologies by establishing relationships between similar concepts. Then, Oscar applies software-based reasoning to (1) discover service extensions for the risk map functionality and (2) perform integrated analysis over the three information models.

2.6.2 Emergency 112

The following use case is based on a fictive scenario described in [102]. The overall scenario is as follows. Severe weather conditions cause a flooding thread in the east of the Netherlands. The regional water management control centre identifies a crisis situation and demands to stop all river traffic. In this phase a severe ship accident takes place in which one ship damages a dike and another ship, loaded with ammonia, is damaged. The latter is feared to explode and to cause a gas plume. Evacuation of the area may be necessary. The remainder of this use case description is an addition to the scenario as described in [102]. During the emergency phase, data is coming in from different data sources (e.g. from governmental agencies and through field observers) at a high pace. The support of machine reasoning is called to be able to quickly integrate the information and, at the same time, keep it consistent with the already available information. An ontology is used to register all the information that is coming in (e.g. through field observers) and to disseminate subsets of information to crisis management units and the public/press. Kora is an information engineer at the emergency room and registers all incoming information in an Emergency Information System (EMIS).

2.6.3 Research Net

This uses case centres around a platform for sharing GI services, called Research Net, a fictive research environment with semantics-based service discovery possibilities to support service chaining.

The use case involves the publishing of services (phase 1) and the discovery of services (phase 2). It demonstrates the principle of *lineage* (the history of the processing of a particular geo-information source).

Phase 1 Jody is a researcher at the International Institute for Geo-information and Earth Sciences (ITC). She wants to evaluate the extent of flooding in the confluence area of the Ganges and Jahmuna rivers in Bangladesh. The following scenario is based on [175]. For the analysis she wants to use three satellite images of different periods: during the dry season, during a moderately severe flood and during a severe flood. First, she is going to search for the satellite image on the web. She knows SPOT images are useful because they make distinction between land and water possible. After finding and retrieving the satellite images Jody follows the analysis procedure as below:

1. Apply band rationing
2. Apply slicing
3. Combine sliced maps
4. Classify

Jody decides to make the analysis available on the web in different ways:

1. **Option 1:** As a map showing the end result, together with ontology based metadata containing lineage information
2. **Option 2:** A link to the raw satellite images and a composite web service that performs the analysis

3. **Option 3:** A link to the raw satellite images, separate web services of each analysis step and a service chain description that can be used to obtain the end result

Phase 2 Jeff is a researcher at the World Resources Institute (WRI) and needs to perform an analysis, similar to the one by Jody in phase 1, but in another geographic area. Jeff searches for the available services on the computer network (amongst them those of Jody) and tries to integrate them on his PC.

2.6.4 Travel Google

On a business trip, Eddie is going to an OGC Technical Committee meeting in the USA. As part of the preparations for the trip he looks for data of the destinations' surroundings, such as the transport facilities and restaurants near his hotel and the conference centre. Supported by his basic GI and ICT knowledge, Eddie searches with a search engine for travel domain ontologies. He then searches for services that are described with such ontology. Amongst those, he selects the ones that provide spatial analysis functionality on hotels, restaurants and public transport.

2.7 Summary and reflection

This chapter has highlighted the prerequisites for on demand geo-processing. Generally speaking, on demand processing requires services to be available at the time a task is being executed. To support a variety of tasks, these services have to be generic (in contrast to task-specific), which may be achieved by introducing modularity of service functionality. What is a good 'size' of software modules depends for an important part on the demand for reusability. This chapter has explained how distributed services can form one virtual information system. Several standardisation efforts in information technology have facilitated the building of interoperable software systems by means of contracts and overcome basic problems of heterogeneity. This involves mainly 'low-level' syntactic and structural interoperability solutions that can be reused by domain specific standards. Specific interoperability issues in GIS are being addressed by OGC and ISO TC/211. Foundational standards for generic service communication and more specifically, geo-feature modelling are in place. In addition, modern spatial data interchange standards are built up hierarchically, providing flexibility for specifying sub-domain interoperability models.

A problem that still hampers the interoperability between services is caused by the incomplete realisation of contracts. This is rather an implementation issue than a design issue. For example, according to a survey of 1200 web services in the year 2003 [143], 67% of the surveyed WSDL files were invalid and 70% were incomplete. Another example is given by the mismatches that exist between implementations of OGC's GML specification, as reported in [87].

Another issue is that most of the standards do not provide computer representations which can be ‘understood’ by machines. This is particularly necessary if we want to carry out semi-automatic processing of information sources, taking into account the semantics of interoperability. Meta-information that sufficiently describes service functionality is essential in this case. Such meta-information should be structured in a way that is as close as possible to the elements (e.g., input/output parameters, tightly-coupled data) of services and service compositions. In addition, a classification of functionality helps in structuring service characteristics. In case of loosely-coupled systems, such meta-information should support each service to be self-descriptive. The structuring of the service meta-information is the subject of the next chapter. The use cases, described in Section 2.6 form the basic context in which the modelling in the rest of this thesis has to be placed.

Chapter 3

Service models for discovery, composition and execution

To evaluate a service's fitness for use and to create meaningful combinations of services, it is necessary to model the essential properties of services that facilitate their discovery, composition and execution. Services can be characterised by the information they deal with and how they process this information. The abstraction of information and processes is an important instrument to simplify the view on the often complex structure of a service [171].

In this chapter, a discussion is provided on the use of abstraction models that allow for machine based reasoning with service metadata. This chapter is organised as follows. Section 3.1 describes an abstraction model for geo-information, which forms the basis for conceptualising geo-information in Chapter 5. Section 3.2 provides background information on the modelling of processes, which is important to understand the composition of services. Section 3.3 describes service chaining as the combination of discovery, composition and execution.

3.1 Information modelling

The unambiguous integration of information requires the resolving of data heterogeneity between information resources, syntactically, structurally and semantically. Heterogeneity issues can be better resolved when the semantics of potential underlying data models are made explicit. For this the use of publicly available common data models within a user community is considered to be an important asset. This section lays a foundation for the data model that is introduced in Chapter 5.

3.1.1 A layered approach

Information systems deal with different types of artefacts to represent real-world phenomena. An artefact can be a drawing, a database record, an XML snippet, etc. The intended meanings of these artefacts differ with the context (e.g., application domain, data structure context) in which they are used. The comparison of artefacts should be done in the same context, in order to avoid the misinterpretation of information. In information exchange the sender and receiver agree on the context in which the artefacts are used. However, such contexts need to be specified and they can be complex. In order to disentangle the contextual elements, one often breaks down complexity by a separation of concerns. This can be done along several axes. Two are distinguished here: information abstraction and application domain abstraction.

Information abstraction Information models exist at different levels of abstraction. The distinction of layers for the purpose of complexity reduction has been proven useful in traditional database design, where a conceptual level, a logical level and a physical level are identified [290]. Similarly, in geo-information science, several models have been proposed, mainly driven by interoperability issues. Figure 3.1 shows a five layer abstraction model adapted from the five-universes paradigm, proposed in [76], and the nine layers of abstraction of the OGC feature model [203], [30].

The top layer contains elements of the real-world that we are living in. In this layer, the elements are as they are in the real-world, and as soon as we describe them, these are becoming abstractions of reality located in one of the layers below. In the second layer, the cognitive world, the real-world elements are captured by intuitive concepts in human thoughts which are communicated by natural language. In the third layer, the formal concepts world, the intuitive concepts are formalised. This is typically done by constructing a consensus based model of concept definitions and concept relationships, possibly supported with a domain ontology. The formalisation can be materialised in standards that are documented in text and model representations such as UML diagrams. In the fourth layer, the symbol world, formalised concepts are represented symbolically. Geographic features are represented according to the object/field model, see [290]. The fifth and lowest level is the implementation world, containing computational elements materialised as software classes and data serialisation. In this stack, the term *data set* (or data) is distinguished from *information*. Information represents a real-world phenomenon at all levels below the real-world level. *Data* resides on the lowest level only and represents all the layers above.

Application domain abstraction Along another axis, orthogonal to the information abstraction axis (that is: horizontally, within each level) different application domains can produce different abstractions. First, in different application domains the same phenomenon may be conceptualised differently. For example,

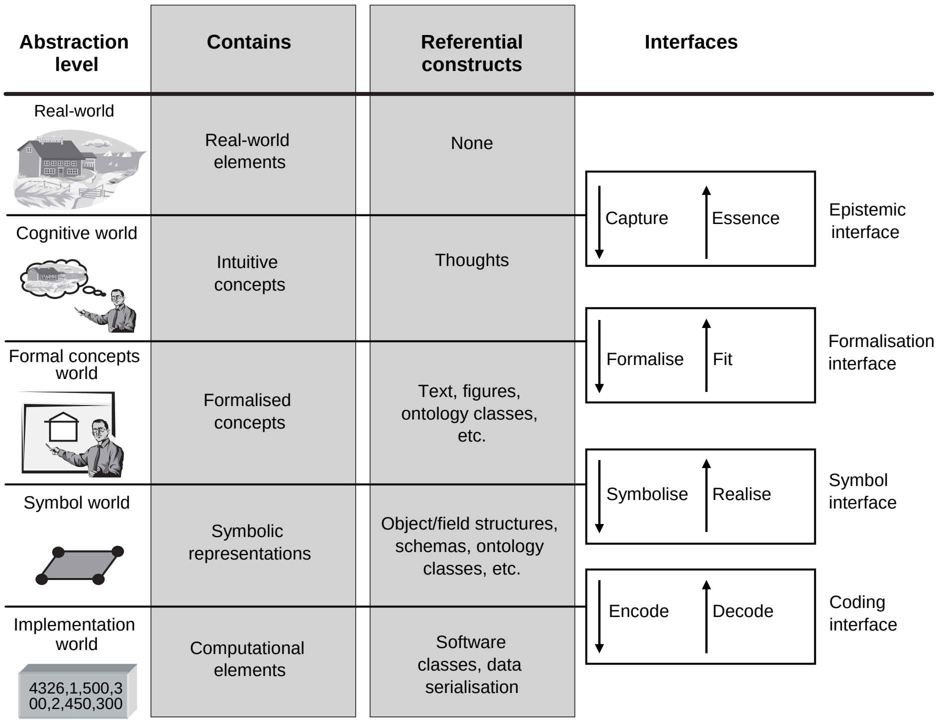


Figure 3.1: Information abstraction stack.

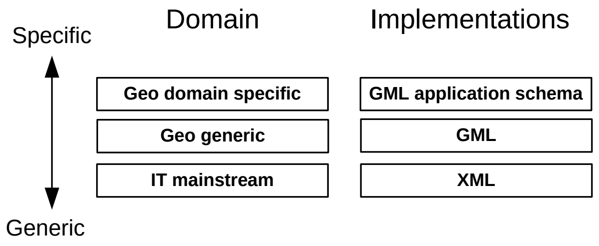


Figure 3.2: Geo-ICT modelling stack.

a theatre may be seen as a location of cultural events or as a building. Second, conceptualisations can be split according to the aspects that they model. For example, geographic phenomena (those phenomena that can be related to the earth surface) have thematic aspects, spatial aspects and temporal aspects, which are all conceptualised differently. Third, intuitive concepts may range from general (e.g., *a physical object*) to specific (e.g., *a house*). Similarly, at the implementation level, data serialisation models range from generic ones (e.g., XML) to specific ones (e.g., GML). Figure 3.2 gives an interpretation of such abstraction in Geo-ICT similar to the levels of data modelling described in [188], together with examples of XML based implementations. The layers form a stack in which each layer is a specification of the layer below it.

The layered model as interoperation stack When two systems interoperate they will exchange data at the implementation level. The data of the sending system has been abstracted from the real-world level to the implementation level, passing the intermediate levels through a sequence of steps named *capture*, *formalise*, *symbolise* and *encode* [1]. In each step, information residing on the upper layer is abstracted to the lower layer. If this is not properly done, information is lost. This information can be safeguarded by creating metadata at each layer and link it to the abstracted artefacts at the layer below. The metadata refers to the referential constructs in each layer (see figure 3.1). At the receiving system the data will pass the layers in reverse order. The metadata is used to derive the correct semantics in the layer above. The principle of passing metadata through the layers is similar to protocol stacks used in computer networks. This analogy has been identified in [185] and [56] and is considered to be valid, despite the fact that abstraction plays no role in computer networking layers. The translation steps between computer networking layers are performed by interfaces, see [261]. Artefacts at the same level, called peers, have an agreement on a protocol. Similarly, in our information abstraction paradigm, artefacts at the same level of abstraction (e.g., 'house' in one application and 'house' in another application) refer to the same

[1] The generalised notion for these steps will be denoted as 'represent'

concept in an ontology. Interfaces (see figure 3.1) are involved with the annotation and interpretation of metadata and the alignment between referential constructs. The latter is referred to as mediation [76].

The practical implication of the layers can be understood as follows. A spatial data set resides at the lowest level, but the same data set *represents* the characteristics of a real-world phenomenon at all levels. A road for example can be characterised by (1) its road class definitions such as 'primary road' (e.g., meaning a 12 metre wide road) at the formal concept level, (2) the geometry that represents the road in the database (representation level) and (3) its data elements (implementation level). However, *descriptions* of data sets do not necessarily contain information about all levels. For data processing, operations may need to 'know' the identity of the data at the lowest level (data format) before any processing can take place, but this is not necessarily so at higher levels in the stack (for example when we perform a low level merge of data sets). The degree of semantic interoperability that we can anticipate on, depends on the availability of meta information linked to the data set at those levels. This meta information is crucial for mechanisms (e.g. agents) that perform a mediation task between systems.

An interesting development is currently taking place at the implementation level. Historically, data formats do not reveal much meta information on the semantic level. However, with the advent of XML-based data set formats, such as GML, it is easier to derive higher level descriptions from the data, through text-based tags, namespaces and schema definitions.

3.1.2 Abstraction hierarchies

Abstraction is needed to simplify the models we use to describe services. Abstraction is a common technique in conceptual modelling and forms a basis for object-oriented information systems. Commonly used forms of abstraction are classification, aggregation and generalisation. These terms are defined in [35] as follows:

Term 3.1 Classification *is the grouping of entities that share common characteristics into a class over which uniform conditions hold.*

A clear distinction is made here between a class, for example *Building* and an instance, e.g., *Louvre.* The opposite of classification is called *instantiation.*

Term 3.2 Aggregation *is treating a collection of classes as a single class.*

The opposite of aggregation is called *partitioning.* Aggregation of classes can be done repeatedly. This results in an aggregation hierarchy. The levels in such an hierarchy are referred to as aggregation levels.

Term 3.3 Generalisation *is extracting from one or more given classes the description of a more general class that captures the commonalities but suppresses some of the detailed differences in the description of the given classes.*

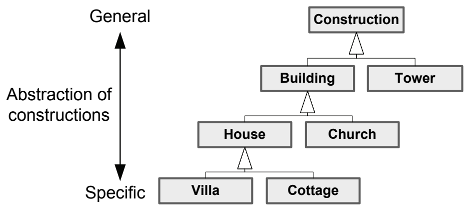


Figure 3.3: UML class diagram showing a concept generalisation hierarchy for a constructions domain.

The opposite of generalisation is called *specification.* Generalisation of classes can be done repeatedly. This results in a generalisation hierarchy. The levels in such an hierarchy are referred to as generalisation levels.

A generalisation hierarchy is also referred to as taxonomy or hyponomy and an aggregation hierarchy is also referred to as partonomy or meronomy [273]. To indicate any of the two types the term *abstraction hierarchy* is used.

3.1.3 Information abstraction

Information abstraction is needed to simplify the models we use to describe services. In abstraction of classes we can distinguish aggregation and generalisation. They can be best described with the help of abstraction hierarchies.

In service interoperation, the data is exchanged between sender and receiver passing the layers of the information abstraction stack as described in Section 3.1.1 and depicted in Figure 3.1. A more detailed view within each level of the information abstraction stack will help in understanding interoperability issues at that level. As we are interested in the semantic interoperability from the system's perspective we will focus only on two levels: the formal concept world and the symbol world.

Figure 3.3 portrays a generalisation hierarchy for constructions. Horizontal lanes in the figure represent levels of equivalent generality. The hierarchy represents a specific focus on the real-world phenomena and is therefore considered to be an example that represents only a part of all the aspects of the domain. For instance for a building, levels of generalisation can be focussed on construction aspects as well as usage aspects. In a 'usage' hierarchy we may also distinguish between 'shop' as a superconcept and 'supermarket' as a subconcept.

To understand abstraction we need also to identify aggregational abstraction, revealing the part/whole relationships in a domain. Figure 3.4 represents an example of an aggregation model of a construction domain. Horizontal lanes in the figure represent aggregation levels. Another example is the partitioning of regions in a hierarchy of country, region, province and municipality. It is the context of the application that determines up to which level (lowest and highest) an abstraction

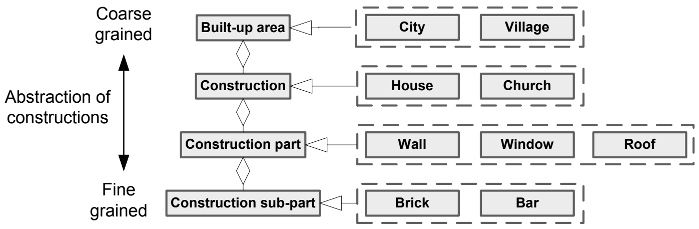


Figure 3.4: UML class diagram showing a concept aggregation hierarchy for a constructions domain. Aggregation is identified on the vertical axis; Examples of subclasses (generalisation) are provided along the horizontal axis.

should be carried out. The lowest level, relevant to the application, is called the *atomic* abstraction level, containing atomic elements.

Generalisation and aggregation hierarchies are often used together to model a domain. Due to inheritance, they can become intertwined to form complex models [187] and they have to be treated with caution. Ontologies are well suited to represent such complex conceptual domain models. Commonly the generalisation relations are modelled as super-/subconcept relations. Aggregation relations are modelled as concept roles of type 'is-part-of'. Concept roles other than this type (e.g., specific associations such as specific 'has-a' relationships) are not considered to be relevant for the notions of generalisation and aggregation. More on ontology design can be found in Chapter 4.

3.2 Process modelling

The integration of services as well as the search for a single service requires the evaluation of each service's data characteristics (input, output and tightly-coupled data) and may require the evaluation of its internal process structure (the latter exposes semantics on its functionality). Evaluation of the process structure can be performed by executing the service in pre-defined tests, but it is typically done first by interpretation of the metadata, describing its internal processes.

In order to define a framework in which one can describe software processes, the notion of software processes can be abstracted into a generalised world of dynamic systems. Numerous formalisms have been developed in the past to model behaviour in terms of events, processes and operations and the artefacts that they act upon. The formalisms differ with respect to the type of system they are able to model, the purpose of the model (e.g., software architecture design, human interaction analysis, simulation, process documentation, etc.) and the formalisms that provide the model's formal semantics.

Figure 3.5 represents the relationships between static and behavioural entities.

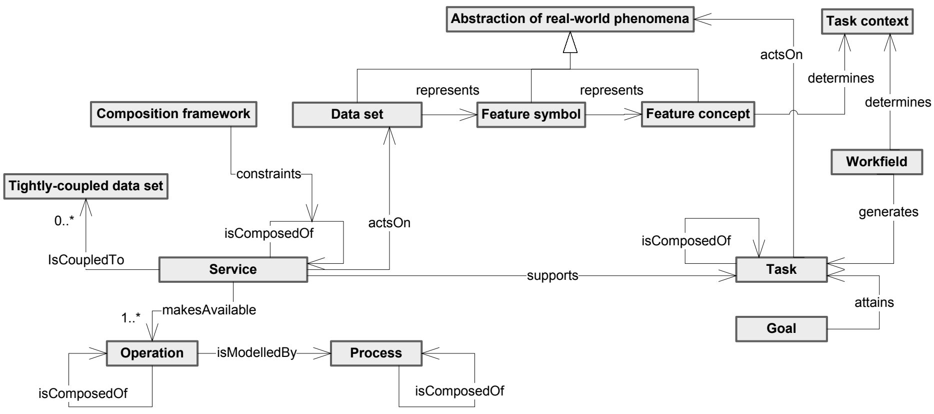


Figure 3.5: UML class diagram showing the context of static and behavioural entities in a service/task environment.

This representation is partly based on the different models presented in [170]. The static entities in the figure (data, feature symbol and feature concept) reside in respectively the implementation world, the symbol world and the formal concept world as described in figure 3.1. A service acts —at the implementation level— on the data as representation of a real-world phenomenon. The service may affect the semantics of the data, represented at higher levels in the geo-information stack. For example, an operation, overlaying a topographic map with a hazard map, may enrich house features with hazard information. Tasks can act on static entities at different levels of the information abstraction stack. For example, a task may be a travel planning task, a hazard risk assessment or the conversion of a vector data set to a raster data set. Kirwin and Ainsworth [144] coin a task as the attempt to attain a goal in a particular context. *Goals* are defined as desired states of systems under control or supervision, for example 'a pleasant journey', or 'a hazard free municipality'.

The entities the task can act upon may be limited. For example moving will act upon a human being, a car, etc. A subset of tasks is tightly coupled with a class of real-world phenomena. For example running involves human beings and animals, not cars. This determines together with the work-field (e.g., 'financial transactions', 'hazard mapping') the context of a task.

Task decomposition is a topic of research in task analysis [144] and activity theory [231]. *Hierarchical task analysis (HTA)* is a technique for deriving task descriptions [144]. HTA uses a top-down approach by describing the goal of a system and then defining iteratively the subtasks and plans to fulfil that goal. These methods are useful for system design, but do not support formal system analysis due to the lack of formalisms.

3.2.1 Modelling dynamic systems

This section introduces relevant principles of dynamic systems in terms of behavioural aspects. A dynamic system can be seen as a set of processes. The terms described below are drawn from the formal description technique LOTOS (Language of Temporal Ordering Specification)[2] [33] and the LOTOS based Interaction System Design Language (ISDL) [278] (see also Section 3.2.2).

In LOTOS, a process is defined as follows:

Term 3.4 *A* process *is an entity able to perform internal, unobservable actions, and to interact with other processes that form its environment.*

A process (also sometimes called a *behaviour*) can be seen as the model of the behaviour of a system and its operations. The simultaneous execution of processes is referred to as *concurrency.* Concurrency is controlled by synchronisation elements, called *actions*, such as interactions and events. Concurrency control is needed to avoid *deadlock*, a situation that occurs when two processes wait for each other to finish. Interaction between processes takes place through *communication* at the processes' *interaction points.* The *signature* of a process comprises its identification, the type of all its parameters and its return type.

Actions are abstract elementary units of a process. They are considered to be *atomic* in the sense that, in contrast to composites of actions, they occur or they do not occur at all. They do not occur partially.

Behaviour can be modelled by two distinct model types, state-based models, which describe processes in terms of states and state transitions (e.g., workflow models) and predicative models, which describe processes in terms of pre/post conditions [167]. These models are also referred to as respectively intensional models (describing what systems do) and extensional models (based on what an outside observer sees) [51]. State-based models allow for simulation whereas, predicative models are more suitable for formal analysis of system behaviour. State-based models entail the group of theories known as process algebras such as pi calculus and other theories such as Petri Nets. A common notion in predicative models is a *trace*, or sequence of atomic actions executed by a system [51]. In behaviour analysis systems are represented by traces. Theories such as the situation calculus (a first-order logical language for reasoning about dynamic systems [192]) and dynamic logic have proven to be very useful for modern formal models that allow for reasoning on system behaviour [234].

In addition, hybrid models can be found, such as applied in the process specification language ConGolog [167] and OWL-S (formerly DAML-S) [181] which is based on Petri Nets and situation calculus. Mappings are used for transitions between these notions [192]. OWL-S, as well as a number of other research efforts, uses ontologies to model the attributes of processes. The OWL-S coalition [59] defines an upper-ontology to enable the creation of service ontologies.

[2]ISO standard 8807:1989

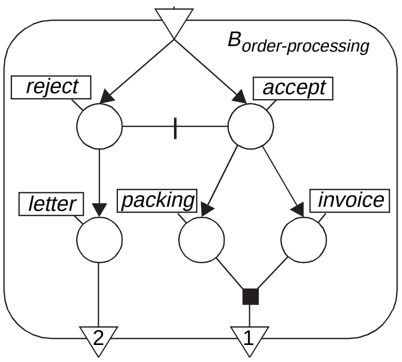


Figure 3.6: ISDL graphic representation for a mail ordering action. Taken from [278].

3.2.2 Formalised process models and languages

Formal models and languages have been developed to aid system architects in designing and analyzing complex systems by creating system representations, based on formal abstracted constructs. The model representations of a specific system are used for implementation work and for sharing the system characteristics amongst system designers and system users in order to facilitate a common understanding of the system's properties. The models are different in terms of the formalisms that they are based on, their modelling constructs and the representations that they provide. In the last decade many languages have been developed and gained extra interest with the increasing popularity of web services. The remainder of this section elaborates on the languages considered to be relevant for the context of this thesis.

ISDL The Interaction System Design Language (ISDL) comes with a set of methods to perform behaviour refinement, based on a careful consideration of the architectural concepts of action and causality relation in distributed systems [230, 278]. Based on the formal description technique and ISO standard LOTOS [33], the language provides constructs to model a system's internal structure and its behaviour as distinct properties. ISDL stands out due to the possibility of combining system structure and behaviour constructs in a well-defined textual as well as graphical representation. Figure 3.6 shows an example graphic representation of a mail ordering behaviour that takes a mail order as an input and depending on the acceptance, produces either a package plus an invoice, or a rejection letter (respectively, exits 1 and 2). Actions are represented by circles, interaction points by triangles. The round-edged box represents a behaviour. Figure 3.7 shows an equivalent textual representation.

```
behaviour B_order-processing
    entries
        entry
    actions
        entry ∧ ¬reject → accept,
        entry ∧ ¬accept → reject,
        reject → letter,
        accept → invoice,
        accept → packing
    exits
        letter → exit2,
        invoice ∧ packing → exit1
endbehaviour #B order-processing #
```

Figure 3.7: ISDL textual representation for a mail ordering action. Taken from [278].

UML The Unified Modelling Language (UML) provides modelling constructs for static structures as well as behaviour. A model is constructed through specifications in a so called semantic backplane. Diagrams provide visual views on the model. For behaviour the following diagram types have been defined: *Activity, UseCase, StateMachine, Sequence, Communication, InteractionOverview and Timing.* As mentioned earlier, a new version of UML, UML 2 has been finalised and its superstructure was adopted in October 2004. In contrast with UML 1.x, UML 2 has adopted formal semantics in its metamodel. Activity diagrams are based on Petri Nets, although the strength of the alignment between the two is questioned by Störrle and Hausmann [255]. The mail ordering example in 3.6 has its counterpart in the UML 2 activity diagram of Figure 3.8.

PSL The situation calculus based Process Specification Language (PSL) was developed as a reference language for other process specification languages [192]. PSL is defined in first-order logic, which facilitates reasoning with its implementations [32]. PSL is also known as the ISO 18629 standard.

PSL has been used to ascribe a semantics to DAML-S (the predecessor of OWL-S) [192]. An extension, built on top of PSL is used in the Semantic Web Services Ontology (SWSO), which is also inspired by OWL-S [259].

An integration of PSL and flow models in general and UML 2 in particular has been demonstrated in [32].

Process languages for web services Since the introduction of web services many XML-based languages have been created for the modelling of web service chains. A leading initiative in industry is the Web Services Business Process Execu-

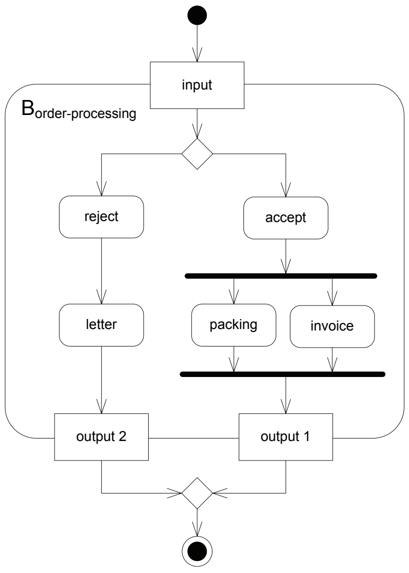


Figure 3.8: UML 2 activity diagram equivalent of the mail ordering activity depicted in Figure 3.6.

tion Language (WSBPEL) [201]. WSBPEL has merged ideas from its predecessors XLANG and WSFL, but has dropped their well-defined semantics [24]. Without such semantic definition it is impossible for a computer program to reason with the process model. For practical purposes of service matching, this is a serious limitation. However, WSBPEL is useful for the composition of services that are known to be interoperable.

The listing below provides an example of a WSBPEL service chain, defining a sequence of three services in a Google search (input, search and output). Services are implemented through *partnerlinks* and *porttypes* which are defined in a corresponding WSDL [3] file. For conciseness, the declarations of namespaces and partnerlinks are left out.

```
<process name="GoogleFlow">
   :
   <!-- Declarations of namespaces and partnerlinks -->
   :
 <sequence>
```

[3]Web Service Description Language

```
    <!-- Receive input from requestor -->
    <receive name="receiveInput" partnerLink="client"
         portType="tns:GoogleFlow"
         operation="initiate" variable="request"
         createInstance="yes"/>

    <!--  invoke the remote Google Search web service -->
    <invoke name="invoke" partnerLink="GoogleSearch"
            portType="gns:GoogleSearchPort"
            operation="doGoogleSearch" inputVariable="request"
            outputVariable="response"/>

    <!-- Asynchronous callback to the requester -->
    <invoke name="replyOutput"
            partnerLink="client"
            portType="tns:GoogleFlowCallback"
            operation="onResult"
            inputVariable="response"/>

  </sequence>
</process>
```

The plethora of web service chain languages have caused recent efforts to embark upon languages that make chain definitions interoperable.

The Web Services Choreography Description Language (WS-CDL) of the World Wide Web Consortium's Web Services Choreography Working Group aims at defining peer-to-peer collaborations between web services [284]. The language specifies workflow constructs at a conceptual level. It leaves the specification of an application's execution logic to process execution languages, such as WSBPEL.

Several efforts, such as OWL-S [181], provide language constructs for modelling semantic properties of web services. OWL-S is described in Chapter 4.

3.2.3 Process modelling in GIS

In addition to the generic approaches of process modelling discussed in the previous sections, a more specific analysis of tasks and operations in the geographic domain is needed. Approaches can be found across different levels of abstraction (cf. Figure 3.1), for example, analyzing the behaviour of objects in geographic space, analyzing system end-user behaviour and classifying GIS operations [174]. Decomposition strategies of processes are reported in [292] for the domain of planning support systems. Other initiatives with similar objectives are reported in [42, 152, 190, 263]. The need for formal modelling of geographic processes has been identified in only a few sources, e.g., [6, 66].

In an effort to provide a basis for creating geo-service specifications, OGC and ISO have developed respectively the OGC service architecture [205] and the ISO 19119 standard for Geographic Information Services [132]. In these specifications, geographic processes are viewed as service chains, following the RM-ODP (see Section 2.4.2). ISO 19119 treats service chains as directed graphs and models them using the UML activity graphs. To exchange service chain information with other users performing tasks in a similar situation, ISO 19119 provides a taxonomy of GIS operations (elaborated in Section 5.4) and introduces a *Service Organiser Folder* (SOF) as a bag of unordered services to be used in a particular chain.

None of the above efforts have resulted in comprehensive formal machine-accessible geographic process models that support semantic interoperability. This forms a part of the motivation to perform this research as described in 1.2. The way in which geographic processes are modelled in this research, is described in Section 5.4.

3.3 Service chaining

The need for modular software processes has been identified in Chapter 2. The integrated exploitation of those processes in distributed systems is facilitated by service chaining, which is defined as follows:

Term 3.5 Service chaining *(adapted from [132]) is combining services in a dependent series to achieve larger tasks. It involves service discovery, composition and execution.*

We assume that service chaining is taking place in the context, depicted in Figure 3.5 and in a more practical sense, may use languages as described in Section 3.2.2. The remainder of this section describes the user context and different modes of service chaining.

3.3.1 User context

In modular software architectures, users will be able to use a single service, create simple service chains or assemble services into their own applications, on demand. In order to evaluate the usefulness of a service, alone or as part of a chain, one needs to interpret the elements of its interface or infer from a service description the syntactic, structural and semantic properties of the service's operations, workflow and input, output and tightly-coupled data. This process of service finding is called *service discovery.* Service discovery is done through the evaluation of service descriptions that are identified with available services [4].

Figure 3.9 shows the user context of service *discovery and composition.* This figure is closely related to Figure 2.2. However, in the figure, the end-user application and both its creator and user are now involved with service discovery,

[4]Service descriptions may be stored with the service and/or published in a separate directory.

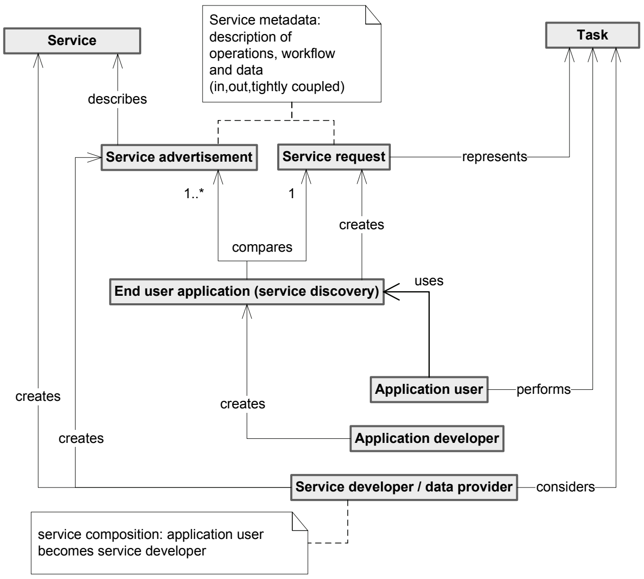

Service
Service metadata: description of operations, workflow and data (in,out,tightly coupled)
Task
describes
Service advertisement
Service request
represents
1..*
1
creates
compares
End user application (service discovery)
uses
Application user
performs
creates
creates
Application developer
creates
Service developer / data provider
considers
service composition: application user becomes service developer

Figure 3.9: User context of service discovery and composition (UML class diagram).

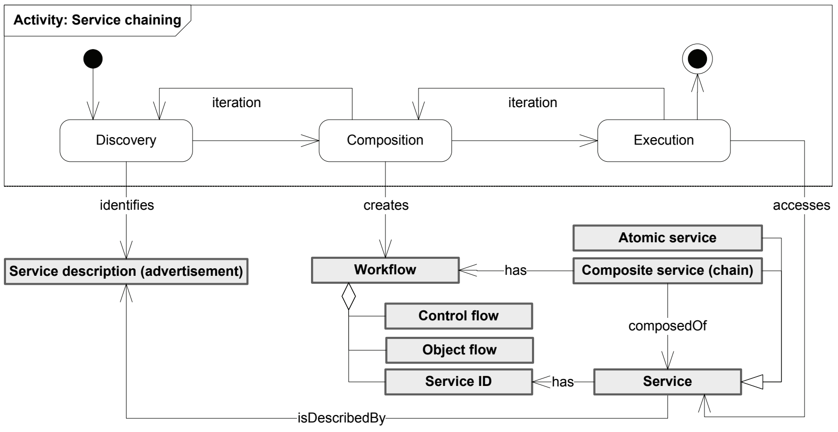


Figure 3.10: The elements of service chaining, depicted in the combination of a UML activity diagram (top) and a UML class diagram (bottom).

rather than service access. In service discovery, service advertisements are created by the service developer and may contain descriptions of operations, workflow and data (input, output, tightly coupled data). An end user, trying to perform a task with services, uses a service discovery application to create a service request and to compare this request with available service descriptions.

3.3.2 Modes of discovery, composition and execution

Service chaining is typically performed as a sequence of discovery, composition and execution, see Figure 3.10. However, sometimes discovery, composition and execution are performed as an iterative procedure, e.g., part of the chain is discovered after another part has been composed or even has been executed. Further, the 'mode' of service chaining is influenced by other types of variations are caused by (1) the degree of control a human user has on the discovery, composition and execution process and (2) the fact that certain services in the chain have been prescribed.

Discovery

There are two starting points for discovery:

1. The user may be confronted with a task and has no prescriptions for the component services in the chain.

2. The user is constrained to use:

   (a) specific types of services. This may involve any type variation along the classification axes in Section 2.1.2. A constraint may be for example that all services must have a web service interface. In another scenario we would allow to have human operated interfaces in the chain.
   (b) specific instances of services (e.g., a service provided by a certain company, because it has been proven reliable)

The user will be tasked to find a match between service request and advertisements. In case of more than one match, the user may choose the best, based on human or machine inference of the properties of each match. There are two options:

1. The user finds a composite service that fulfils the task.
2. There is no single composite service that fulfils the task. The service request has to be decomposed, for which there are three options:
   (a) The user 'manually' performs the decomposition and repeatedly uses the discovery application for finding each service part.
   (b) The discovery application performs the decomposition automatically.
   (c) The user performs the decomposition semi-automatically, i.e., by 'suggesting' service parts and using the discovery application to 'fill the gaps'.

Composition

Once the right service parts are found, the user composes these services in a workflow. This workflow becomes executable once binding of its operations and data is performed by creating a new composite service. In this phase the user is considered to be a service developer[5]. When composition and execution platforms are merged, a workflow may be directly executed. In other cases it may be exported and stored to be executed at another place and/or at another time. Workflows can be stored in languages such as described in 3.2.2.

Execution

ISO 19119 (Services) [132] defines three different architecture patterns for service chaining. Apart from one pattern (transparent chaining), they focus merely on *execution* details. A distinction is made between the following patterns:

- Transparent chaining (see Figure 3.11). The user is in full control of the discovery process and the execution process. He invokes each service through a software client.

[5]The reader is reminded that a workflow in the form of a description is not executable. If it takes the form of e.g., a script it must be embedded in a service.

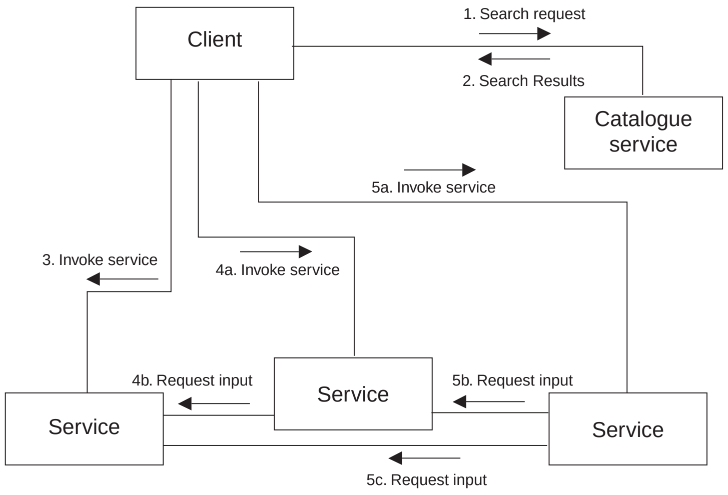


Figure 3.11: UML communication diagram showing transparent chaining. Figure taken from [132].

- Translucent chaining (see Figure 3.12). A chain is given and the user invokes it through a workflow service. The status of the service chaining process is communicated to the user.
- opaque chaining (see Figure 3.13). A chain appears as a single service to the user. It is executed by a so called aggregate service that handles all the transactions in the chain.

3.4 Summary and reflection

This chapter has proposed an approach for the abstraction of geo-information to support the semantic modelling of geo-information and geo-services. It has been derived from several approaches existing in both ICT and geo-specific communities. The principle idea is to break down the complexity of these models, so that they have a clear structure and incorporate a separation of concerns. This approach aims at filling a niche in the existing way of dealing with interoperability as discussed in Chapter 2. The ideas are used in the design of a semantic interoperability framework, which is elaborated upon in Chapter 5.

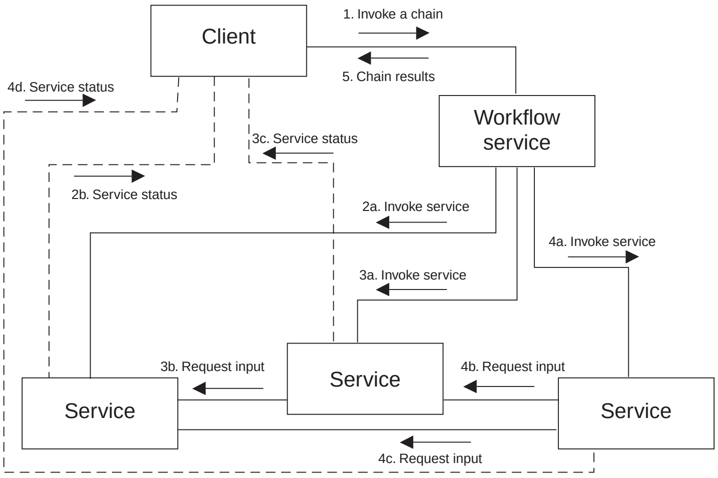


Figure 3.12: UML communication diagram showing translucent chaining. Figure taken from [132].

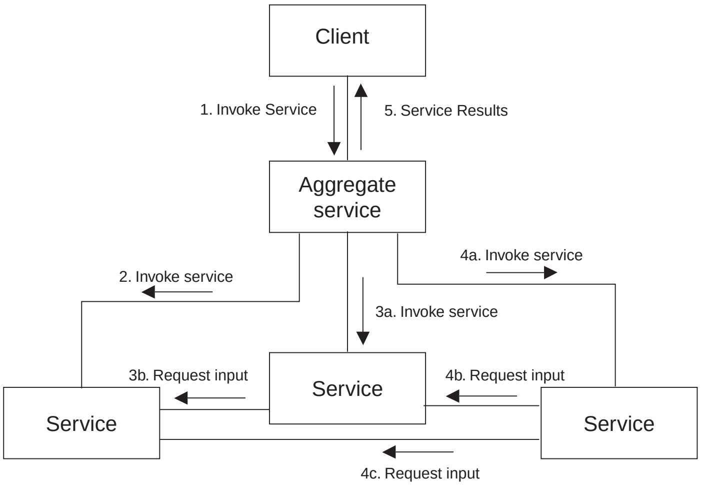


Figure 3.13: UML communication diagram showing opaque chaining. Figure taken from [132].

Chapter 4

Semantic modelling

The goal of semantic modelling is to provide a conceptual model of a domain of discourse by formalising the semantics of the concepts and inter-concept relationships in this domain. In this thesis work, the aim of such formalisation is to facilitate machine reasoning about geo-services for the purpose of interoperability. As discussed in Chapter 2, the agreements on the communication syntax (rules for the serialisation of elements in a message), communication structure (rules for the structure of the underlying schema of a message) and communication semantics (the meaning of message elements and the message as whole) are key to the interoperability between services. Those agreements can be laid down in several ways, ranging from informal (e.g., human language-based) to formal (e.g., mathematical logic-based) contracts, and from human-readable to machine-readable contracts. In practice, the specifications of such contracts appear as documents, which increasingly exist of a combination of human language and machine processable language (e.g., XML schemas). Relevant examples of interoperability models that specify such contracts for software systems have been discussed in Section 2.4. In this chapter we focus on the building blocks of the Semantic Web (introduced in Section 2.4.4), which facilitate the creation of semantic models for the purpose of machine reasoning. Our target is to formalise (geo-)service models as discussed in Chapter 3.

This chapter starts with an explanation of the notion of *ontology* as one of the corner stones of the Semantic Web (Section 4.1). Section 4.2 presents the foundations for the type of ontologies that can be used for machine reasoning. Practical issues of ontology design and representation are discussed in respectively Sections 4.3 and 4.4. Section 4.5 presents the notion of *knowledge base* and techniques of machine reasoning that are used to infer relationships between concepts and instances in a knowledge base. Section 4.6 discusses the embedding of ontologies in a semantic interoperability framework as a basis for creating and using ontology-based descriptions of information sources. This chapter ends with a description of methods for the semantic modelling of services (Section 4.7) and a summary and reflection (Section 4.9).

4.1 What is an ontology?

Ontologies provide a way to share the semantics of concepts in some area of interest, such as car driving, choosing a pizza or processing geographic information. It is all about common understanding of essential concepts, such as the type of cheese used on a Pizza Margherita and, in the latter case, what is meant for example by 'geometric object'.

Much debate has been (and is still) taking place on the interpretation of the term 'ontology' itself.

A generally accepted short definition of ontology is given by Gruber:

Term 4.1 *(From [103]) An* ontology *is an explicit specification of a conceptualisation.*

The term conceptualisation is defined as follows:

Term 4.2 *(From [88, 103]) A* conceptualisation *is the combination of objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualisation is an abstract, simplified view of the world that we wish to represent for some purpose.*

To emphasise the intended sharing of semantics, some ontology definitions indicate the notion of agreement. Uschold et al. [268] and Guarino [104] cite the following statement from Tom Gruber:

> *Ontologies are agreements about shared conceptualizations. Shared conceptualizations include conceptual frameworks for modelling domain knowledge; content-specific protocols for communication among inter-operating agents; and agreements about the representation of particular domain theories. In the knowledge sharing context, ontologies are specified in the form of definitions of representational vocabulary. A very simple case would be a type hierarchy, specifying classes and their subsumption relationships. Relational database schemata also serve as ontologies by specifying the relations that can exist in some shared database and the integrity constraints that must hold for them. (Tom Gruber, 1994, SRKB Mailing list)*

Another definition of ontology is used in [267]:

> *An ontology is a collection of shared concepts. More formally, an ontology is 'a structured, limitative collection of unambiguously defined concepts'. This definition contains four elements:*
>
> 1. *An ontology is a collection of concepts.*

2. *The concepts are to be unambiguously defined.*
3. *The collection is limitative. Concepts not in the ontology cannot be used.*
4. *The collection has structure. Structure means that the ontology contains relationships between the concepts.*

To clarify the context of the term ontology further, the most important contextual concepts are depicted in a UML diagram, see Figure 4.1. This contextual view integrates the ideas of leading literature in the field of ontology research [103, 104, 105, 183, 257, 268].

Ontologies can be distinguished by their degree of formality, extent of explication, structure complexity, scope, specification language, expressiveness and representation. Each of these characterisations are explained below.

Degree of formality Ontologies may differ in their degree of formality. Uschold [268] defines four levels:

- Highly informal. Concepts are expressed loosely in natural language.
- Semi-informal. Concepts are expressed in a restricted and structured form of natural language.
- Semi-formal. Concepts are expressed in an artificial formally defined language. Examples are models that are expressed in UML, RDF, OWL, etc.
- Rigorously formal. Concepts are defined with formal semantics, theorems and proofs of such properties as soundness and completeness.

These levels reside in a continuum of formality, which means that any intermediate level may occur, depending on the formalisms used to design the ontology.

In recent research efforts in artificial intelligence and software engineering, ontologies are often considered to be formal (semi-formal or rigorously formal). In this thesis, those ontologies are termed as *formal ontologies.* With the increasing importance of ontology-based machine reasoning, ontologies are often also considered to be machine accessible and give some support to machine reasoning. Ontologies of this kind are considered to make use of some kind of formalism. In this thesis, those ontologies are indicated with the term *machine ontologies.* As an example, ontologies that are expressed in OWL, are machine ontologies.

Extent of explication A conceptualisation provides the meaning of a set of terms, called vocabulary [1]. A conceptualisation also provides the context of these terms and places them in a structure. A conceptualisation may exist in the mind of a human or can be (partially) made explicit in an ontology. The extent to

[1] also coined as 'shared' vocabulary, because it is almost always intended to be shared

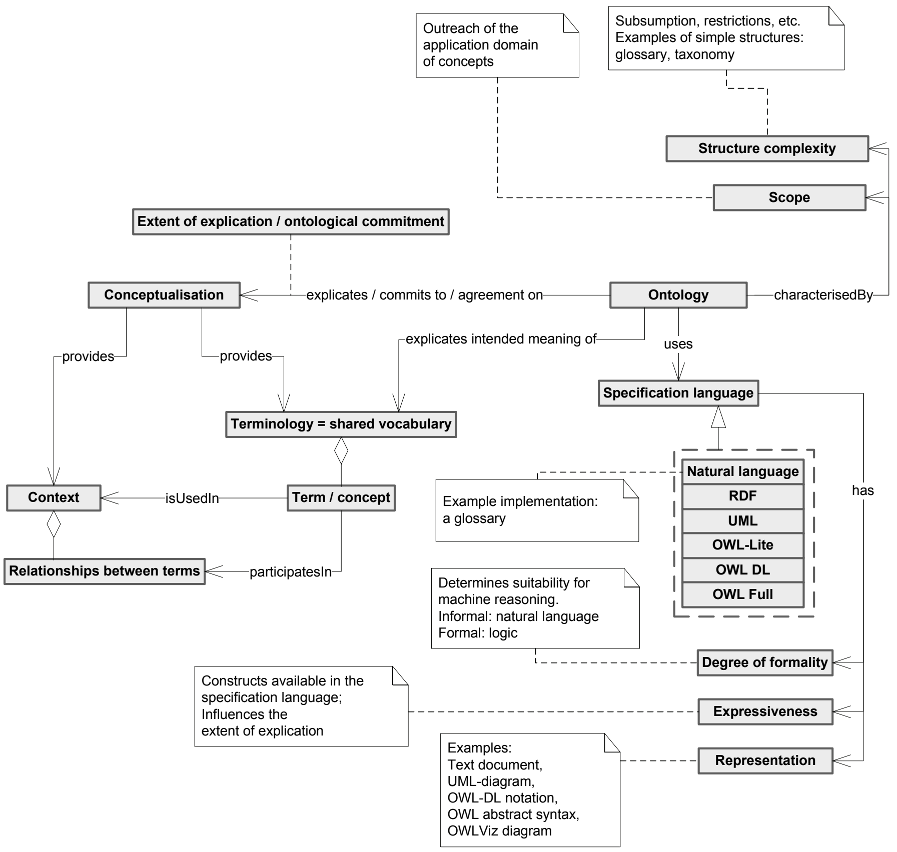

Outreach of the application domain of concepts
Subsumption, restrictions, etc. Examples of simple structures: glossary, taxonomy
Structure complexity
Scope
Extent of explication / ontological commitment
Conceptualisation
explicates / commits to / agreement on
Ontology
characterisedBy
explicates intended meaning of
uses
provides
provides
Specification language
Terminology = shared vocabulary
Natural language
RDF
UML
OWL-Lite
OWL DL
OWL Full
Example implementation: a glossary
has
Context
isUsedIn
Term / concept
Relationships between terms
participatesIn
Determines suitability for machine reasoning. Informal: natural language Formal: logic
Degree of formality
Constructs available in the specification language; Influences the extent of explication
Expressiveness
Examples: Text document, UML-diagram, OWL-DL notation, OWL abstract syntax, OWLViz diagram
Representation

Figure 4.1: UML class diagram showing the context and characteristics of the concept of 'ontology'.

which the ontology makes the conceptualisation explicit may vary. A lower level of explication, or ontological commitment, means that the conceptualisation relies more on implicit assumptions (which are not directly verifiable).

Structure complexity An ontology explicates the structure between concepts by using relational constructs provided by the specification language. An example of a simple structure is a taxonomy, or generalisation hierarchy, which makes us of so called 'is-a' relationships. More complex structures are formed when also 'part-of' relationships are involved or any other type of relationship, such as 'uses' or 'produces'. It is important that the semantics of these relationship types are sufficiently defined. Complex structures may be defined with formal languages that are equipped with special constructs for that purpose (e.g. OWL). In fact, natural language may also be used for creating complex structures, but they will easily become obscure. However, the use of natural language remains important for explaining the meaning of specific constructs used in formal languages.

Scope The scope of an ontology is determined by the area of interest of the underlying conceptualisation (also called the *domain of discourse*). A 'land cover' ontology typically contains concept definitions such as 'Forest' and 'Rice field' but not 'Population density' or 'Hotel rates'. When the scopes of two ontologies overlap, a mapping of similar terms may be required to enable their integral use.

Specification language An ontology can be specified in natural language or in a computer language. Natural language may be used to create definitions of terms, such as in a shared vocabulary, but it is not efficient in creating a structure of relationships between them. An object-oriented design language such as UML provides several formalisms, such as generalisation and aggregation, to create conceptual structures. However, UML was not designed to allow for reasoning on the conceptual structures implemented with it and it allows for the creation of quite informal models. Although OCL provides powerful additional constructs to formalise the semantics in a UML-based model, it brings about computational complexity of reasoning [55] and lacks the basis of XML as communication mechanism. XML-based languages such as RDF and OWL have the advantage of machine-accessibility over the web. In addition, OWL provides the formal constructs that allow for decidable reasoning with conceptual structures[2]. .

Expressiveness The expressiveness of a language is determined by the variety of expressions that can be created with its constructs, given the constraints to which the constructs must comply. Expressive languages (of which natural languages are good examples) allow for a wide spread application, but implementations expressed with them often suffer from ambiguity. The ontology language OWL has three

[2]in fact only a subset of OWL, named OWL-DL, supports decidable reasoning, (see a description of OWL in Section 4.2.3)

sub languages, namely OWL-lite, OWL-DL and OWL-full. OWL-DL contains more expressions and less restrictions than OWL-lite. OWL-full contains more expressions and less restrictions than OWL-DL. This makes OWL-full the most expressive sub language. However OWL-full suffers from undecidability and cannot be used by current ontology reasoners. More details on OWL can be found in Section 4.2.3.

Representation An ontology may be represented through its specification language by a variety of artefacts. Natural language artefacts are normally text documents. UML specifications are laid down in UML diagrams accompanied with textual explanation. Implementations based on ontology languages may be represented by mathematical notation, computer code (such as XML), diagrams and/or textual explanations.

4.2 Foundations for machine ontology

Ontology languages provide the building blocks for machine ontologies. An ontology language consists of constructs that are used (1) to declare the concepts of a domain, (2) to define the relationships between these concepts and (3) to impose restrictions on those relationships [9]. The expressiveness of an ontology language determines the range of ontological statements that we can create with these constructs. The constructs of a machine ontology language are defined by their syntax and semantics and typically support a certain level of machine reasoning. Several machine ontology languages have been developed for use in conjunction with the web. Amongst those, OWL is currently the most prominent one. The remainder of this section describes the two major foundations of OWL, i.e., Description Logics and RDF, and then provides an overview of its constructs.

4.2.1 Description Logics

Research in artificial intelligence in the 1970s brought forth two basic knowledge representation approaches, i.e., logic-based formalisms and cognitive representations. In the latter, knowledge is represented by ad hoc data structures, for example, semantic networks and frames [14]. Logic-based systems were considered to be more powerful, network systems to be more practical. A fusion began with the development of *terminological systems* that evolved in the mid 1980s in *Description Logics* (DLs). In these approaches, logic-based constructs are used to establish a basic terminology of the domain of discourse. Description Logics are a family of languages, each with their own set of constructs [245]. The term 'Description' Logics stems from the fact that the notions of a domain are represented by concept *descriptions* [15]. The basic notions of DLs draw upon first order predicate logic. They are concepts (unary predicates) and roles (binary relations). For example, a *subsumption* relation (also referred to as *subclass* or *IS-A* relation), can be described as:

$$\mathit{Building} \sqsubseteq \mathit{Construction} \tag{4.1}$$

The following expression describes a building that has at least three floors and has an accommodational function. It uses the *intersection* construct ($\sqcap$), the *unqualified number restriction* construct ($\geq$) and the *existential quantifier* construct ($\exists$, which means: 'there exists at least one'):

$$\mathit{Building} \sqcap (\geq 3\ \mathit{hasFloor}) \sqcap \exists \mathit{hasFunction}.\mathit{Accommodation} \tag{4.2}$$

The expressions below characterise respectively a country with a coast and a landlocked country:

$$\mathit{Country} \sqcap \exists \mathit{hasBorderWith}.\mathit{Sea} \tag{4.3}$$

$$\mathit{Country} \sqcap \forall \mathit{hasBorderWith}.(\neg \mathit{Sea}) \tag{4.4}$$

The latter expression uses the *universal quantifier* construct ($\forall$, which means: 'for all') and the *negation* construct ($\neg$). Expressions with universal quantifiers have to be used with care as we will see in Section 4.5.1.

Concept and role descriptions

In Description Logics, a concept (e.g., 'construction') is interpreted as a set of individuals (also called the *extension* of the concept [9], e.g., 'Eiffel Tower', 'Taj Mahal', etc.), and a role (e.g., 'hasMaterial') is interpreted as a pair of individuals (e.g., the pair ('Eiffel Tower','Steel')). As a notational convention, concepts are denoted with capital characters C and D, roles with capitals R and S, and individuals with lower case a and b. Formally, an *interpretation* consists of an (arbitrary) interpretation domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps every concept to a subset of $\Delta^{\mathcal{I}}$, every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and every individual to an element of $\Delta^{\mathcal{I}}$ [82]. *Atomic* concepts (unary predicates) are subsets of the interpretation domain, other concepts are obtained by using constructs such as concept intersection and union. The notion of interpretation is used to define the semantics of the constructs of a DL. For example, the semantics of the basic Description Logic $\mathcal{ALC}$ are defined by extending the interpretation function, such that its mappings satisfy the following equations [14]:

$$\begin{array}{rcll}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} & \text{(top concept)} \\
\bot^{\mathcal{I}} & = & \emptyset & \text{(bottom concept)} \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} & \text{(negation)} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} & \text{(intersection)} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} & \text{(union)} \\
(\exists R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} & \text{(existential quantif.)} \\
(\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} & \text{(universal quantif.)}
\end{array}$$

The notions on the left-hand side are called *concept constructors*. The right-hand side of the equation for the existential quantification has to be read as [245]:
'for any $a \in \Delta^{\mathcal{I}}$, there exists a $b \in \Delta^{\mathcal{I}}$, with $(a, b) \in R^{\mathcal{I}}$ and $b \in C^{\mathcal{I}}$',
and for the universal quantification (also called 'value restriction') as:
'for any $a \in \Delta^{\mathcal{I}}$ holds that for all $b \in \Delta^{\mathcal{I}}$, if $(a, b) \in R^{\mathcal{I}}$, then $b \in C^{\mathcal{I}}$'.

The syntax of DLs is *variable-free*. According to Nardi and Brachman, [193] a concept definition C compares to the first-order logic notion $C(x)$, where the variable ranges over all individuals in the interpretation domain and $C(x)$ is true for all individuals that are member of concept C. In the light of this translation, description languages can be seen as fragments of first order predicate logic [16].

Within the family of DLs, a specific DL, such as $\mathcal{ALC}$ or $\mathcal{SHIQ}$, is characterised by the particular constructs that are used to form complex concepts and roles from atomic ones [245]. Each DL is named with a combination of characters, each of which represent a set of constructs. Several extensions to $\mathcal{ALC}$ exist. For extensions that add concept constructors, capital letters are added to $\mathcal{ALC}$, for role constructors, symbols are added as superscripts and for restrictions on the interpretation of roles, symbols are added as subscripts [14]. For example, the DL $\mathcal{ALCQHI}_{R^+}$ extends $\mathcal{ALC}$ with qualifying number restrictions ($\mathcal{Q}$), role hierarchies ($\mathcal{H}$), inverse roles ($\mathcal{I}$) and transitively closed primitive roles ($_{\mathcal{R}^+}$) [122]. The influential $\mathcal{ALCQHI}_{R^+}$ is also known as $\mathcal{SHIQ}$. In addition, if DL languages are extended with simple concrete datatypes (e.g., integer or string), the symbol ($\mathcal{D}$) is added (cf. $\mathcal{SHIQ}(\mathcal{D})$) [120].

Knowledge bases

A knowledge base consists of a set of terminological axioms (called the *TBox*) and a set of assertional axioms or 'assertions' (called the *ABox*) [14]. The syntax and semantics of these axioms is found below:

Terminological axioms

Syntax	*Semantics*	*Name*
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	(concept inclusion)
$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	(role inclusion)
$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$	(concept equality)
$R \equiv S$	$R^{\mathcal{I}} = S^{\mathcal{I}}$	(role equality)

Assertional axioms

Syntax	*Semantics*	*Name*
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	(concept assertion)
$R(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$	(role assertion)

TBox The TBox is the component of a knowledge base that contains *intensional* knowledge [193] and constitutes a terminology of a problem domain (the T in TBox refers to 'terminology'). Formally, if the TBox $\mathcal{T}$ is a set of axioms, then the interpretation $\mathcal{I}$ satisfies $\mathcal{T}$ if and only if $\mathcal{I}$ satisfies each element of $\mathcal{T}$. In this case $\mathcal{I}$ is called a *model* of $\mathcal{T}$. An axiom in a TBox can be of the form $C \equiv D$. Such an equality whose left-hand side is an atomic concept is called a *definition* [14], giving necessary and sufficient conditions to the concept. In an ontology, such a concept is also referred to as a *defined class* [118][3]. An example of a definition is:

$$CountryWithCoast \equiv Country \sqcap \exists hasBorderWith.Sea \tag{4.5}$$

The above axiom defines 'CountryWithCoast' as a country that has at least one of its borders with 'Sea'. At the same time, reading in the opposite direction, any country that has at least one of its borders with 'Sea', is considered to be a 'CountryWithCoast' (due to the sufficient condition).

In case we are unable to define a concept completely, a concept inclusion can be used to state necessary conditions only. Such a concept is also referred to as *primitive* class in an ontology. An example of a concept inclusion is:

$$Hotel \sqsubseteq Building \sqcap \exists hasFunction.Accommodation \tag{4.6}$$

The above axiom states that a hotel is a building with an accommodational function, but not every building with an accommodational function is a hotel. The *classification* of concepts is seen as the basic task of building a terminology and involves the validation of subsumption relationships between concepts. *Reasoning* makes the implicit knowledge of the knowledge base explicit by the calculation of inferences. Reasoning is discussed in Section 4.5.

ABox The ABox is the component of a knowledge base that contains *extensional* knowledge [193], which is specific to the individuals of a problem domain. The A in ABox refers to 'assertional' knowledge. Baader [14] also denotes the ABox as *world description*, due to the fact that it introduces individual names and their properties as the specific states of affairs of concepts and roles. The example below states that the individual 'Tuvalu' is an instance of the class 'Country', or in other words, it belongs to the interpretation of 'Country'. An implication of this assertion is that, if Country is a defined as a subclass of, for example, 'Administrative area', then Tuvalu is also an instance of 'Administrative area'.

$$Country(Tuvalu) \tag{4.7}$$

An example of a role assertion is given below. It specifies that Tuvalu borders the Pacific Ocean. From the descriptions 4.5 and 4.7 it can be inferred that Tuvalu is a country with a coast.

$$hasBorderWith(Tuvalu,PacificOcean) \tag{4.8}$$

[3]A class can be seen as concrete representations of a concept

Further ABox reasoning is discussed in Section 4.5.

Practical modelling constructs

Current systems that support the creation of machine ontologies, such as Protégé (see Section 8.2.1), provide specific ontology design constructs, that can be retraced to DL-constructs. Some of them are described below.

Domain and range A role can be assigned a domain and range. The domain constraints the concepts to which a role can apply. This is equivalent with the DL axiom $\exists R.\top \sqsubseteq C$. The range constraints the concepts that can fill the role (so called role fillers), which is equivalent with the DL axiom $\top \sqsubseteq \forall R.C$.

Intersection In Protégé, multiple asserted conditions are treated as intersecting classes. Restricting the class C by the conditions $C \sqsubseteq D$ and $C \sqsubseteq E$ means that C is a subclass of the intersection of D and E:

$$C \sqsubseteq (D \sqcap E) \tag{4.9}$$

The latter can be used as one condition in Protégé as well, having the same semantics as the combination of the two above.

Nested conditions Axioms that contain conditions with 'nested' roles are, in this thesis, called *nested conditions.* Nested conditions can involve existential quantification and universal quantification. The example below contains a nested existential quantification:

$$\exists R.(\exists S.C) \tag{4.10}$$

4.2.2 Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web [179]. The basic notion of RDF is a data model that uses statements, (called *triples*), containing a subject, an object and a predicate that relates subject and object (see Figure 4.2). The RDF data model is a graph [249]. RDF uses an XML syntax for exchanging its graphs, called RDF/XML.

RDF *Schema* (RDFS) is a separate language which can be used to specify a domain specific vocabulary for usage in RDF. It can be seen as a type system for RDF [179]. RDF Schema provides constructs such as 'Class' and 'subClassOf' to establish such vocabulary and is considered to be an ontology language that enables a user to create a classification hierarchy with typing of properties [9]. Figure 4.3 shows an example of an RDFS triple.

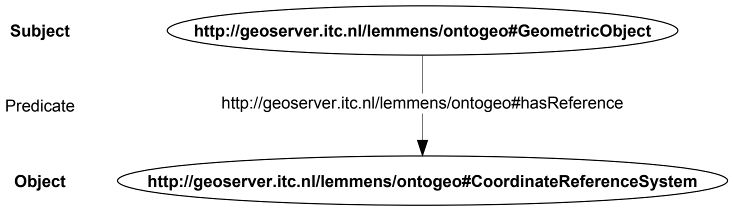

Subject
http://geoserver.itc.nl/lemmens/ontogeo#GeometricObject
Predicate
http://geoserver.itc.nl/lemmens/ontogeo#hasReference
Object
http://geoserver.itc.nl/lemmens/ontogeo#CoordinateReferenceSystem

Figure 4.2: Example of an RDF triple.

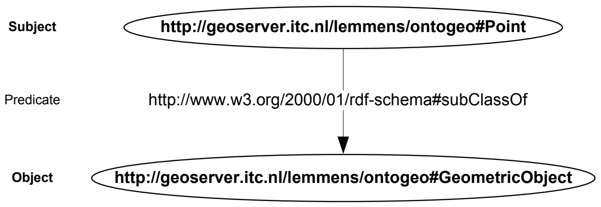

Subject
http://geoserver.itc.nl/lemmens/ontogeo#Point
Predicate
http://www.w3.org/2000/01/rdf-schema#subClassOf
Object
http://geoserver.itc.nl/lemmens/ontogeo#GeometricObject

Figure 4.3: Example of an RDFS triple.

4.2.3 Web Ontology Language (OWL)

The need for more expressive ontology languages than RDFS has initiated the development of new ontology languages (e.g., RDFS does not contain constructs for intersections, unions or complements of classes, nor does it support cardinality constraints [179]). In 1999, the DARPA Agent Markup Language (DAML) program was started to facilitate semantic interoperability among the projects in the DARPA program and resulted in the DAML-ONT language [119]. At the same time a European initiative embarked upon the development of another web based ontology language, called OIL (the Ontology Inference Layer), which applied the semantics of the Description Logic $\mathcal{SHIQ}$. The merger of the two initiatives has resulted in the DAML+OIL language. From here, the further development of the ontology language was taken up by the World Wide Web Consortium (W3C) Web Ontology Working Group. The Web Ontology Language became a W3C recommendation in February 2004 [184]. The Web Ontology Language (OWL) has been designed on the basis of Description Logics, RDF and, obviously, DAML+OIL [121]. Further, OWL makes use of frames, that group the information on each class (its super class and its properties), a principle that makes the code easier to read by humans and that is also used in the knowledge representation mechanism of Protégé (see Section 8.2.1). OWL draws upon the fact-stating capability of RDF and the class/property structure of RDF Schema. In addition, OWL provides more powerful constructs for class expressions. Within the OWL statements, RDF statements are used in their original form, see the OWL code example below. It defines the concepts of *Point*, *Line* and *Polygon* as subconcepts of the concept *GeometricObject*.

```
<owl:Class rdf:ID="Polygon">
  <rdfs:subClassOf rdf:resource="#GeometricObject"/>
</owl:Class>
<owl:Class rdf:ID="Point">
  <rdfs:subClassOf rdf:resource="#GeometricObject"/>
</owl:Class>
<owl:Class rdf:ID="Line">
  <rdfs:subClassOf rdf:resource="#GeometricObject"/>
</owl:Class>
```

OIL can be seen as a syntactic variant of the DL $\mathcal{SHIQ}(\mathcal{D})$ [120]; OWL is very close to $\mathcal{SHOIN}(\mathcal{D})$ [121], which, compared to $\mathcal{SHIQ}(\mathcal{D})$, supports also nominals[4] ($\mathcal{O}$) and is restricted to unqualified number restrictions ($\mathcal{N}$ instead of $\mathcal{Q}$). The formal semantics of OWL is provided as part of the OWL recommendation [222] and is very similar to the semantics provided for Description Logics.

Specific OWL constructs are listed in Table 4.1. Note that the OWL language also includes RDF and RDFS constructs (e.g., rdfs:subClassOf), which are not included in this table.

The elements of Table 4.1 form the basis for three sub-languages of OWL: OWL-Lite, OWL-DL and OWL-Full (in order of increasing expressiveness).

- OWL-DL: With 'DL' standing for Description Logics, OWL-DL's semantics are based on the semantics of the DL $\mathcal{SHOIN}(\mathcal{D})$. OWL-DL is designed to maximally exploit the formalisms of Description Logics *and* its computational tractability [10]. In order to achieve this, it puts restrictions on the use of the constructs as listed in Table 4.1, which basically yields that it —unlike RDF and RDFS— does not allow classes to act as individuals, and the language constructs cannot be applied to the language itself (e.g., to apply a cardinality constraint to a subclass construct). These restrictions are required for current reasoners to guarantee a decidable reasoning procedure [22]. For this reason, OWL-DL is also called *decidable*.

- OWL-Lite: The design of OWL-lite aimed at an 'easier' OWL for users and reasoners. OWL-Lite is more restricted than OWL-DL. Besides further restricting the use of OWL elements, it prohibits the use of the following OWL elements:

[4]Nominals are individual names expressed in class descriptions. They are used to identify a specific set of individuals, such as {Italy,France}.

owl:AllDifferent	owl:intersectionOf
owl:allValuesFrom	owl:InverseFunctionalProperty
owl:AnnotationProperty	owl:inverseOf
owl:backwardCompatibleWith	owl:maxCardinality
owl:cardinality	owl:minCardinality
owl:Class	owl:Nothing
owl:complementOf	owl:ObjectProperty
owl:DataRange	owl:oneOf
owl:DatatypeProperty	owl:onProperty
owl:DeprecatedClass	owl:Ontology
owl:DeprecatedProperty	owl:OntologyProperty
owl:differentFrom	owl:priorVersion
owl:disjointWith	owl:Restriction
owl:distinctMembers	owl:sameAs
owl:equivalentClass	owl:someValuesFrom
owl:equivalentProperty	owl:SymmetricProperty
owl:FunctionalProperty	owl:Thing
owl:hasValue	owl:TransitiveProperty
owl:imports	owl:unionOf
owl:incompatibleWith	owl:versionInfo

Table 4.1: Language elements of OWL (taken from [22]).

owl:oneOf
owl:unionOf
owl:complementOf
owl:hasValue
owl:disjointWith
owl:DataRange

OWL-Lite has similarities with the DL $\mathcal{SHIF}(\mathcal{D})$. Despite its decreased expressive power, it improves the tractability with respect to OWL-DL [121].

- OWL-Full: OWL-Full is the least restricted OWL sub language. It was designed to support upward compatibility with RDF and RDFS. The trade-off of this increased expressiveness is that OWL-Full is undecidable.

The following rules concern the compatibility between the three OWL sub languages (literally from [10]):

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.

- Every valid OWL DL conclusion is a valid OWL Full conclusion.

The code example below shows the OWL-DL code that represents Description 4.4 of a land-locked country as stated in Section 4.2.1. One can recognise the OWL language elements, as listed in Table 4.1.

```
<owl:Class rdf:ID="LandLockedCountry">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Country"/>
        <owl:Restriction>
          <owl:onProperty>
            <owl:ObjectProperty rdf:ID="hasBorderWith"/>
          </owl:onProperty>
          <owl:allValuesFrom>
            <owl:Class>
              <owl:complementOf rdf:resource="#Sea"/>
            </owl:Class>
          </owl:allValuesFrom>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
```

4.3 Ontology design and creation

The conceptualisation of a domain and the implementation of this conceptualisation into a machine ontology is not trivial. It requires a thorough understanding and careful formalisation of the domain semantics. As with other languages, the constructs of ontology languages facilitate multiple ways to formalise a single domain. Several methods of good practice can be followed to carry out the conceptualisation process, see for example [198]. Based on their recommendations and the experiences gained in this research (especially in the work reported in Chapters 5 and 6), the steps below are considered to be efficient and sufficient in building ontologies.

1. Determine the scope of the ontology.

2. Determine how it will be used; if for querying, what constitutes a typical query?

3. Create an ontology 'sketch' (on paper or with a software tool for drawing diagrams) containing the most general concepts and associations between them.

4. Create a list of all important concepts (at all detail levels) to be included.

5. Create the ontology concepts and their relationships with axioms in OWL with an ontology editor. Carefully consider the meaning of using subsumption relations, role restrictions, disjointness, etc. Repeat this step for any newly added concepts. Check every new axiom to see if the ontology remains consistent and is well-formed.

6. Instantiate the concepts with individuals.

UML can be used to make the ontology sketches in step 3 as UML supports the characterisation of subsumption, aggregation and other associations between concepts. UML's Object Constraint Language (OCL) may be used for further specification of concept constraints. However, the use of OCL may be overkill if an informal UML model is given to information engineers for further elaboration [11].

Ontology editors help the creator to apply the language constructors' syntax and semantics correctly. In the Protégé ontology editor (discussed in Section 8.2.1), some common ontology design patterns have been build into the user-interface, such as the creation of a *covering axiom* (see [118]) and the possibility to convert primitive classes into defined classes and vice versa. Apart from the ontology editors, dedicated tools exist for the verification of the correct syntax of the OWL constructs used in the ontology and the determination of the OWL sub-language (Lite, DL or Full), see for example the Wonderweb[5] and BBN[6] OWL validators. In some cases it is possible to apply semi-automatic creation of ontologies from information sources. This is known as *ontology learning*. Several approaches have reported positive results, see for example [177, 243].

In this thesis the 'Manchester House Style' [118, 237] has been followed as good practice for manual ontology creation. A brief summary of guidelines is listed below (taken from [237]):

1. Always paraphrase a description or definition before encoding it in OWL, and record the paraphrase in the comment area of the interface.

2. Make all primitives disjoint – which requires that primitives form trees.

3. Use the existential quantification ($\exists$) as the default quantification in restrictions.

[5]http://phoebus.cs.man.ac.uk:9999/OWL/Validator
[6]http://owl.bbn.com/validator/

4. Be careful to make classes defined - the default is primitive. The classifier will place nothing under a primitive class (except in the presence of axioms/domain/range constraints).
5. Remember the open world assumption (explained in Section 4.5.1). Insert closure restrictions if that is what you mean.
6. Be careful with domain and range constraints. Check them carefully if classification does not work as expected.
7. Be careful about the use of 'and' and 'or' (*intersectionOf*, *unionOf*).
8. To spot trivially satisfiable restrictions early, always have an existential quantification (∃) corresponding to every universal quantification (∀), either in the class or one of its superclasses (unless you specifically intend the class to be trivially satisfiable).
9. Run the classifier frequently; spot errors (e.g., inconsistent classes) early.

4.4 Ontology representations and notation

The vast number of relationships that can exist in an ontology, even with a relatively low number of classes, often poses problems of representation. A number of research initiatives have provided visualisation methods to support the building and communication of ontologies, see for example [5, 74]. In this thesis, ontologies are represented in various ways:

- **Concept hierarchies and role hierarchies** are used to indicate multiple subsumption relations. A concept or role hierarchy is represented with indentations for subclasses. An isolated full colon demarcates a group of classes that appears in the hierarchy but is not represented in the displayed list:

```
GeographicFeature
    Building
    :
    Country
        CountryWithCoast
        LandLockedCountry
    Sea
```

- **Description Logic statements** (also called *descriptions*) are used to indicate single subsumption relations, equivalence relations, conditions, etc.:

$$CountryWithCoast \equiv Country \sqcap \exists hasBorderWith.Sea \tag{4.11}$$

To avoid verbose pairs of existential quantifications and universal quantifications with similar elements, a new symbol is introduced in this thesis: ∃∀, which is a conjunction of ∃ and ∀. Its semantics are follows:

$$\exists\forall R.C \Longleftrightarrow \exists R.C \sqcap \forall R.C \tag{4.12}$$

For nested conditions, this yields:

$$\exists\forall R.(\exists\forall S.C) \Longleftrightarrow \exists R.(\exists S.C \sqcap \forall S.C) \sqcap \forall R.(\exists S.C \sqcap \forall S.C) \tag{4.13}$$

- **Ontology capture diagrams** are frame-based diagrams, generated with the Ontoviz software from the implemented OWL ontology (see also Section 8.2.2). They are used to visualise instance relationships and multiple roles, see an example in Figure 4.4.

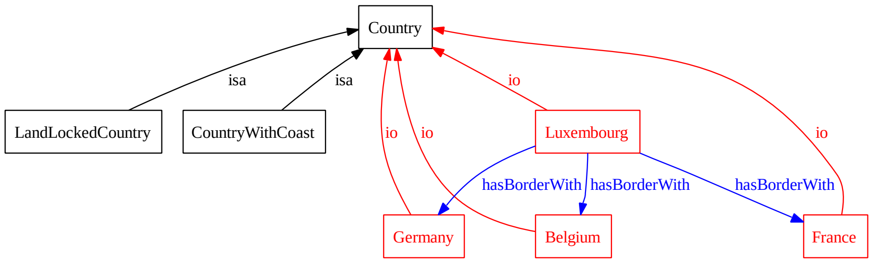


Figure 4.4: An example of an *ontology capture diagram.* In this diagram, a class is depicted as a box with black edges; An individual/instance is shown as a box with red edges. A black arrow represents an 'isa' (a subsumption) relation between classes; A red 'io' (instance of) arrow represents an instantiation relation between an individual and a class; A blue 'hasBorderWith' arrow is an example of a property (or role, in DL terms).

- **Venn diagrams** are used to visualise the containment of ontology classes with respect to their instances, see the example of Figure 4.5. This often clarifies the effects of subclassing, disjointness, overlap, role restrictions, etc. This way of visualising is also used in [118].

4.5 Reasoning with a knowledge base

A knowledge base can be seen as the set of axioms that represent an ontology. Reasoning with the concepts in an ontology is in fact performed with these axioms, based on logic-theoretical inferences.

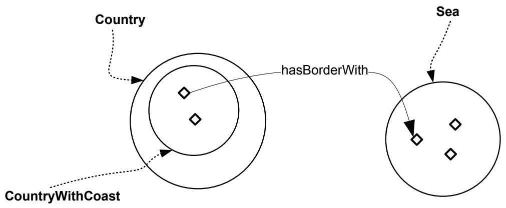


Figure 4.5: Example of a Venn diagram representing classes (circles), individuals (diamonds) and a property (arrow between individuals). Note: Individuals are called instances if their class membership is being emphasised.

4.5.1 TBox and ABox reasoning

A knowledge representation system based on Description Logics contains explicit knowledge, stored as a set of assertional axioms, and implicit knowledge, that can be made explicit by means of reasoning [16]. Such explicit knowledge is represented by logical inferences. The process of calculating inferences for the TBox(es) and ABox(es) of a knowledge base (as introduced in Section 4.2.1), is referred to as respectively TBox reasoning and Abox reasoning. An important part of the remainder of this section is based on [193] and [16].

TBox reasoning

Typical TBox inferences are defined below (literally from [16], page 62; the reader is referred to Section 4.2.1 for a definition of interpretation $\mathcal{I}$):

Let $\mathcal{T}$ be a TBox.

Satisfiability A concept C is satisfiable with respect to $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. In this case we say also that $\mathcal{I}$ is a *model* of C.

Subsumption A concept C is subsumed by a concept D with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.

Equivalence Two concepts C and D are equivalent with respect to $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.

Disjointness Two concepts C and D are disjoint with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$.

All of the above inferences can be *reduced* to either subsumption problems or unsatisfiability problems, which implies that reasoning mechanisms only have to check for either subsumption or unsatisfiability of concepts. The following statements provide the reduction to subsumption [16]:

C is unsatisfiable $\Leftrightarrow C$ is subsumed by $\bot$ (the bottom concept);
C and D are equivalent $\Leftrightarrow C$ is subsumed by D and D is subsumed by C;
C and D are disjoint $\Leftrightarrow (C \sqcap D)$ is subsumed by $\bot$.

The following statements provide the reduction to unsatisfiability:

C is subsumed by $D \Leftrightarrow (C \sqcap \neg D)$ is unsatisfiable;
C and D are equivalent $\Leftrightarrow (C \sqcap \neg D)$ and $(\neg C \sqcap D)$ are unsatisfiable;
C and D are disjoint $\Leftrightarrow (C \sqcap D)$ is unsatisfiable.

Reasoning mechanisms that check on subsumption should contain an intersection operation and an unsatisfiable concept. Those that check on satisfiability should, in addition, contain a negation operation. The latter group of reasoners make use of so called *tableau* algorithms. These algorithms follow a procedure of generating individuals and test their concept memberships. This is done by constructing a tree structure with nodes, containing individuals, labelled with the concepts of which they are assumed to be an instance [245]. In a process of breaking down concepts and generating new nodes, a tableau algorithm applies transformation rules to infer constraints and new tree nodes. If this process results in a so-called completion without a contradiction, then the concept is satisfiable. Details on the working of tableau algorithms are described in [16, 61, 122].

ABox reasoning

As described in Section 4.2.1, the ABox is filled with concept assertions and role assertions, which respectively define concept membership (individuals as instances of concepts) and role relationships between individuals. The basic task of ABox reasoning is *instance checking*: checking whether individuals are instances of concepts. Typical reasoning services that can be defined in terms of instance checking are [193]:

- Checking the *consistency* of an ABox, i.e., whether every concept in the knowledge base admits at least one individual.
- Realisation (finding the most specific concept of which an individual is an instance).
- Retrieval (e.g., finding the individuals that are instances of a given concept).

The consistency of an ABox is essential, because otherwise we could draw arbitrary conclusions from it. An ABox $\mathcal{A}$ is considered consistent with respect to a TBox $\mathcal{T}$, if there is an interpretation that is a model of both $\mathcal{A}$ and $\mathcal{T}$ [16]. ABox reasoning can be performed with the aforementioned tableau algorithms that have been extended to ABoxes, see for example [122].

Open World Assumption

An important aspect of DL knowledge bases (and OWL ontologies) is that they are based on the *Open World Assumption* (OWA), which has implications for the reasoning on them. OWA means that we cannot assume something doesnt exist until it is explicitly stated that it does not exist [118]. This implies that the absence of information in a knowledge base only indicates the lack of knowledge. In this respect, instances of DL concepts are treated differently as instances of database schemas [16]. A database instance is considered to represent one interpretation, a DL instance as part of its ABox, represents all its models. The following example clarifies that a DL reasoner uses this principle for computing its inferences. Consider the definition of a landlocked country, based on expression 4.4:

$$LandLockedCountry \equiv Country \sqcap \forall hasBorderWith.(\neg Sea), \tag{4.14}$$

and a country totally consists of land and not sea:

$$Country \sqsubseteq \neg Sea \tag{4.15}$$

We fill the ABox with instances by means of the following concept assertions:

$$\begin{aligned} &Country(Luxembourg)\\ &Country(France)\\ &Country(Belgium)\\ &Country(Germany), \end{aligned} \tag{4.16}$$

and role assertions that specify the border relationships for each country. In the case of Luxembourg, this is materialised with the following role assertions:

$$\begin{aligned} &hasBorderWith(Luxembourg,Belgium)\\ &hasBorderWith(Luxembourg,France)\\ &hasBorderWith(Luxembourg,Germany) \end{aligned} \tag{4.17}$$

Under the Open World Assumption, a reasoner will not identify Luxembourg as a landlocked country, unless we specify the number of boundaries a country has (by means of a number restriction). Otherwise, the set of role assertions in 4.17 is considered to be in an incomplete ABox.

In the same context, another issue has to be handled with care, i.e., whether or not the reasoner applies the Unique Name Assumption. This issue is described below.

Unique Name Assumption

If a reasoner applies the Unique Name Assumption (UNA), it assumes that all individuals are unique entities. In contrast, if a reasoner does not apply UNA, then two individuals with different names (e.g., 'Belgium' and 'Belgique', but also 'Belgium' and 'France') may be refer to the same individual, but they may also refer to different individuals [118]. This holds unless we explicitly state that whether they are the same or different. In the case of the landlocked country example, we have to perform these declarations for every country, otherwise the reasoner may infer that under certain axioms 'Belgium' and 'France' refer to the same country.

Tools

Research in the field of the Semantic Web has provided several DL reasoners of which currently KAON2, Cerebra, RacerPro and Pellet are the most commonly known (for specific information on these reasoners, the reader is referred to [244]. The reasoners basically differentiate in the DL they support (i.e., the supported expressiveness), whether they support OWL, their internal algorithms and whether they support TBox and/or ABox reasoning. A list of current reasoners is maintained at [244]. A comparison of reasoners has been carried out in [150], which also reports on the lack of support of nominals in current reasoners. Obviously, the performance of reasoners is an issue for large ontologies. It has to be noted that although OWL-DL is decidable, it is still difficult to reason with [121].

This research has opted for the RacerPro reasoner, due to its extensive support of both TBox and ABox reasoning and its OWL support and compatibility with the Protégé ontology editor. RacerPro implements the aforementioned tableau algorithms. It applies the Open World Assumption and by default it does not apply the Unique Name Assumption. More details on RacerPro can be found in Section 8.2.6.

4.5.2 Application of reasoning: matchmaking

Reasoning with a knowledge base is basically used for building and maintaining the knowledge base and for obtaining information from it [193]. More specifically, the information retrieval that is supported by reasoning services can be used (amongst others) to get answers to specified queries on the knowledge base. Sometimes such queries are called *semantic queries*, in order to distinguish them from database queries. The term *matching* or *matchmaking* is used in case knowledge structures are matched and/or partial matches are taken into account, or if queries are used for the (partial) matching of entities (e.g., documents, services, etc.) in a consumer-provider paradigm. Partial matching is a useful asset for environments in which (1) the searched items are composites (e.g., services) and/or (2) the searched items do not expose detailed meta-information and/or (3) the users are not likely to specify queries precisely (e.g., due to their unfamiliarity with the domain). Partial matching is a common way to increase the *recall* of search results (the proportion of relevant documents retrieved), see also [28]. Moormann Zaremski and Wing [189] emphasise that for example in software component matching it is rarely the case that we aim for an exact match between components, but rather a close one, and so, sacrificing *precision* (the proportion of retrieved documents which are relevant) for recall. Initial precise queries may also be *relaxed* for this reason. Obviously, on the other hand one would not want to loose too much on precision. Well-defined semantics of the model that underlies the information sources is an essential key to maintaining precision. Such a semantic model may be a domain ontology as part of a semantic interoperability framework (see Section 4.6). A common matching strategy as reported in [168] considers exact matches, disjoints and partial matches. Some approaches (for example [37, 241]) go further by defining semantic similarity of domain entity classes through quantitative identifiers. Recent research in information retrieval has embarked upon the development of specific ontology query languages. Examples are OWL-QL [73] and nRQL (see Section 8.2.6).

4.6 Semantic interoperability frameworks

Ontologies do not come as a single solution to a demand for information integration. They are typically embedded in a *framework* and an *infrastructure*. The definitions below are adaptations of the notions of *framework* and *infrastructure* which are used in [257] to characterise a specific information sharing approach. In this thesis, they are defined as follows:

Term 4.3 *A* semantic interoperability framework *is the combination of ontologies, their relationships, and methods for ontology-based description of information sources (services, data sets, etc.). The framework serves the semantic interoperability between information sources.*

A semantic infrastructure is defined as follows:

Term 4.4 *A* semantic infrastructure *comprises a semantic interoperability framework and the tools to maintain and use the framework as well as the (meta-)information sources that are produced with these tools.*

The application domain of the ontologies determines the application domain of the framework and infrastructure in which they are used. Examples of application domains for which semantic infrastructures are currently being built are document management, multimedia applications, medical and bio-informatics applications and human networks. Illustrative demonstration examples can be found in the yearly organised Semantic Web challenge [7].

Important aspects of building, maintaining and using a semantic infrastructure are:

[7]http://challenge.semanticweb.org/

1. Ontology creation and access
2. Ontology integration
3. Ontology-based description of information sources (annotation)
4. Reasoning-based information retrieval
5. Creation and use of ontology meta-information (information about ontologies)

Aspects 1 and 4 have been discussed in respectively Sections 4.3 and 4.5. The remaining aspects are covered in the remainder of this section.

4.6.1 Ontology integration

The process of combining information resources in a semantic interoperability framework is not trivial. It requires the careful evaluation of the effects on the interpretation of the combined content.

An ontology is typically built with a specific scope that suits an application domain. When it is used for information source integration, it plays the role of *global ontology*, representing the shared vocabulary of these sources [257]. Once the use of ontology-based information systems requires processing *across* application domains, ontologies are subject to integration. Stuckenschmidt and van Harmelen [257] identify two possible ways to use integrated ontologies as alternatives to the single ontology approach:

- In a *multiple local ontologies approach*, each information source is described by its 'own' ontology. Information integration is established by creating relationships between each pair of ontologies.
- The *hybrid approach* uses multiple ontologies as well, but in addition, it uses a global ontology with primitive concepts of the shared domain, in which the other concepts can be expressed.

In the Semantic Web, mutually inconsistent ontologies will be very common [45] and ontology integration has therefore recently received much attention in semantics research of which prominent examples can be found in [140]. Ontology *mapping* is key to many approaches. It is defined as follows (adapted from [140]):

Term 4.5 Ontology mapping *is the task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the mathematical structure of ontological elements (classes and their relationships) and their intended interpretations, as specified by the ontological axioms, are respected.*

Ontology mappings are used in two different forms of ontology integration, i.e., *alignment* and *merging*. Ontology merging is aimed at the creation of a single coherent ontology based on the source ontologies. Ontology alignment involves making the source ontologies consistent and coherent, while leaving them separated [199]. Prior to mapping, the process of alignment and merging involves the determination of:

- corresponding concepts among the source ontologies
- the set of overlapping concepts that are similar in meaning but have different names or structure
- concepts that are unique to each of the sources

A major challenge lies in the identification of suitable mappings that model these ontology relationships. Although it is expected that ontology mapping will not be fully automated for quite some years [57], recent research efforts have come up with successful semi-automatic methods [197]. We can distinguish three types of approaches, i.e., the use of lexical correspondences, the use of corresponding ontological structures (see for example [114]) and the evaluation of instance-pairs (an instance-pair is a pair between a class member in one ontology and a (semantically) corresponding class member in another ontology). The latter has been exploited by using formal concept analysis (FCA) in [239] and [258]. Instance-pairs may be drawn from user communities. In semi-automatic methods, a common approach is that an information engineer and a computer program iteratively suggest mappings.

Actual mappings may be created by introducing new concepts, stored in the source ontologies ('intra storage') or by means of an intermediate ontology ('extra storage'). Note that both intra and extra storage may serve both the above mentioned multiple local ontologies approach and hybrid approach. Currently, three groups of methods can be distinguished for the *representation* of mappings [197]:

- With *bridging axioms*, relationships are created between concepts and roles of the source ontologies. These relationships can be expressed by simple equivalence and subsumption relations, but can also involve role restrictions in more complicated bridges [62]. Examples of bridging axioms can be found in [196] and Chapters 5 and 6, e.g., Description 5.1.
- Instances (class members) are used to bridge classes between ontologies. Each instance in the source ontology is translated into the most similar instance in the target ontology [176].
- Mappings are expressed in first order logic with a query mechanism that is more expressive than current DL [45].

4.6.2 Ontology classification

The paradigm of using global and local ontologies (see Section 4.6.1) forms the design principle for so called *upper* ontologies, or also referred to as *top-level* or *foundation* ontologies. They provide primitive concepts at a general level, spanning multiple application domains. Prominent examples are Wordnet[8] [71] , SUMO[9] [194], Cyc[10] [166] and DOLCE[11] [182]. In contrast to upper ontologies, *domain* ontologies or also called *lower* ontologies have a narrower scope and provide more specific concepts. They may be built upon upper ontologies. Obviously, the notions of foundation and domain ontology are relative ones. One domain ontology may be the foundation for another ontology that contains more specific concepts. In addition, the generality difference between a foundation and domain ontology may vary from one framework to the other.

In addition to the above kinds of ontologies, a number of researchers [105, 265, 270] distinguish *task ontologies* and *application ontologies.* Task ontologies provide concepts about generic tasks and methods, which may apply to several domains. In such an ontology one may for example define a shortest path calculation that can be used in any kind of network structure. Similarly to domain ontologies, concepts in an a task ontology may specify certain concepts of an upper ontology.

Application ontologies are associated with a specific application (typically a software application) and draw upon the concepts of domain and/or task ontologies, and may directly or indirectly specify concepts of an upper ontology. For example in a railway application ontology, the concept 'shortest route' may be defined as the railway route between two train stations with a calculated shortest distance.

4.6.3 Semantic annotation and markup

Information resources are considered to be the 'carriers' of information in the infrastructure and may involve documents and services. The relationship between information resources and ontologies is made through a process called *annotation.* It can be seen as the creation of meta-information, using ontologies as reference frameworks. The term annotation can refer to both the process itself and the result of it (the relationships)[12]. More specifically, an annotation holds a link between a characteristic of an information source (e.g., an HTML or XML tag value, an object in an image, an entire image, etc.) and an ontology element (e.g., a class, class property, etc). The characteristic of the information source can involve an element of the *actual* information source or an element of a *meta*-information source. The annotation can be 'stored' in three ways:

1. If the information source (or meta-information source) is encoded with an anchor that links to the annotation or with an ontology reference, we speak of *semantic markup* (cf. [271]).

[8]http://wordnet.princeton.edu/

[9]http://www.ontologyportal.org/

[10]http://www.cyc.com/cyc/

[11]http://www.loa-cnr.it/DOLCE.html

[12]From a different viewpoint (i.e., towards the ontology) they are also referred to as respectively *registration* and *registration mappings*, see [38]

2. If an annotation is stored in the ontology, it will in this thesis be referred to as *registration* (cf. [38]).

3. If an annotation is stored in a third source, separate from, but holding identifiers for ontology and information source, it will in this thesis be referred to as a *registration mapping* following the same notion as in [38] and which has been applied in [173].

Depending on the 'extend' of the metadata, we speak of a certain certain *level of annotation*. For example, for HTML sources, such level may reach from scraping the presentation (the actual web page) to exposing the structure and context of the database or information structure that formed the basis for creating the web page. The inclusion of structure and context is referred to as *deep annotation* [110].

In some cases, the ontology that we would like to act as a reference framework, does not have suitable elements to bind annotations to. In such cases an *annotation template ontology* is used as intermediate between reference ontology and information sources (see an example in [117]).

Several annotation methods have been proposed in tools that annotate HTML files (e.g. SMORE [141], iAnnotate [221]) and Web Services (e.g. WSDL-S, see Section 4.7). Currently there is no standard for semantic annotation.

4.6.4 Ontology meta-information

In a setting where multiple ontologies are used, whether or not related in a framework, software agents and human users may require to analyse the characteristics of an ontology first, before using it for annotation, querying, etc. This can be done by abstracting the ontology content through creating ontology meta-information and abstract representations, including visualisations (for the latter, see Section 4.4).

Ontology meta-information can be used in registries for discovery purposes. The research efforts in this relatively new field are showing resemblance with the meta-information and registry mechanisms for information sources (see for instance RM-ODP in Section 2.4.2). An example of a registry for ontologies is *SchemaWeb*[13] and a semantic search engine called *Swoogle*[14] is currently under development by the University of Maryland. An ontology meta-information standard has been proposed in [112].

4.7 Semantic web services

On the crossroads of Semantic Web and web service applications we find approaches that make use of explicitly stated semantics of web service characteristics. In such approaches, the so-called *Semantic Web services* (SWSs) are considered to

[13]http://www.schemaweb.info/
[14]http://swoogle.umbc.edu/

be *semantically enriched* and support semi-automatic discovery, composition and execution. Currently, four prominent SWS approaches are OWL-Services (OWL-S) [181], Semantic Web Services Framework (WSMF) [285], internet Reasoning Service (IRS) [191] and Meteor-S [223]. OWL-S and WSMF (and its ontology WSMO) are current submissions under the World Wide Web Consortium Semantic Web Services Interest Group [283].

OWL-S, WSMF and IIRS have been compared by Cabral et al. [44]. An important conclusion of this paper is that the approaches are far from mature and that end-user applications based on them, still need a human in the loop. OWL-S and the WSMF ontology (WSMO) have been compared by Lara et al. [155]. They state that generally, OWL-S covers a wide range of applications and WSMO is more focussed on e-commerce applications. For solving heterogeneity problems, WSMO uses mediators, which are special services defined for that purpose.

In this research, OWL-S has been selected as starting point. Although the actual processes, described with the OWL-S model can become rather complex, the model itself has a clearly defined structure. It is embedded in OWL and it is thus well rooted in the well-established theoretical foundation of Description Logics. Further, OWL-S is generally considered to be more adaptive than the other approaches and implementation independent because it does not prescribe to use specific services. Finally, at the time of implementing the prototype, OWL-S was the most mature of the SWS approaches and was offering the necessary and sufficient constructs for modelling the characteristics of geo-services as targeted in this thesis.

Service matchmaking The application of matchmaking (see Section 4.5.2) to Semantic Web services has been subject to several research initiatives. Some approaches take into account the task that services perform, by using service profiles [168], others by additionally applying a task hierarchy [18, 46, 262] and/or by analysing the service's internal workflow [8, 235].

4.7.1 OWL-S

OWL-Services [181], or OWL-S in short, is an upper-ontology based on OWL that models the characteristics of Web services and which can be used to create semantically enriched Web Service descriptions.

OWL-S provides three modelling constructs at the top level, i.e., the service *profile* (what the service does), the service *grounding* (how the service can be accessed) and the service *model* (how to use the service in terms of semantic content, including its workflow). These three basic models are depicted in Figure 4.6. OWL-S provides classes that can be instantiated by a service provider to create specific service descriptions. This implies that such descriptions are expressed as OWL individuals in all three OWL-S sub-ontologies. Because OWL-S is an upper-ontology, it obviously does not provide domain ontologies. These have to

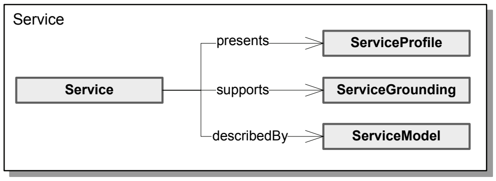


Figure 4.6: UML class diagram showing the overall service model of OWL-S (taken from [181]).

be established by service communities themselves. The remainder of this subsection concentrates on the process model, because it is used as process model in Chapters 5 through 8.

The process model of OWL-S (and its implementation in machine ontology) is based on principles that have been developed in previous work on process modelling, amongst others pi calculus, PSL and Golog [181] (see also Sections 3.2.1 and 3.2.2) . Figure 4.7 shows the basic structure of the OWL-S process ontology. Note that the service model class appears in both Figures 4.6 and 4.7. A participant in OWL-S is a client or a server. OWL-S supports the paradigm of IOPE parameters (Input, Output, Precondition and Effect). For preconditions and effects it uses local parameters and expressions, which can be declared in a specific language, such as SWRL (Semantic Web Rule Language).

A process can be one of three types, i.e., atomic, simple or composite. An atomic process is defined in [181] as *a description of a service that expects one (possibly complex) message and returns one (possibly complex) message in response. Atomic processes are directly invocable, they have no subprocesses and execute in a single step, as far as the service requester is concerned.* Composite processes can be decomposed into other composite or non-composite processes. A simple process is an abstraction of an atomic or a composite process.

The way in which a composite process is constructed, is depicted in the middle part of Figure 4.7. A control construct can be any of nine control flow types. These types can be directly related to the workflow patterns defined in [1]. A composite process is constructed in the form of a tree with branches following either ‘first’ or ‘rest’. The ‘Perform’ construct is used to instantiate the branches (i.e., all branches and leaves are expressed as OWL individuals). An example of such a tree is discussed in Section 6.2 and depicted in Figure 6.6.

Shorthand notation

For the sake of an effective human communication one needs to represent the control flow of an OWL-S composite service in a comprehensive way. The OWL encoding itself is quite verbose and therefor not very suitable. Better options are

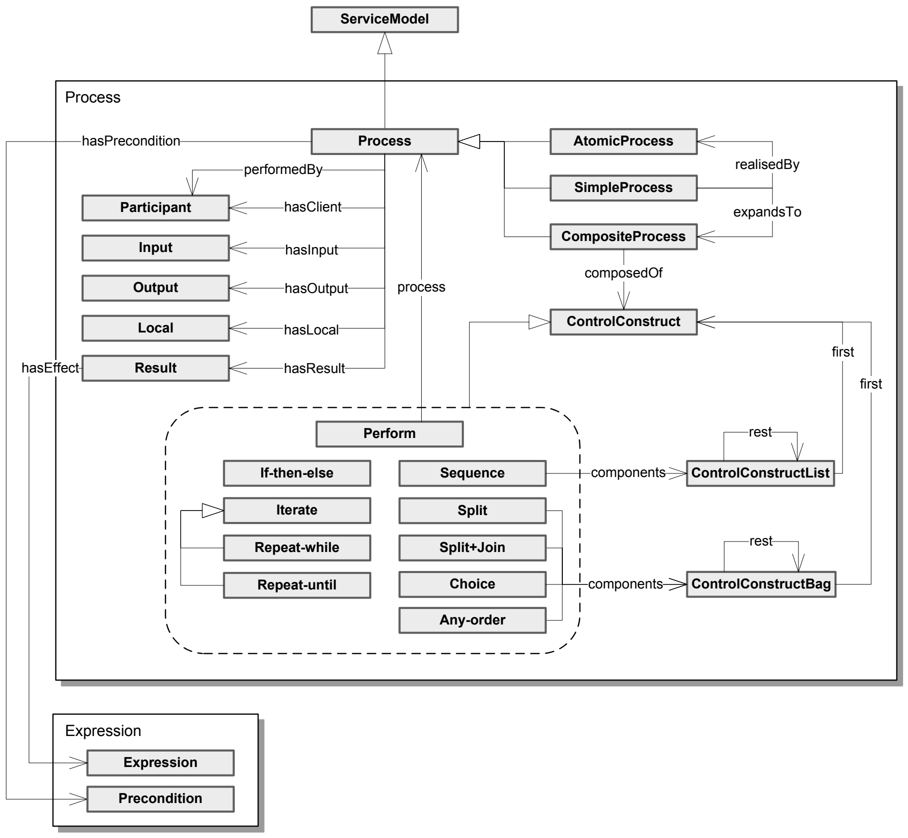


Figure 4.7: UML class diagram showing the process model of OWL-S (adapted from [181] and [19]).

to (1) visualise the control flow in a graph or (2) use a shorthand notation of the most important control flow elements. The current version of OWL-S (1.1) does not provide such a shorthand notation. This thesis proposes an XML style notation similar to the one used in [289] for the representation of WSBPEL control flow . The example below denotes a process sequence that first performs Service1 of type ServiceTypeA and then performs Service2 of type ServiceTypeB and Service3 of type ServiceTypeC in parallel:

```
<Sequence>
    Service1 : ServiceTypeA
    <SplitAndJoin>
        Service2 : ServiceTypeB
        Service3 : ServiceTypeC
    </SplitAndJoin>
</Sequence>
```

This notation uses the control constructs (sequence, split, etc.) from OWL-S, but because these (or semantically equivalent) constructs are commonly used in other contexts, its application is not necessarily restricted to OWL-S.

4.7.2 Semantic annotation of web services: WSDL-S

Semantic annotation is used for linking an information source to an element in an ontology (see Section 4.6.3). Semantic annotation of *web services* will support web service discovery and composition. A standard for the annotation of web services has been proposed by Akkiraju et al. [4] and has been submitted to the World Wide Web Consortium Semantic Web Services Interest Group [283]. WSDL-S specifies the placement within the WSDL file of specific tags with prefix 'wssem:', referring to the namespace *http://www.ibm.com/xmlns/WebServices/WSSemantics*. The following tagging example specifies that one of the output parameters of the service at hand is a (geographic) point:

```
<wsdl:message name="getCoordinatesResponse">
    <wsdl:part name="output" element="xsd1:ResponseType"
    wssem:modelReference="Ontology0#OP_Point"/>
</wsdl:message>
```

The definition of 'Point' is given in 'Ontology0', which is defined in the same WSDL file as follows:

```
xmlns:Ontology0="http://geoserver.itc.nl/lemmens/owl/ontogeo.owl"
```

This annotation example is further elaborated upon in Section 5.6.2.

4.8 Geo-semantic modelling and spatial relevance

We should ask ourselves whether the semantic modelling of geo-services is essentially different from semantic modelling in general. Section 1.1 has already provided some introductory background on this issue. The essential question is whether the information retrieval process has a spatial relevance, as pointed out by [257]. In fact, it has with respect to several aspects. Although the research work of this thesis does not embark upon spatial reasoning, the most prominent aspects and consequences for semantic modelling are highlighted below.

- Geo-information is meant to exhibit spatial relationships between features. These relationships are used for spatial analysis in GIS by computations on topology and metrics of geometries. Currently, ontology languages such as OWL do not contain specific constructs that model spatial relationships. As a workaround, common practice is to specify roles with a spatial connotation, such as 'Touch', 'Overlap' and 'North'. Examples can be found in [13, 280]. Another alternative is to outsource the spatial analysis to conventional computational solutions.

- Geo-information is multi-dimensional. The integrated spatio-temporal and thematic aspects of geo-information contribute to its importance in a wide variety of application domains. At the same time, the interrelation of aspects introduces complexity of reasoning over multiple aspects when geo-information is involved. Geo-services make use of operations that may act on all dimensions of geo-information. For a proper understanding of the functionality of those operations, it is important to identify these dimensions in an operation.

- Geo-information is characterised by multiple-representations. Typical for geographic information (and geo-services) is that it is very common for a geographic phenomenon to take many different feature representations in multiple or even a single geodata set. Representations may differ in terms of spatial, temporal and thematic attributes. Such attributes may vary also along different levels of generalisation and aggregation (as explained in Section 3.1). Reasoning about geo-services has to take into account these aspects, specifically with respect to the meaning of its input and output parameters.

- Geo-services often depend on tightly-coupled geodata. Tightly-coupled data (defined in Term 2.8 , Section 2.1.1) determines the validity area of a service and may contain (part of) the semantics of the service's input and output parameters.

The above aspects have been taken into account in the implementation of the proposed semantic interoperability framework described in Chapter 5 and determine the essence of the reasoning about geo-information and services as deployed in Chapter 6 and 7.

4.9 Summary and reflection

This chapter has provided an overview of semantic modelling paradigms, mainly as a result of recent developments in the Semantic Web. Semantic modelling in this field centres around the creation and reasoning with machine ontologies, represented in OWL. Aspects have been highlighted that are considered to be important for building a semantic interoperability framework for geo-services in the context of the use cases, described in Section 2.6. Several of these aspects are rather domain-generic than geo-specific, such as ontology creation and reasoning. The geo-specific aspects involve the handling of spatial relationships, multiple-dimensions, multiple-representation of geo-information and the geodata that may be tightly-coupled to a geo-service. More geo-specific implementation issues will become apparent in Chapter 5.

Chapter 5

Semantic interoperability framework for geo-services

This chapter describes the basic elements of a semantic interoperability framework for geo-services, based on the principles of semantic interoperability frameworks, discussed in Section 4.6. It describes the framework of formal ontologies and the way they support the characterisation of geo-information and geo-operations for the purpose of machine reasoning. The formal semantics of information and operation concepts is expressed in Description Logic axioms and implemented as machine ontologies in OWL. Section 5.1 introduces the basic parts of the framework. The ontological implementation of the framework is described in Sections 5.2 through 5.5. Section 5.2 presents *feature symbols* as the basic constructs for describing geographic features. Section 5.3 provides examples of geographic domain ontologies that make use of feature symbols. Section 5.5 presents an ontology for geo-operations. Section 5.6 elaborates upon the creation of service descriptions, based on the presented ontologies.

5.1 Semantic framework overview

This section gives an overview of the semantic framework and its ontologies that we apply for the modelling of geo-information and geo-operations as contained in geo-services.

The ontology design follows the principles as discussed in Section 4.3. The scope of the ontology framework covers both geo-information and geo-services. For demonstration purposes, four sub domains have a more specific scope. They comprise respectively the Dutch generic geo-information domain (NEN3610), the Dutch topographic mapping domain (TOP10NL), a risk mapping domain and a travel domain.

The ontology framework will be used for geo-service discovery and service chaining, as discussed in Chapter 6, by means of ontology queries. Typical queries that will be posed to this ontology framework are:

- Find all operations that match a set of input and/or output parameter types.
- Find all operations that fit an existing service chain with respect to their input and/or output parameter types.
- Find all operations that are composed of operations that instantiate a given set of operation types.
- Find all information/service concepts that are sub- or superclasses of a given concept.
- Find all data sets that contain a specific feature type (e.g., *Building*).

UML is used for sketching an overview of each ontology. OCL is not used, because concept constraints can be directly entered in the ontology without the need for an intermediate documentation step. This would have made sense if the design and implementation of the ontologies were separated, but in our case they are not.

The proposed framework is depicted in Figure 5.1. It contains basic artefact types such as data and operations, as defined in Chapters 2 and 3, and will be elaborated upon in the sections below. Section 5.1.1 gives an overview of the information model and operation model used. Section 5.1.2 introduces the ontologies that implement the framework elements. Details of the information models, operation models and ontologies are discussed in Sections 5.2 through 5.5.

5.1.1 Information and operation model

Figure 5.1 depicts how feature types (e.g. *Building*, *River*) are described together with the operations that act on them. The figure contains classes, indicated with shaded boxes and instances (or 'objects'), indicated with open boxes. The figure is divided into two vertical parts, i.e., one that contains *information* elements and one that contains *operation* elements.

Information part In line with the ideas of OGC and ISO, in this model we consider the fundamental unit of geographic information to be a *feature*. A feature is defined in ISO 19101 (Reference) [128] as follows:

Term 5.1 *A feature is an abstraction of real world phenomena.*

The definition intentionally uses the term phenomena in plural form. The fact that features may be defined recursively, causes them to exist on different levels of generalisation and aggregation [203]. Examples of features are: a road, the connection between two roads, a satellite image, etc.

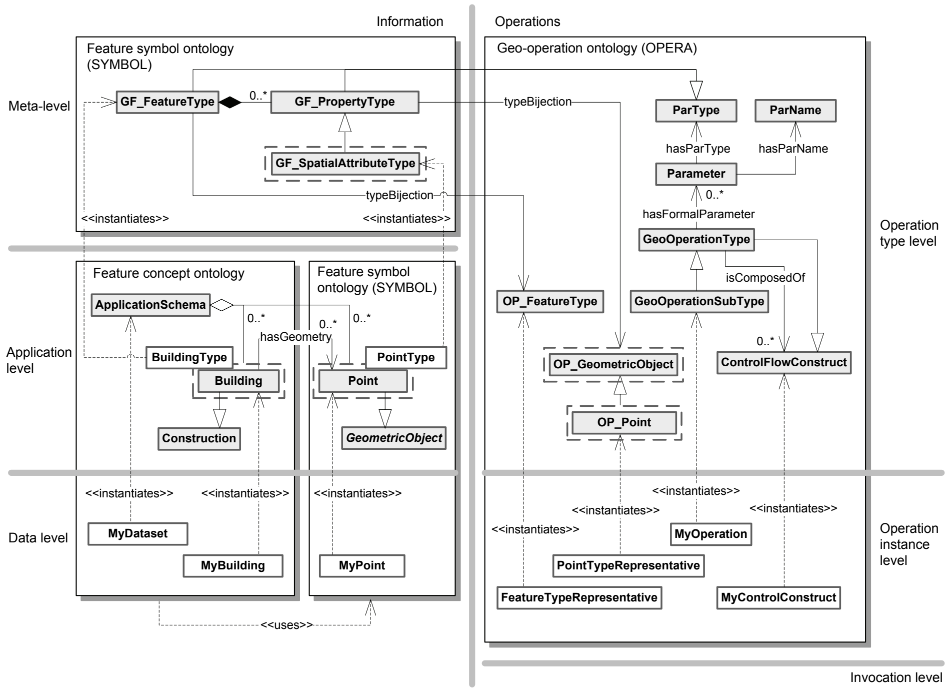


Figure 5.1: UML class diagram, depicting an overview of the proposed semantic interoperability framework for geo-information and geo-operations. Shaded boxes represent classes, open boxes represent instances. Boxes with a dashed line contain an example class.

The information part is divided in three horizontal levels, which represent a meta-level, an application level and a data level, according to the architecture layers, used in the *Conceptual Schema Modelling Facility* of the ISO 19101 standard (Reference) [128]. The meta-level contains classes of the ISO General Feature Model (GFM) as described in ISO 19109 (GFM) [131]. These are metaclasses that are used for the classification of features (abstractions of real world phenomena) and the specification of their relationships. The GFM makes use of the metaclass 'GF_FeatureType' and metaclasses for feature properties (feature attributes, feature behaviour and associations between features).

At the application level, an instance of the class GF_FeatureType is a specific feature type that represents a class of real world phenomena, e.g., *Building.* At this level, a feature is given specific attributes, such as *Point* as instantiation of a spatial attribute type.

At the data level, we find instances of feature types such as *Louvre* and *My-Building.*

Metaclasses In a system with metaclasses, such as the one presented here, a class can also be seen as an object [58]. However, currently, Description Logics does not provide a facility to treat classes as objects [34]. As a workaround, a so-called meta-individual can be created, which is related to the class by, for example, naming convention. Figure 5.1 shows the meta individual *BuildingType* and its associated class 'Building' as two attached boxes. The combination of these two boxes is also called a *two-facetted construct*, containing a *class facet* and an *object facet* [58, 224]. The relation between class (e.g., *GeometricObject*) and metaclass (*SpatialAttributeType*) can be made explicit by means of a *materialisation* abstraction pattern as presented in [224, 225].

Operation part The operations part of Figure 5.1 is divided in an operation type level, an operation instance level and an invocation level. Operation types are part of an operation type hierarchy and their input and output parameter types are the metaclasses, defined in the information meta-level. More specific operation definitions require the definition of operation parameters also in terms of classes such as 'Point'. However, associations between parameter types and classes at the information application level would not be proper, i.e., this would imply that *MyPoint* (which is an actual point with coordinates) would be an instance of *ParType.* For this reason, the class structures at the information application level are copied to the operation part. Their elements are labelled with an 'OP' prefix. The *typeBijection* relationship is introduced as a bridge between the information meta-level classes and these 'OP' classes. This bridge can be seen as a *materialisation* between a class (e.g., *OP_Point*) and a metaclass (*GF_SpatialAttributeType*) in which the class has only one instance (*PointTypeRepresentative*). In this way, a specific operation instance (*myOperation*) can be associated to an individual-representative of a parameter type.

5.1.2 Framework ontologies

At the basis of the framework, there are three types of formal ontologies: feature concept, feature symbol and geo-operation ontology. They are briefly introduced below. Specific elements of the three ontologies are discussed in more detail in Sections 5.2 through 5.5.

- A *feature concept ontology* formally defines the conceptualisations of real-world phenomena and the relationships between them. This is done at the formal concept world, as shown in Figure 3.1. Examples of elements in the feature concept ontology are *Building* and *ConstructionMaterial.* The elements of a feature concept ontology make part of an application schema (the conceptual schema for data required by one or more applications [128]). An element in the feature concept ontology is defined by the relationships with other elements in the feature concept ontology (for example, *Building* is a subclass of *Construction*) and by the relationships with elements contained in the feature *symbol* ontology (for example, 'Building' is represented in a particular application schema by a *Point.* The latter relationship is indicated in Figure 5.1 by *hasGeometry* for the specific example of 'Building'. Generally speaking, the feature concept ontology 'uses' the classes of the feature symbol ontology for its definitions.

- A *feature symbol ontology* formally defines the elements that make up a feature at the symbol world level and the relationships between them. The term 'symbol' does not necessarily refer to a visual symbol, but rather to a semantic symbol, cf. Figure 3.1. Examples of these elements are *GF_FeatureType* (at the meta-level) and 'Point' and 'Enumeration' (at the application level). A feature symbol ontology may also contain instances at the data level. For example, an instance of the class *Point* is an actual point with coordinates.

- A *geo-operation ontology* formally defines types of operations in terms of their behaviour. Each type is characterised by the behaviour of one out of a set of well-known atomic GIS operations and their typical input and output parameters. The input and output parameter types are described by referring to elements from the feature symbol ontology. For example, these elements may indicate that a service needs thematic attributes to function properly (and, for example, does not need spatial attributes). Further, for composite operations, the ontology contains control flow elements, such as *Sequence* and *Choice.*

Each of the above ontology types may be materialised by a specific ontology or a combination of ontologies. A specific feature symbol ontology may reflect how a specific set of services of a software manufacturer handles its geographic feature representations. Another feature symbol ontology may implement standardised geographic feature representations, such as defined in ISO or OGC specifications.

Similarly, geo-operation ontologies may be manufacturer-dependent (*proprietary*) or contain standardised elements. With respect to feature concept ontologies, we do not differentiate between proprietary and standardised ones, but rather between generic ontologies and domain-specific ones. The application domain that a semantic framework has to serve, requires the feature concept ontology to include a particular scope. For example, a semantic framework that serves the travelling domain requires a feature concept ontology that defines travelling concepts. A feature concept ontology is typically built by experts within information communities and tends to have a limited scope. An information community is defined in [207] as follows:

Term 5.2 *An* information community *is a collection of people (a government agency or group of agencies, a profession, a group of researchers in the same discipline, corporate partners cooperating on a project, etc.) who, at least part of the time, share a common digital geographic information language and common spatial feature definitions.*

The integration of ontologies can be a substantial effort in building the semantic framework. In this thesis, different feature concept ontologies and geo-operation ontologies are used to demonstrate such integration. These ontologies make use of the elements of one generally applicable symbol feature ontology, which is based on the ISO 19100 series of standards.

In terms of the ontology classification provided in Section 4.6.2, a feature concept ontology and a feature symbol ontology are both considered to be domain ontologies. A geo-operation ontology falls under the category of task ontologies. Application ontologies are not part of the interoperability framework described in this chapter. They are considered to contain the specific concepts that describe the data and services in the use case applications in Chapter 7. The relationships between the different types of ontologies is addressed in Section 8.1.2.

Each ontology has been implemented with an individual namespace, such as *http://geoserver.itc.nl/lemmens/owl/symbol.owl*, and its concepts appear, once imported in another ontology, with a corresponding prefix, such as *symbol:*. For the sake of carrying out experiments in the prototype (described in Chapter 8), all ontologies are integrated in one 'super'-ontology, called *OnToGeo*. However, apart from their interrelationships, they may be considered as individual ones in terms of design, storage and maintenance. This is the reason that they will be treated individually here. A detailed description of the concepts within each ontology is given in Sections 5.2 through 5.5.

5.1.3 Relations with ISO specifications

The framework, introduced in Sections 5.1.1 and 5.1.2, is based on a number of ISO standards for geo-information. They are briefly introduced here, and discussed in more detail in the remainder of this chapter. The ISO 19131 standard for data product specifications [137] provides guidelines for creating data product specifications (such as those of national mapping agencies) in terms of other existing ISO specifications. Application schemas and feature catalogues are given a central role in representing the content of data. An application schema defines the data structure and the data content in accordance with ISO standard 19109 (GFM) [131]. The application schema implements a feature catalogue, which provides the semantics of feature types, their attributes, attribute values, feature behaviour and relationships between features. Specifications for creating a feature catalogue are described in ISO standard 19110 (Catalog) [133]. Both the ISO models for application schema and feature catalogue draw upon the rules given in the ISO General Feature Model (described in ISO 19109 [131]). The new Dutch geo-information model NEN3610 (version 2) is following the above ISO standards. Apart for application schema and feature catalogue, the ISO 19131 (Product) further specifies the inclusion of other elements such as data delivery parameters and data quality parameters.

A feature symbol ontology as introduced in Section 5.1.2 can be considered as a representation of a feature model. In this thesis, it implements the ISO General Feature Model. A feature concept ontology represents a feature catalogue. This may be a feature catalogue conforming to the ISO 19110 (Catalog) standard. The ISO 19119 (Services) standard is relevant with respect to operation metadata and operation type classification.

5.2 Feature symbol ontology

For the purpose of this thesis, a feature symbol ontology is developed in OWL. In the remaining text it will be denoted as SYMBOL. To support the integration mechanism depicted in Figure 5.1, this formal ontology was designed with two design criteria in mind:

- To serve as upper ontology for feature concept ontologies, allowing classes of the latter to use feature symbol concepts.
- To support the description of input and output parameters of geo-operations (as defined in operation ontology OPERA). Copies of the classes in SYMBOL appear as 'OP' classes in OPERA.

SYMBOL is based on ISO standards, in particular on the ISO 19109 (GFM) standard. The conceptual models that form the basis for the ontology are depicted as UML diagrams in Figures 5.2 through 5.4. SYMBOL deviates from ISO 19109 (GFM) and related standards, in the following aspects:

- Some simplifications were introduced to reduce the overall complexity of the model. For example, the model in this thesis follows the simple feature model of ISO 19125 (SFeature). It does not implement the topological model as described in ISO 19107 (Spatial)—it only refers to topological objects as 'black box' entities— nor does it include 3D objects. Another limitation is put on coverages. The model only includes Quadrilateral grid coverages (also known as raster images) as specified in ISO 19123 (Coverage).

- Some additions to ISO 19109 (GFM) were thought useful, such as an identification of different subclasses of association roles (relationships between feature classes).

By convention, classes in ISO standards are denoted with bialphabetic prefixes. ISO 19109 (GFM) uses GF_ (General Feature Model) for most of its classes. Some classes in 19109 (GFM) stem from other standards, such as GM_Object (denoting geometry model, ISO 19107 (Spatial)) and SC_CoordinateReferenceSystem (from ISO 19111 (RefCoord)). The following classes, appearing in Figures 5.2 through 5.4, do not have ISO prefixes. They have been assigned prefixes specific to this thesis:

- Prefix 'SF_' for classes of the Simple Feature Model (ISO 19125),
- Prefix 'SY_' (*Symbol*) for classes that are additions to the ISO model in the context of this thesis work.

The following sections describe the feature model with help of UML class diagrams.

5.2.1 Feature overview

The basis of this model is formed by the *feature* concept. In the ISO General Feature Model, the term 'Feature' is used to denote 'GF_FeatureType' at the meta-level, or a specific feature type at the application level or an instance of a feature type at the data level. In the ISO model, a feature is not necessarily associated with a spatial object or field. A person is also considered an example of a feature. A feature type has three kinds of properties (see Figure 5.2), i.e., attributes, behaviour (possible operations that can act upon the feature type) and associations with other feature types. In the feature symbol ontology these properties are implemented with the following role hierarchy:

```
hasProperty
    hasAttributeType
    hasFeatureBehaviour
    hasAssociationRole
```

with domain and range defined for each role. For example, the domain of *hasAttributeType* is the class *GF_FeatureType* and its range is *GF_AttributeType*.

Spatial associations and topology The class GF_AssociationType is a meta-class of which the instances themselves are associations. An example of an association is 'Road touches Crossing', which is a spatial (or more specifically, a topological) association between two feature types. GF_SpatialAssociationType, as subclass of GF_AssociationType has a role, represented by the class GF_AssociationRole. In the example above, the association can be instantiated by the *link*

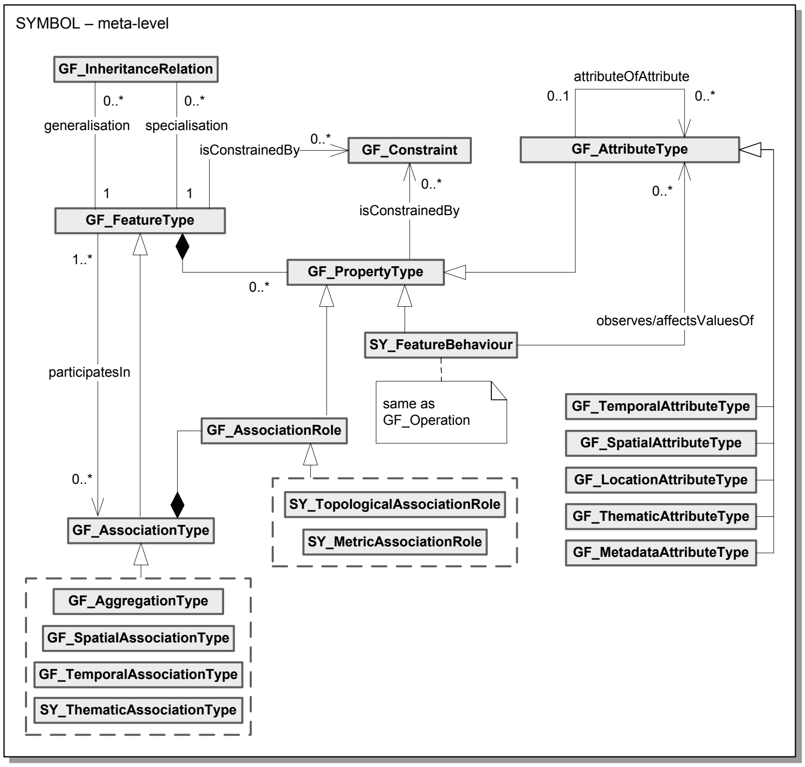


Figure 5.2: UML class diagram showing the overview structure of the feature symbol ontology (SYMBOL), based on the ISO general feature model (ISO 19109).

‘FourthAvenue touches UnionSquare’. In the General Feature Model, spatial associations between feature types can be realised explicitly by such association types, but also implicitly by means of topological primitives of their geometric representations. The mechanisms for topological primitives are provided in ISO 19107 (Spatial) [129].

Value types Feature attributes are classified as spatial, location, temporal, thematic or metadata attribute types. An attribute type is constrained by its value type. Thematic attributes typically take values of strings, integers, floats or enumerations. Spatial attributes have geometric or topological objects as their value types, according to their definitions in ISO 19107 (Spatial) and ISO 19125 (SFeature). Location attributes have geographic identifiers as their value types, as defined in ISO 19112 (RefIdent), which is typically a name, an address or a reference to another feature type.

Domain of values A value type is restricted by its *domain of values*. For example, the domain of values of an enumeration is the set of enumeration values; the domain of values for a geometric object is known as its *envelope* [129].

5.2.2 Feature geometry

For feature geometry, we apply the model of ISO 19125-1 (SFeature). Figure 5.3 shows the ISO 19125-1 model within the dashed boundary. The classes outside this boundary are added in order to support specific operation descriptions such as interpolation (see their description in Appendix C). ISO 19125 (SFeature) is

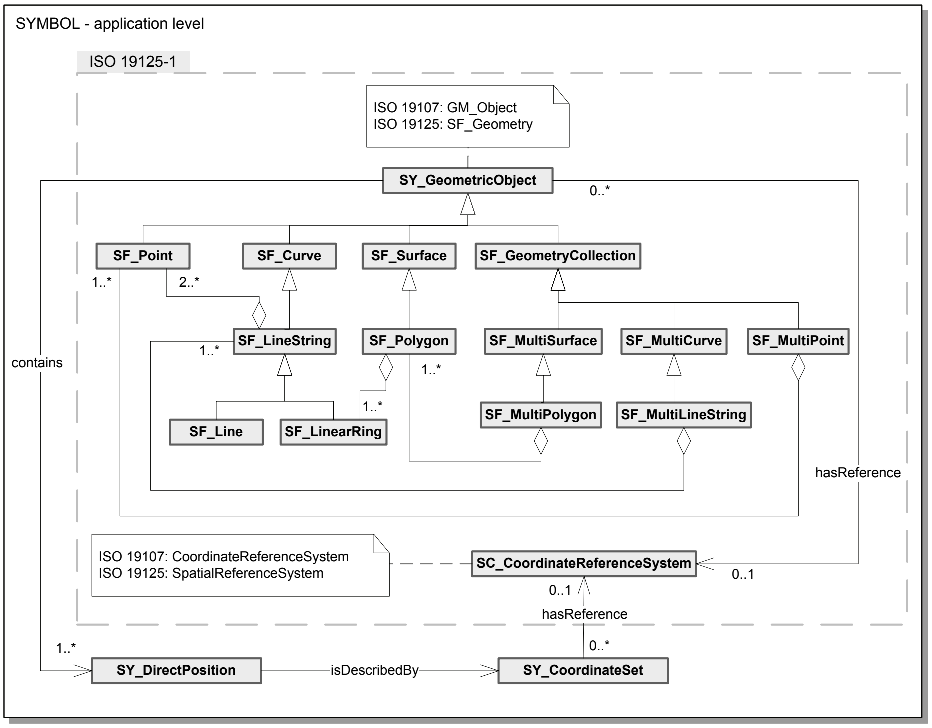


Figure 5.3: UML class diagram showing the feature geometry model of the feature symbol ontology (SYMBOL). Figure partly adapted from ISO 19125-1 (Simple feature access) [135].

a limited model compared to ISO 19107 (Spatial), because it does not support topological objects nor 3D objects. The ISO 19125 specification [135] contains mappings for its classes to corresponding ISO 19107 classes.

5.2.3 Thematic attributes

Thematic attributes represent the descriptive characteristics of a feature other than the spatial, temporal, location or metadata attributes [131]. Attributes in general are constrained by their value type. Value types may be basic data types such as CharacterString, Integer, Boolean, Date and Enumeration or specific measurement types such as Angle or Time. Data types are defined in ISO 19103 (Concept) [136] and GML [?, ?]. On a more abstract level, thematic attributes are distinguished in terms of levels of measurement. The following classification is used to distinguish the parameters of geo-operations as described in Appendix C.4.3:

- Nominal: Measurements are classified into named groups. Example: a 'land use' attribute with values 'Forest' and 'ArableLand'.
- Ordinal: Ordinal measurements have the properties of nominal measurements. In addition, measurements can be ordered. Example: a 'suitability' attribute with values 'High', 'Medium' and 'Low'.
- Interval: Measurement is done with respect to a quantitative scale with a fixed interval and an arbitrary zero reference. The operations *add* and *subtract* can be meaningfully performed on interval measurements. Example: (1) a 'calender year' attribute with example value '2000' (representing the period 1-1-2000 till 31-12-2000) , (2) a 'Celsius temperature' attribute with example value '2 degrees' (representing the interval 1.5 - 2.5).
- Ratio: Attributes can be measured with respect to a quantitative scale with a fixed zero reference. Ratio measurements support the operations *multiplication* and *division*. Example: a 'distance' attribute with values '250 metres', '330 metres', '1000 metres'. Ratio measurements may be re-scaled, e.g., from miles to metres.
- Count: A measurement is expressed as the count of a number of objects. Counts are not nominal, nor ordinal. They cannot be re-scaled. Example: a 'population count' attribute with values '15234', '120505', '1240709'.
- Absolute: A ratio measurement becomes absolute if the unit of measurement is not arbitrary [50]. They cannot be re-scaled. Example: a 'probability' attribute with values '0', '0.5', '1'.

The nominal, ordinal, interval and ratio levels originate from research by Stevens [254]. The absolute and count levels were introduced as extensions to Stevens' levels by Chrisman [50]. Figure 5.4 shows the above classification in relationship with SYMBOL. Specific data types can be made subclasses of the more abstract ones shown in the list above.

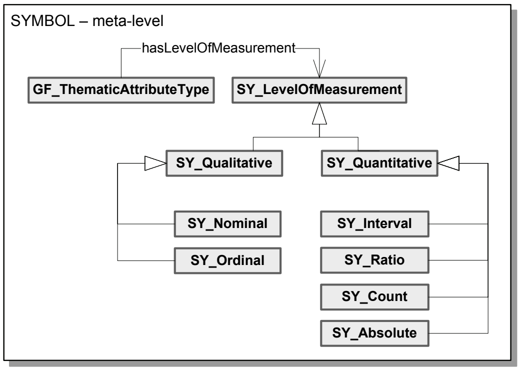


Figure 5.4: UML class diagram showing the Classification of thematic attributes in the feature symbol ontology (SYMBOL).

5.2.4 Coverages

In the ISO 19124 standard (ImgComp) [134] a coverage is defined as follows:

Term 5.3 *A coverage is a feature that acts as a function to return values from its range for any direct position within its spatial, temporal, or spatiotemporal domain. In other words, a coverage is a feature that has multiple values for each attribute type, where each direct position within the geometric representation of the feature has a single value for each attribute type.*

A direct position is defined as a position described by a single set of coordinates within a coordinate reference system. The (spatio)temporal domain of a continuous coverage consists of a set of direct positions defined in relation to a collection of (spatio)temporal objects [134]. A coverage can be seen as a feature collection (which is a feature itself, so it does not violate the definition of coverage). The grid cells of a quadrilateral grid coverage (commonly known as a raster image) are the individual features that carry feature properties, such as colour values or classified raster values. The conceptualisation of coverage as described here allows single grid cells as well as a complete coverage to be subject to geo-operations as defined in Section 5.4.

5.2.5 Example usages of feature symbols in feature concepts

Table 5.1 provides some examples of features, feature properties and feature property values, making use of the constructs in the SYMBOL ontology that is described in Sections 5.2.1 through 5.2.4.

Feature (all GF_FeatureType at meta-level)			Feature property			
Example feature type (applic. level)	Example feature instance (data level)	Data structure	Example GF_Feature-PropertyType (meta-level)	Example feature property (applic. level)		Example feature property value(s) (data level)
				Property name	Value type	
Building	Louvre	Object	GF_Thematic-AttributeType	hasBuilding-Function	Enumeration	Museum
			GF_Spatial-AttributeType	hasGeometry	Geometric-Object.-Polygon	Actual polygon representing the feature instance with populated coordinates
Satellite image	Aster ID: AST_L1B.003: 2023061715	Quad. grid coverage	GF_Thematic-AttributeType	hasGreyValues	Array:Integer	Array with actual grey values [0..255]
			GF_Thematic-AttributeType	hasLanduse	Enumeration	[Forest, ..,Urban]
Satellite image grid cell	Grid cell row=50, column=1521	Quad. grid cell	GF_Thematic-AttributeType	hasGreyValue	Integer	255
			GF_Thematic-AttributeType	hasLanduse	Enumeration	Forest
Road touches Crossing (GF_Spatial-Association-Type)	FourthAvenue touches Union-Square	Object	GF_Association-Role.SY_Topo-logical-Association-Role	represents-Topological-Association	Topological relationship	Touches (link between two feature instances)
			GF_Spatial-AttributeType	hasGeometry	Geometric-Object	Actual geometric object (point) that represents the 'touches' association
Buffer zone	Hazard zone around Borsele nuclear reactor	Object	GF_Association-Role.SY_Metric-Association-Role	represents-Distance-Association	Distance	50 kilometres (link between two feature instances)

Table 5.1: Examples of elements in the feature symbol ontology (SYMBOL).

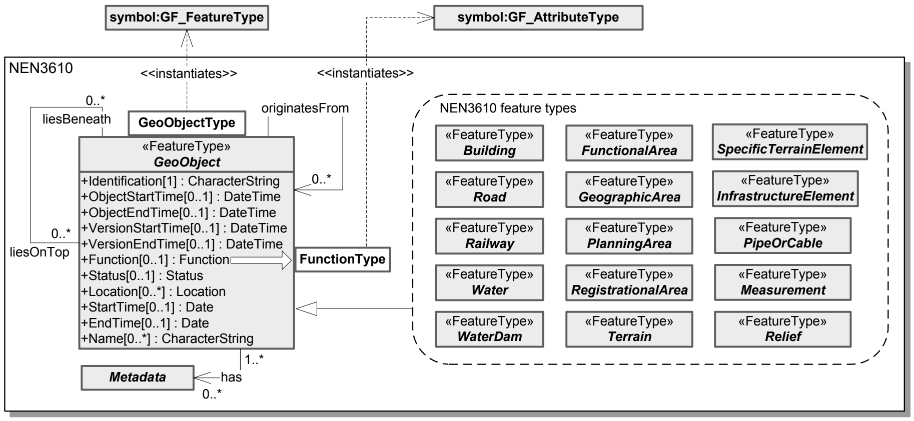

symbol:GF_FeatureType
symbol:GF_AttributeType
NEN3610
<<instantiates>>
<<instantiates>>
0..*
liesBeneath
GeoObjectType
originatesFrom
«FeatureType»
GeoObject
+Identification[1] : CharacterString
+ObjectStartTime[0..1] : DateTime
+ObjectEndTime[0..1] : DateTime
+VersionStartTime[0..1] : DateTime
+VersionEndTime[0..1] : DateTime
+Function[0..1] : Function
+Status[0..1] : Status
+Location[0..*] : Location
+StartTime[0..1] : Date
+EndTime[0..1] : Date
+Name[0..*] : CharacterString
0..*
liesOnTop
0..*
FunctionType
1..*
Metadata
has
0..*
NEN3610 feature types
«FeatureType» Building
«FeatureType» FunctionalArea
«FeatureType» SpecificTerrainElement
«FeatureType» Road
«FeatureType» GeographicArea
«FeatureType» InfrastructureElement
«FeatureType» Railway
«FeatureType» PlanningArea
«FeatureType» PipeOrCable
«FeatureType» Water
«FeatureType» RegistrationalArea
«FeatureType» Measurement
«FeatureType» WaterDam
«FeatureType» Terrain
«FeatureType» Relief

Figure 5.5: Overview of the NEN3610 data model. All classes in the dashed box are subclasses of GeoObject. Translated from [195].

5.3 Feature concept ontologies

A feature concept ontology formally defines the conceptualisations of real-world phenomena and the relationships between them. This section describes the construction of formal ontologies for the following conceptual models:

- NEN3610: The standardised base data model for geo-information exchange in The Netherlands [195].
- TOP10NL: The data model for the object-oriented topographic data, acquisition scale 1:10000, of the Dutch Topographic Service [17].
- Riskmap: The conceptual framework that forms the basis for the Dutch provincial risk maps.
- Travel: A small set of concepts that models a travel domain.

5.3.1 NEN3610

NEN3610 is a Dutch geo-information model based on ISO 19100 standards. The recently-finalised NEN3610 data model [195] describes a basic set of feature types, in NEN3610 called *geo-object* types, such as 'Road' and 'Planning area' in terms of their textual definition, attributes, associations with other classes and subclass/superclass relationships, and some constraints, see an overview in Figure 5.5 and a detail in Figure 5.6.

All classes are abstract (indicated by italic font), which implies that they are not designed to be directly instantiated. Instantiation is supposed to occur at

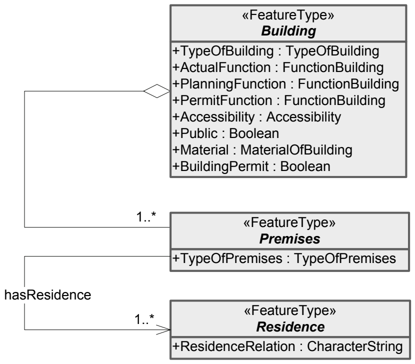

«FeatureType»
Building
+TypeOfBuilding : TypeOfBuilding
+ActualFunction : FunctionBuilding
+PlanningFunction : FunctionBuilding
+PermitFunction : FunctionBuilding
+Accessibility : Accessibility
+Public : Boolean
+Material : MaterialOfBuilding
+BuildingPermit : Boolean
1..*
«FeatureType»
Premises
+TypeOfPremises : TypeOfPremises
hasResidence
1..*
«FeatureType»
Residence
+ResidenceRelation : CharacterString

Figure 5.6: Part of the NEN3610 data model, conceptualising the *Building* class. Translated from [195] (the Dutch terms for *Building*, *Premises* and *Residence* are respectively *Gebouw*, *Pand* and *Verblijfsobject*).

more specific *sector* models (read: by more specific domain models, such as the TOP10NL data model of the Dutch Topographic Service). The NEN3610 model provides the superclasses for the non-abstract classes in the sector models. An example is given in Figure 5.7: the TOP10NL concept RoadSegment is a *concrete* class (indicated by regular font) that can be instantiated.

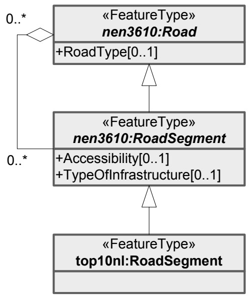


Figure 5.7: Relation between the abstract class *RoadSegment* of NEN3610 and the *RoadSegment* class of TOP10NL. Translated from [195].

All NEN3610 classes reside at the application level (the information middle level of Figure 5.1). The relationship between the application level and the meta-level is depicted in Figure 5.5 by using two-facetted constructs for *GeoObject* and *Function*, similarly to using the two-facetted constructs for *Building* and *Point* in Figure 5.1. *GeoObject* and *Function* serve as examples; such two-facetted con-

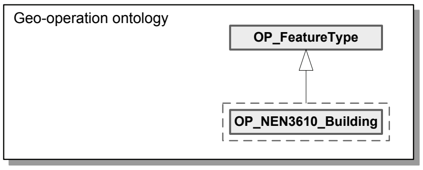


Figure 5.8: Accommodation of NEN3610 feature types in the geo-operation ontology. The dashed box includes all the NEN3610 feature types; the *OP_NEN3610_Building* serves as an example.

structs exist for all feature types and attributes in Figure 5.5. The use of NEN3610 concepts as parameter types in operations is depicted in Figure 5.8. The concept *OP_NEN3610_Building* is subsumed by *OP_FeatureType*, which serves as a feature type placeholder in the operation ontology (see Figure 5.1).

The concept ontology that represents NEN3610 in this thesis differs from its UML modelling elements with respect to the fact that specific feature types (e.g., *Road*) are not considered to be 'abstract', in contrast to the UML model. In the ontology, a NEN3610 *Road* is allowed to be instantiated, which facilitates reasoning with its instances and does not violate the essential characteristics of the model. Note however that the actual ontology reasoning does not involve geographic instances of real-world objects (see the statement about research scope in Section 1.2). In the NEN3610 ontology, feature attributes can be modelled in two different ways:

1. Associating the role *nen3610:hasAttribute* to *nen3610:GeoObject* by declaring the domain of *nen3610:hasAttribute* to be *nen3610:GeoObject* and the range to be *nen3610:BuildingAttribute*.

   The concept *nen3610:BuildingAttribute* is part of the concept hierarchy as below:

   ```
   nen3610:Attribute
       nen3610:BuildingAttribute
           nen3610:BuildingActualFunction
           nen3610:BuildingPlanningFunction
   ```

   Attribute values of *nen3610:BuildingActualFunction* are defined in an enumeration *nen3610:BuildingActualFunctionValue* with values such as *nen3610:-AccommodationalFunction* and *nen3610:OfficeFunction*. Because these attribute values are shared among the building attributes *nen3610:Building-ActualFunction* and *nen3610:BuildingPlanningFunction*, two similar enumerations are needed with values such as *nen3610:ActualOfficeFunction* and *nen3610:PlanningOfficeFunction*.

2. Modelled in a role hierarchy as follows (example of building attributes):

   ```
   nen3610:hasBuildingAttribute
       nen3610:hasBuildingActualFunction
       nen3610:hasBuildingPlanningFunction
   ```

   nen3610:hasBuildingActualFunction is conditioned by:

   $$\exists\forall\ \textit{nen3610:hasBuildingActualFunction.nen3610:BuildingFunctionValue}$$

   '∃∀' is a conjunction of ∃ and ∀. Its semantics can be found in Section 4.4. The enumeration *nen3610:BuildingFunctionValue* has subclasses such as *nen-3610:AccommodationalFunction* and *nen3610:OfficeFunction*.

Option 1 keeps the number of roles in the ontology limited through the reuse of the role *nen3610:hasAttributeType*. However, there are the following disadvantages to this way of modelling, compared to option 2:

- Cardinality restrictions can be applied only to *nen3610:hasAttributeType* and cannot be refined to *nen3610:BuildingActualFunction*. This obstructs us from defining different restrictions for different building attributes.
- In option 1, the entry of individuals does not provide the user with a selection of attributes, as is the case in option 2. In the latter, the ontology editor Protégé provides each attribute with a separate entry form, which eases the instantiation process. The Protégé software is introduced in Section 8.2.1 and its entry forms in Section 8.4.1.
- In option 2, the sharing of attributes can be implemented without further effort. In option 1 these concepts need separate trees with similar values. In this research, option 2 was followed in building the concept ontology.

5.3.2 TOP10NL

The TOP10NL feature model is an object-oriented data model that forms the conceptual basis for the TOP10NL topographic data of the Dutch Topographic Service (in Dutch: Topografische Dienst Kadaster). As the TOP10NL is earmarked as one of the models that should serve as specific sector model of the NEN3610 model, it is being revised along with the NEN3610 modelling efforts. At the same time NEN3610 draws some of its conceptualisations from the TOP10NL model. The current TOP10NL data model (version 2.3) defines its concepts in Dutch [17]. A translation of a limited set of terms was carried out in an earlier version of the data model [148], but is considered to be a side-product. For the sake of this thesis, all concepts and roles of the current version (2.3) have been translated into

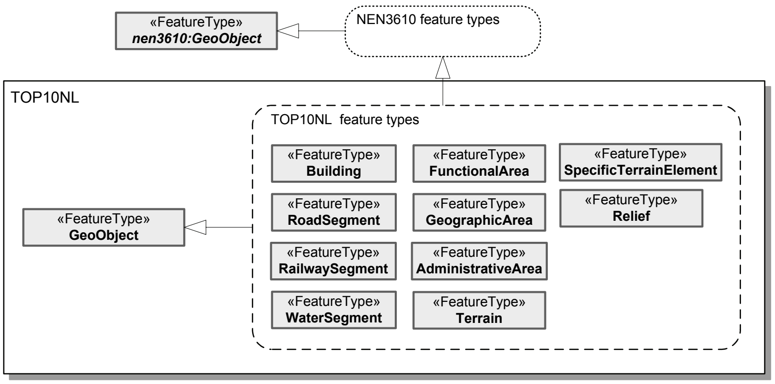


Figure 5.9: Generalised structure of the TOP10NL concept ontology.

English. In the concept ontology, the TOP10NL element structure is similar to the one of NEN3610. An overview is depicted in Figure 5.9. All TOP10NL feature types are part of TOP10NL two-facetted constructs, similarly to the two-facetted constructs for NEN3610 concepts as depicted in Figure 5.5. TOP10NL feature types also appear as ‘OP’ classes in the geo-operation ontology, similar to what is depicted in Figure 5.8.

The relationships with NEN3610 are modelled by means of mappings, declared in the TOP10NL ontology. They are described below. All TOP10NL feature types are subsumed by NEN3610 types of the same name, for example:

$$\textit{top10nl:RoadSegment} \sqsubseteq \textit{nen3610:RoadSegment} \tag{5.1}$$

Besides this overlap between the two ontologies, there are eleven NEN3610 feature types that do not have a TOP10NL counterpart. A TOP10NL class that is subsumed by a NEN3610 class, for example *top10nl:Building*, inherits the attributes of the NEN3610 class *nen3610:Building*. TOP10NL *attributes* are *not* subclasses of NEN3610 attributes. In the concept ontology the top10nl:hasBuildingAttribute is modelled at the same level as the nen3610:hasBuildingAttribute, i.e., as a sibling.

Additional mappings have been created between NEN3610 and TOP10NL in various ways. On the attribute level, direct subclassing is not sufficient and mappings have been created as follows (example of the concept *Hotel*). *Hotel* is defined in NEN3610 as:

$$\begin{aligned} &\textit{nen3610:Hotel} \equiv \\ &\quad (\exists\ \textit{nen3610:hasActualBuildingFunction.} \\ &\qquad \textit{nen3610:AccommodationalFunction}) \end{aligned} \tag{5.2}$$

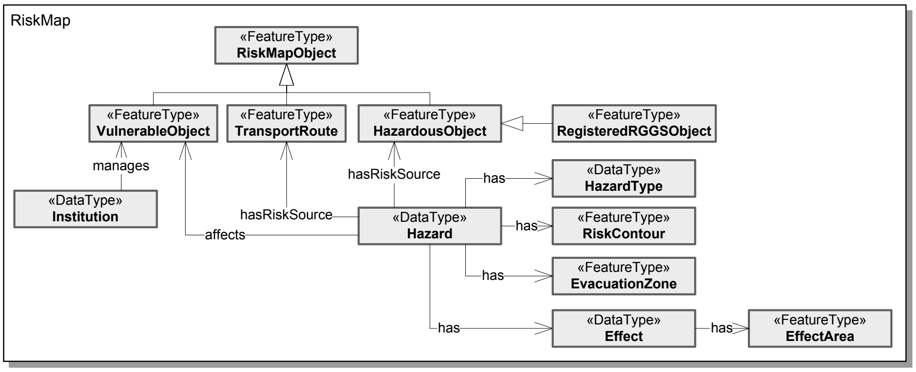

RiskMap
«FeatureType» RiskMapObject
«FeatureType» VulnerableObject
«FeatureType» TransportRoute
«FeatureType» HazardousObject
«FeatureType» RegisteredRGGSObject
manages
«DataType» Institution
hasRiskSource
hasRiskSource
affects
«DataType» Hazard
has
«DataType» HazardType
has
«FeatureType» RiskContour
has
«FeatureType» EvacuationZone
has
«DataType» Effect
has
«FeatureType» EffectArea

Figure 5.10: Generalised structure of the Riskmap concept ontology.

and in TOP10NL as:

$$\begin{aligned}&\mathit{top10nl{:}Hotel} \equiv \\ &\quad (\exists\ \mathit{top10nl{:}hasBuildingType.top10nl{:}HotelType})\end{aligned} \tag{5.3}$$

An equivalence between *nen3610:Hotel* and *top10nl:Hotel* is created as a *property value mapping* as also presented in [196] and uses in Protégé two separate 'necessary & sufficient' conditions. This mapping makes it possible to create an instance of a hotel in TOP10NL and to access it through the NEN3610 concept, and vice versa. The corresponding OWL encoding is used by a reasoner to infer the correct semantics of any instance (e.g., 'Amstel Hotel') across both data models. It has to be noted that for the sake of simplicity the above mapping example excludes potential other features with accommodational function, such as top10nl:Motel and top10nl:RecreationCentre. However, they can be easily added in the asserted condition. It is up to the knowledge engineer, who integrates the models, to make the choices to include the right concepts in those mappings [256].

5.3.3 Riskmap

In this thesis work, an ontology has been created for the domain of risk mapping, as indicated in the Riskmap use case (see Section 2.6.1). A risk survey manual [246] and a glossary [240] formed the basis for this ontology. Its structure is depicted in Figure 5.10.

All Riskmap feature types are part of Riskmap two-facetted constructs, similarly to the two-facetted constructs for NEN3610 concepts as depicted in Figure 5.5. Riskmap feature types and data types also appear as 'OP' classes in the geo-operation ontology, similar to what is depicted in Figure 5.8.

The Riskmap ontology was integrated with the other ontologies (NEN3610 and TOP10NL) through a number of mappings. These mappings serve the purpose of interoperability, i.e., the *provider* of risk map information is bound to the class definitions of NEN3610 and TOP10NL, that are mapped a priori to the Riskmap classes. This makes the intended meaning of the information more explicit to the user than without these mappings. Four example mappings with different kinds of background are given below.

Hotel A simple equivalence mapping is created between the definitions of *Hotel*:

$$\textit{riskmap:Hotel} \equiv \textit{nen3610:Hotel}$$

Nuclear The class *Nuclear* (a subclass of *Hazard*) in the Riskmap ontology represents a nuclear object. Similarly to *riskmap:Nuclear*, *top10nl:NuclearReactor* and *top10nl:NuclearPowerStation* are defined as concepts. However, *riskmap:-Nuclear* does not distinguish the power station as a subclass like TOP10NL does. The mapping is effectuated as a simple concept equivalence:

$$\textit{riskmap:Nuclear} \equiv \textit{top10nl:NuclearReactor}$$

Road Riskmap maintains a classification of roads that is a subset of TOP10NL roads, but this subset is not mapped by a 1-to-1 concept mapping. Further, the Riskmap road classes are direct subclasses of riskmap:Road and TOP10NL defines road classes as road attributes. The Riskmap road class hierarchy is as below:

```
riskmap:Road
    riskmap:NationalRoad
        riskmap:Highway
    riskmap:ProvincialRoad
```

The mappings with TOP10NL are as follows (the concept starting with *M_* is a mapping concept):

$$\textit{riskmap:Highway} \equiv \exists\ \textit{symbol:hasThematicAttributeType.top10nl:Highway}$$

$$\textit{riskmap:ProvincialRoad} \equiv$$
$$\exists\ \textit{symbol:hasThematicAttributeType.top10nl:RegionalRoad}$$

$$\textit{M\_Top10-Riskmap-MainRoad} \sqsubseteq \textit{riskmap:NationalRoad}$$

$$\textit{M\_Top10-Riskmap-MainRoad} \equiv$$
$$\exists\ \textit{symbol:hasThematicAttributeType.top10nl:MainRoad}$$

Note that the TOP10NL road concept is also mapped to the NEN3610 road concept (see Axiom 5.1).

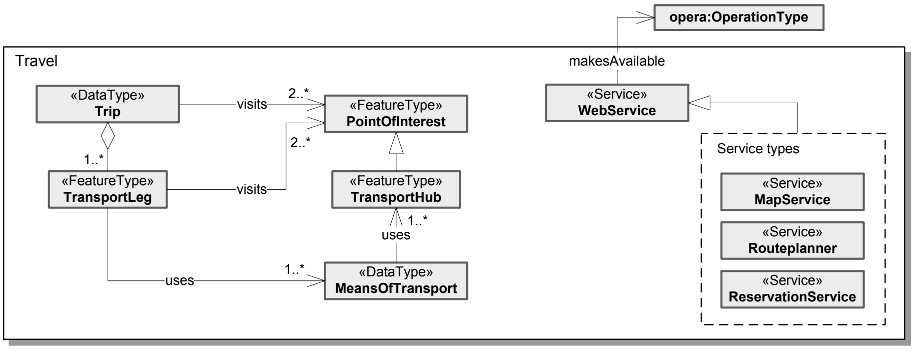


Figure 5.11: Generalised structure of the Travel concept ontology.

Tunnel The concept *Tunnel* in the Riskmap ontology is defined as 'a location where train, tram, or car traffic passes under a full-covering construction for at least 250 metres'. Each tunnel has a category that puts restrictions on the type of traffic that makes use of it. The *riskmap:Tunnel* class is a subclass of the nen3610 tunnel, not its equivalent. TOP10NL does not know the class *Tunnel* as object but describes infrastructural elements in tunnels. *Riskmap:Tunnel* is defined as follows:

$$riskmap{:}Tunnel \sqsubseteq nen3610{:}Tunnel$$

$$\begin{aligned}
&riskmap{:}Tunnel \sqsubseteq \\
&\quad \exists\ symbol{:}hasThematicAttributeType.(\\
&\qquad top10nl{:}RailroadSegmentInTunnel \sqcup \\
&\qquad top10nl{:}RailroadSegmentOnFixedPartOfTunnel \sqcup \\
&\qquad top10nl{:}RoadSegmentInTunnel \sqcup \\
&\qquad top10nl{:}TerrainInTunnel\ )
\end{aligned}$$

5.3.4 Travel

A simple travel ontology has been created as part of the thesis work to demonstrate the discovery of travel information, such as local transport means and points of interest. This information serves the use case discussed in Section 2.6.4. An overview of the important concepts is given in Figure 5.11.

All Travel feature types are part of Travel two-facetted constructs, similar to the two-facetted constructs for NEN3610 concepts as depicted in Figure 5.5. Travel feature types and data types also appear as 'OP' classes in the geo-operation ontology, similar to what is depicted in Figure 5.8. In addition, Figure 5.11 shows three examples of web services which operation descriptions make use of the Travel 'OP' classes. For example, the description of a route planner operation may make use of the concept *OP_PointOfInterest* to describe its parameters for the start and destination of a route. Section 7.4 contains an example of the use of such a description for service discovery.

5.4 Geo-operation characterisations — OPERA

This section describes the starting points for the design of an ontology of geo-operations, called *OPERA*. From a design perspective, we would like such an ontology to (1) have a hierarchical structure (for ease of human understanding), (2) to have non-overlapping classes as much as possible, (3) to include the most important geo-operations and (4) to be extensible. In the past, several attempts have been made to create a classification of geo-operations. The most relevant ones are stemming from research by Albrecht [6, 7], Chrisman [49, 50], from standardisation efforts, such as [132] and from software models [92, 111]. Most of them are, however, informal and lack sufficient semantics for classifying individuals. A major problem was caused by the lack of sufficient mechanisms to support multiple inheritance. With modern ontology languages such as OWL, we can now overcome this problem.

OPERA is based on the principles of OWL-S and particularly implements its process model. However, specific to OPERA is its classification of geo-operations and its descriptions of specific geo-operation parameter types, both of which are not part of the OWL-S ontology. The main strategy in classifying geo-operations is to consider the elements of the feature concept ontology and the feature symbol ontology to be the basic representatives for the input and output parameter types of the operation classes.

5.4.1 Atomic geo-operations

The design of a geo-operation ontology in this thesis is based on a distinction between atomic geo-operations and composites of them. The definition of an atomic geo-operation, as provided below, follows the definition of an atomic process in OWL-S (see Section 4.7.1). In OWL-S, an atomic process is defined as a directly invocable process that executes in a single step. It takes an input message and returns an output message. These messages can take as many formal input c.q. output parameters as required.

An atomic geo-operation may *include* functionality also performed by another atomic operation. This does not make it a composite. For example, consider an overlay operation that is performed on a feature collection of which the topological relationships between its features are not made explicit. As part of the operation it must first calculate the intersections between the features and then combine the thematic attributes. This (sub)operation could be performed by the atomic operation 'MakeTopologyRelationshipsExplicit'. The overlay operation is still considered to be atomic. Further, an *atomic geo-service* is considered to be a service that makes available an atomic geo-operation and is not composed of other services.

A distinction has to be made between operation implementations and operation descriptions. An operation implementation is the invocable software artefact (e.g., a software component) that carries out the operation. A description is considered to be represented by (1) an allocation of the service to an operation class in the geo-operation ontology or (2) a workflow with its sub-operations, each of which is allocated to an operation class in the geo-operation ontology.

At first glance, one would choose to describe an atomic operation implementation with option (1) and a composite operation implementation with option (2). However, cross options are also possible, which are explained below.

Case 1 The operation is considered to execute in one step (see definition atomic geo-operation). It may have internal workflow constructs, but they are unknown. The description contains only formal input and output parameters.

Case 2 The operation is considered to execute in one step (see definition atomic geo-operation). However, the description may detail out internal workflow. This workflow may even be a virtual one, which is not the exact workflow implemented, but which has a similar effect. The description contains formal input, output parameters and workflow constructs.

Case 3 The operation is a composite one, exposing its workflow constructs during execution. The description treats the operation still as atomic, it contains only formal input and output parameters.

Case 4 The operation is a composite one, exposing its workflow constructs during execution. The description contains formal input and output parameters and workflow constructs. This workflow may be a virtual one, which is not the exact workflow implemented, but which has a similar effect.

5.4.2 Ontology design — OPERA-R and OPERA-D

Our geo-operation ontology (OPERA) is developed with two sets of concepts:

- A *reference geo-operation ontology* (OPERA-R), containing atomic geo-operation types that act as building blocks for all other geo-operation types.
- *Derived geo-operation ontologies* (OPERA-D), which may contain
  - atomic geo-operation types, each of which is defined in terms of an operation type, existing in the reference ontology (OPERA-R).

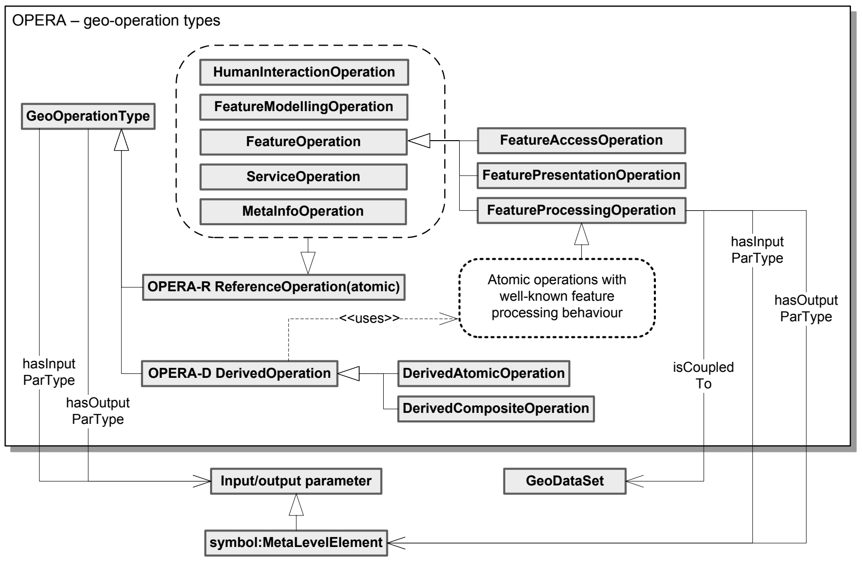

OPERA – geo-operation types
GeoOperationType
HumanInteractionOperation
FeatureModellingOperation
FeatureOperation
ServiceOperation
MetaInfoOperation
FeatureAccessOperation
FeaturePresentationOperation
FeatureProcessingOperation
OPERA-R ReferenceOperation(atomic)
Atomic operations with well-known feature processing behaviour
<<uses>>
OPERA-D DerivedOperation
DerivedAtomicOperation
DerivedCompositeOperation
hasInput ParType
hasOutput ParType
isCoupled To
Input/output parameter
GeoDataSet
symbol:MetaLevelElement

Figure 5.12: Generalised structure of the OPERA geo-operation ontology.

– composite geo-operation types, each of which is defined in terms of a workflow and its component operation types, being geo-operation reference types or derived types.

An overview of the main structure of OPERA is given in Figure 5.12. At the root is the class *GeoOperationType*. It has five direct subclasses that involve respectively human interaction operations, feature modelling operations, feature operations, operations on services and meta-information operations. This top level classification and sub classification follows the classification as given for services in Table 2.3 and details out the structure given in Figure 5.1. OPERA-R describes atomic geo-operation types. Each type can be instantiated by an atomic geo-operation instance as depicted in Figure 5.1. A more detailed description of the informal semantics of the classes of OPERA-R is provided in Appendix C.

The class *FeatureProcessingOperation* has been selected to be specified in more detail as it forms a broad basis for geo-information processing and analysis activities in practice. In contrast to feature *modelling* operations, feature processing operations are not meant to add feature types to an application schema, although they may create feature types and feature properties as placeholders for feature instances and their attributes. For example, an operation of type *Buffer* may have an output parameter, named *BufferZone* of type *GF_FeatureType* with *GF_SpatialAttributeType* as a property type. An instance of this operation (e.g., *MyBufferOperation*) has an output parameter of type OP_FeatureType with an attribute of type OP_Polygon. At the operation invocation level, the output of this operation may be an actual buffer instance, represented by an actual polygon with coordinates.

The input and output parameter types of a feature processing operation are expressed in terms of elements of the feature symbol ontology, described in Section 5.2. A *FeatureProcessingOperation* may be coupled to a data set. This may constrain the validity of usage of that operation. For example, a specific route planner operation may operate within the boundaries of a national data set. Feature processing operations are discussed in more detail in Section 5.5.

Examples of OPERA-D instances are elaborated upon in Chapter 6.

5.5 OPERA-R — Feature processing operations

This section describes types of feature processing operations as part of the semantic interoperability framework. The feature processing operations in this framework apply the OWL-S process model. This implies the distinction between atomic and composite operations. In OWL-S, an atomic feature processing operation does not maintain state, but a composite operation does. The notion of atomicity is clarified in Section 5.5.1.

5.5.1 Atomicity

In this thesis, an atomic feature processing operation is a specific case of an atomic geo-operation (see the subsumption relationship in Figure 5.12). It is defined as follows (see Section 5.1 and Figure 5.1 for a description of terms such as ‘application level’ and ‘feature type’). A characterisation of an atomic operation is depicted in Figure 5.13.

Term 5.4 *A feature processing operation is atomic if, and only if, it has all of the following characteristics:*

- *It must create, modify or extract one or a combination of the following:*
  - *feature type (at the information application level),*
  - *feature instance (at the data level),*
  - *feature property type(at the information application level),*
  - *feature property value (at the data level).*
- *It may use and/or produce non-feature values (values not associated with features, e.g., the average car speed, needed as input for a fastest route calculation).*
- *It must be performed as a single step; the operation does not expose a workflow.*

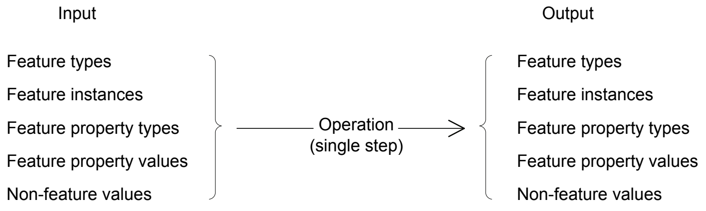


Figure 5.13: Atomic geo-operations are performed in a single step. Input and output parameters are defined in the feature symbol ontology.

- *The input and output parameters must allow specification in terms of (1) classes that exist on the meta-level of the feature symbol ontology (SYMBOL) and (2) OP-class copies of the application level of SYMBOL.*

Note that Definition 5.4 includes feature associations, due to the fact that in the ISO General Feature Model *GF_FeatureType* subsumes *GF_AssociationType* (see Figure 5.2).

A *composite* feature processing operation is defined as follows:

Term 5.5 *A* composite *feature processing operation is composed of an atomic feature processing operation or another composite feature processing operation.*

Atomic feature processing operations and composite feature processing operations borrow the semantics from, respectively, OWL-S atomic and composite processes (described in Section 4.7.1).

5.5.2 Classification of operations

An overview of feature processing operations is provided in Figure 5.14, their informal semantics are described in Appendix C.

Each operation class is given a short name for easy reference, such as 'Change-CRS' (a class of operations that change the coordinate reference system of a coordinate set). The operations are grouped in sections. This classification is principally based on the kind of feature properties that they act upon or return. The classification of GIS operations by Chrisman [50] follows for the major part the same principle and is used in particular in this thesis for the description of attribute operations, overlay and distance-based operations. Other parts of that classification are not directly followed, such as his concept of *transformation*. Chrisman defines a transformation as an operation that changes a measurement framework (a scheme with measurement rules). This mechanism is rather complex and seems to lead to an ambiguous classification of operations. The concept of transformation that is used in this thesis classifies geometric transformations as operations

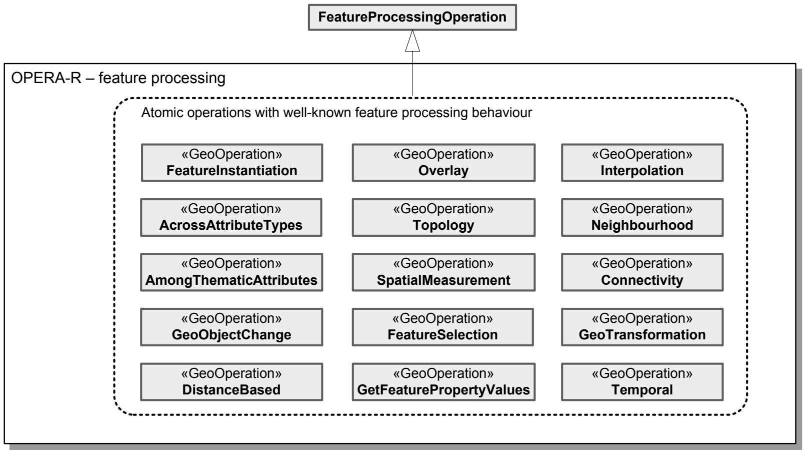

FeatureProcessingOperation
OPERA-R – feature processing
Atomic operations with well-known feature processing behaviour
«GeoOperation» FeatureInstantiation
«GeoOperation» Overlay
«GeoOperation» Interpolation
«GeoOperation» AcrossAttributeTypes
«GeoOperation» Topology
«GeoOperation» Neighbourhood
«GeoOperation» AmongThematicAttributes
«GeoOperation» SpatialMeasurement
«GeoOperation» Connectivity
«GeoOperation» GeoObjectChange
«GeoOperation» FeatureSelection
«GeoOperation» GeoTransformation
«GeoOperation» DistanceBased
«GeoOperation» GetFeaturePropertyValues
«GeoOperation» Temporal

Figure 5.14: Feature processing operation classes in the OPERA-R ontology.

that change the coordinates of positions (see Section C.4.14 for a subclassification of the *geometric transformation* operation type).

Other resources used, such as the ISO 19100 standards are mentioned throughout Appendix C.

In some cases, an operation type may be classified under another section as well. For example, the *GridSlope* operation type is classified under both the category *GridFilter* and *CalculateSlope* (both subclasses of the *Neighbourhood* operation type), which implies that the latter two categories are not mutually exclusive.

5.5.3 Operation type descriptions

Each operation type as depicted in Figure 5.14 is characterised by its functional description in Appendix C, its input and output parameters and its tightly-coupled data. Pre- and postconditions may be represented in these parameters.

Descriptions of the specific input/output parameters attached to each reference operation are provided in Appendix D. The input parameters attached to an operation type by the *hasInputPar* are the formal parameters minimally needed for that operation type. The output parameters attached to an operation type by the *hasOutputPar* are the formal parameters that are minimally needed to contain the output of the operation type. Operation instances may make use of other parameters, but they are irrelevant for describing the operation type. With respect to parameter passing, a single formal parameter may be used to pass multiple values (e.g., instances of geometric objects) in an array. In this respect, the operation descriptions in Appendix D are generic in the sense that they do not specify cardinality constraints for parameter values. This is left to more specific descriptions that are potentially created to describe subclasses of generic operation types. For example, a more specific description of a route optimisation operation may specify the number of given points in a route. Further, our input/output model assumes a call-by-value parameter evaluation; the parameter descriptions in Appendix D assume feature modifying operations (e.g., the *SimplifyGeo* class of operations) to modify its local copy of the feature.

The input/output model assumed here, follows the one of OWL-S, which supports stateless services (with input and output parameters) as well as stateful services (with, in addition, pre- and/or post-conditions). A feature processing operation may require a data set to comply with a precondition, i.e., to be in a specific state. This may involve the features being processed, but also any other feature in the data set. For example, the entire data set must be defined in a specific coordinate system. A postcondition may be of a similar kind. Other operations than classified in Appendix C can be defined, based on the reference classes and they are called 'derived operations', which may be atomic or composites. Composite operations are described in Chapter 6. In OPERA, all geo-operation types are subsumed by opera:GeoOperationType. The class hierarchy as indicated in Figure 5.12 is translated into ontology classes. The ontology also contains so-called *support concepts*. They define the general characteristics of operations such as operation specific details, like a property named *topological selection method* that is (only) attached to the *overlay* operation.

Disjointness Two operation types are considered to be disjoint if, based on their functionality, an individual operation can only be classified as one of the types (and not both). Disjointness has been created between some, but not all of the operation types in OPERA-R. This choice has been made for each individual class, based on its semantics, as described in Appendix C. For example, the *LocSpat* operation type (representing the class of operations that have as input parameter type *GF_LocationAttributeType* and as output *GF_SpatialAttributeType*) is disjoint from the *SpatLoc* operation type (input type: *GF_SpatialAttributeType* and output type: *GF_LocationAttributeType*), but *GridFilter* and *CalculateSlope* are not disjoint.

Primitive vs. defined classes The operations in OPERA-R have been implemented as primitive classes and not as defined classes (see the distinction made in Section 4.2.1, paragraph 'Tbox') due to the following. Some geo-operations types have identical input and output parameter types, but have different functionality. We would like such types to be disjoint in the ontology. If these operation types are described by their parameters only, defined classes cannot be disjoint, but primitive classes can.

Example 1

An example of an operation definition that is used for instantiation and reasoning is given below for the *LocSpat* operation. The name '*LocSpat*' represents an operation type that reads a location attribute type (e.g., instantiated as an address type) and produces a spatial attribute type (e.g., instantiated as a geometric object type), which is typically found in a gazetteer:

$$\begin{aligned}
&\textit{opera:LocSpat} \sqsubseteq \\
&\quad \textit{opera:AcrossAttributeTypes} \sqcap \\
&\quad (\exists\forall\ \textit{opera:appliesToDataStrucType}. \\
&\qquad (\textit{symbol:ObjectFeature} \sqcup \textit{symbol:GridCell})) \sqcap \\
&\quad (\exists\forall\ \textit{opera:hasInputPar}.(\exists\ \textit{opera:hasParType}. \\
&\qquad \textit{symbol:GF\_LocationAttributeType})) \sqcap \\
&\quad (\exists\forall\ \textit{opera:hasOutputPar}.(\exists\ \textit{opera:hasParType}. \\
&\qquad \textit{symbol:GF\_SpatialAttributeType})) \sqcap \\
&\quad (\geq 1\ \textit{opera:isCoupledToDataset})
\end{aligned} \tag{5.4}$$

The above operation description is a translation of the LocSpat entry in Appendix D into Description Logic. Note the notational occurrence of pairs of existential quantifications (∃...) and value restrictions (∀...) as a combination (∃∀...) for each role as introduced in Section 4.4. In the above example, the *opera:hasInputPar* is conditioned by an existential quantification to state that the LocSpat operation needs at least one parameter of type *symbol:GF_LocationAttributeType.* The value restriction is used to make sure that any input parameter is of type *symbol:GF_LocationAttributeType* and nothing else. The cardinality restriction of the role opera:isCoupledToDataset indicates that the operation is coupled to at least one data set.

Example 2

The use of a support concept (which notion was introduced in 5.5.3) is illustrated in the definition of an overlay operation below. Shown is the generalised definition of a geometric intersection operation with the support concept *opera:Attribute-*

CombinationMethod:

$$\begin{aligned}
&\textit{opera:GeometricIntersection} \sqsubseteq \\
&\quad \textit{opera:ObjectOverlay} \sqcap \\
&\quad (\exists\forall\ \textit{opera:appliesToDataStrucType.symbol:ObjectFeature}) \sqcap \\
&\quad (\exists\forall\ \textit{opera:hasInputPar.}(\exists\ \textit{opera:hasParType.} \\
&\qquad ((\exists\ \textit{opera:typeBijection.symbol:GeometricObject}) \sqcup \\
&\qquad \textit{symbol:GF\_ThematicAttributeType}))) \sqcap \\
&\quad (\exists\forall\ \textit{opera:hasOutputPar.}(\exists\ \textit{opera:hasParType.} \\
&\qquad ((\exists\ \textit{opera:typeBijection.symbol:GeometricObject}) \sqcup \\
&\qquad \textit{symbol:GF\_ThematicAttributeType}))) \sqcap \\
&\quad (\exists\forall \textit{opera:hasAttributeCombinationMethod.} \\
&\qquad \textit{opera:AttributeCombinationMethod}) \sqcap \\
&\quad (=0\ \textit{opera:isCoupledToDataset})
\end{aligned} \tag{5.5}$$

The above description states that the geometric intersection operation needs as input the combination of geometric objects AND thematic attribute types (the cardinality is not defined here) and produces a similar combination as output. Further, it states that it uses an attribute combination method (the method that combines thematic attributes of the intersected features and assigns them to the newly formed features, see Section C.4.9), but it does not specify which one. In a more specific definition, this can be accommodated by adding the following role restriction in Description 5.5:

$$\begin{aligned}
&(\exists\forall\ \textit{opera:hasAttributeCombinationMethod.} \\
&\quad (\textit{opera:EnumerationRule} \sqcup \textit{opera:DominanceRule}))
\end{aligned} \tag{5.6}$$

The above definition represents a type that can be instantiated by a geometric intersection operation like ESRI's ArcGIS Intersection tool.

5.6 Geo-service descriptions

Service descriptions represent the syntactic, structural and semantic properties of a service's operations, workflow, input, output and tightly-coupled data. A machine-processable service description based on the semantic framework described above extracts the semantic properties (concept identifiers) from the ontologies, relevant to the service at hand. If, in addition to service discovery, service execution is relevant, then the semantic properties are linked to the syntactic parameters, necessary for invoking the service. There are two options for integrating syntactic, structural and semantic properties, i.e., (1) annotating ontology documents with parameters representing syntax and structure and (2) annotating syntax/structure-providing service descriptions with ontology concepts. They are discussed in Section 5.6.2 after an overview of general description methods with OWL in Section 5.6.1.

5.6.1 Semantic descriptions based on OWL

Descriptions using classes

In our semantic framework, the relation between a service and its operations is as follows. The *serv* prefix represents 'service', local to OPERA, and is distinct from the *service* prefix used in OWL-S:

$$\textit{serv:GeoService} \sqsubseteq \exists\forall\ \textit{serv:makesAvailable.opera:GeoOperation} \tag{5.7}$$

To clarify the description of an existing service, the Alexandria Digital Library (ADL) Gazetteer service [115] (introduced in Section 1.1) is taken as example for the following description:

$$\begin{aligned} &\textit{serv:ADLGazetteerService} \sqsubseteq \\ &\quad (\textit{serv:GeoService}) \sqcap \\ &\quad (\exists\forall\ \textit{serv:makesAvailable.opera:ADLGazetteer}) \end{aligned} \tag{5.8}$$

The above description states that the (only) operation type of *serv:ADLGazetteer-Service* is *opera:ADLGazetteer*, which is a subclass of *opera:LocSpat* (described in Section 5.5.3, example 1, Description 5.4). The description provided for *opera:-LocSpat* can be refined for *opera:ADLGazetteer* as follows:

$$\begin{aligned} &\textit{opera:ADLGazetteer} \sqsubseteq \\ &\quad \textit{opera:LocSpat} \sqcap \\ &\quad (\exists\forall\ \textit{opera:appliesToDataStrucType.symbol:ObjectFeature}) \sqcap \\ &\quad (\exists\forall\ \textit{opera:hasInputPar.}(\exists\ \textit{opera:hasParType.} \\ &\qquad (\exists\ \textit{opera:typeBijection.opera:OP\_Address}))) \sqcap \\ &\quad (\exists\forall\ \textit{opera:hasOutputPar.}(\exists\ \textit{opera:hasParType.} \\ &\qquad (\exists\ \textit{opera:typeBijection.opera:OP\_Point}))) \sqcap \\ &\quad (=1\ \textit{opera:isCoupledToDataset}) \sqcap \\ &\quad (\exists\ \textit{opera:isCoupledToDataset.ADLWorldDataset}) \end{aligned} \tag{5.9}$$

This description states that any ADLGazetteer operation takes as input *opera:-OP_Address* as input and produces as output *opera:OP_Point*. It deals with object features, not with grid coverages. For its functioning, it relies on a data set, called *ADLWorldDataset*. Further refinement of the definition can be established by including other role restrictions, such as the definition of the coordinate reference system to which the output of the service is restricted:

Refinement 1

A refinement of the description of *opera:ADLGazetteer* is made when the role restriction below is included in Description 5.9.

$$\begin{aligned} &(\exists\forall\ \textit{opera:hasOutputPar.}(\exists\ \textit{opera:hasParType.} \\ &\quad (\exists\ \textit{opera:typeBijection.} \\ &\quad (\exists\ \textit{opera:hasReference.opera:OP\_SC\_CRS\_LatLon})))) \end{aligned} \tag{5.10}$$

Another refinement specifies that *opera:ADLGazetteer* accepts a city name (as partial address) as input:

Refinement 2

In this refinement the role restriction of Description 5.9

$$\begin{aligned}(\exists\forall\ &\textit{opera:hasInputPar}.(\exists\ \textit{opera:hasParType}.\\ &(\exists\ \textit{opera:typeBijection}.\textit{opera:OP\_Address})))\end{aligned} \tag{5.11}$$

is replaced with

$$\begin{aligned}(\exists\ &\textit{opera:hasInputPar}.(\exists\ \textit{opera:hasParType}.\\ &(\exists\ \textit{opera:typeBijection}.\textit{opera:OP\_CityNameAddress})))\end{aligned} \tag{5.12}$$

The omitted 'forall' quantifier means that *symbol:CityNameAddress* always has to be provided, but that the operations can also take additional input types (however, they are all *symbol:GF_LocationAttributeType*, as specified in Description 5.4).

Descriptions using individuals

The gazetteer operation, made available by the ADL Gazetteer service can also be described using individuals, in two ways:

1. As instance of the root class *opera:LocSpat* as defined in Description 5.4. The characterisation of its parameters are established by role assertions, for example for its input:

   $$\textit{opera:hasInputPar(ADLGazOperation,ADLGazInputPar\_1)} \tag{5.13}$$

   where the role *opera:hasInputPar* is attached to the individual *_ADLGaz-Operation* and has as filler the individual *ADLGazInputPar_1*. This and other role assertions and instantiating concepts of *_ADLGazOperation* are depicted in Figure 5.15. The diagram shows the *_ADLGazOperation* as instance of the *LocSpat* operation with the characterisation its input and output parameters. Concept-instance relations are only displayed for the input parameters (similar concept-instance relations exist for output parameters, but they omitted to avoid congestion in the figure). An OWL representation of this service description can be found in Appendix E. The description contains an import statement that imports all the elements (concepts, roles and individuals) of the OnToGeo ontology. Because of this import, the service description itself does not contain these elements and can therefore be relatively short. In addition to the import statement it contains the import namespaces and the individuals that instantiate the import concepts.

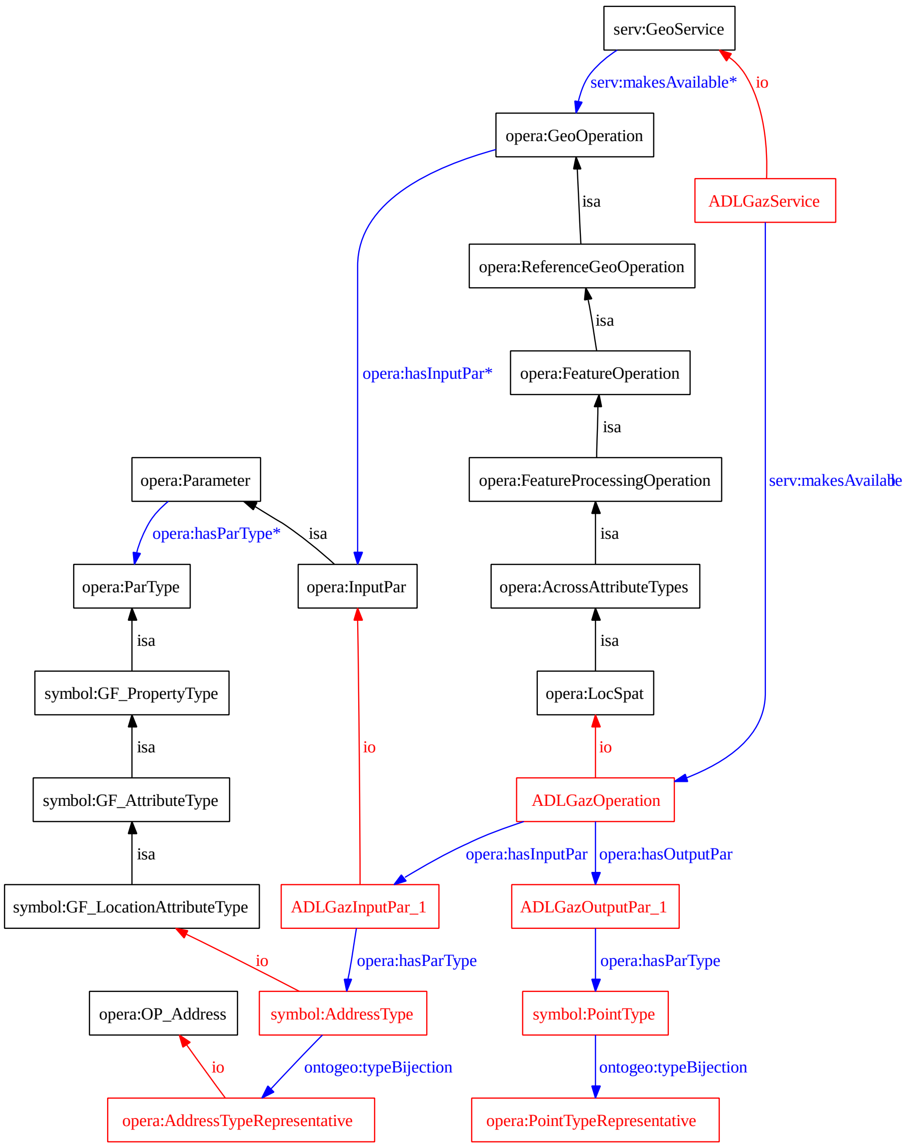


Figure 5.15: Ontology capture diagram showing the Alexandria Digital Library Gazetteer service (*ADLGazService*) as instance of the *serv:GeoService* concept.

2. As instance of the class opera:ADLGazetteer as defined in Description 5.9 (with OWL representation in Appendix E). The characterisation of its parameters is done by role assertions, similar to option 1. However the allowed instances are restricted to a more confined set of individuals, due to the more specific concept conditions. In contrast with option 1, this option obviously requires for each operation description a specific concept definition.

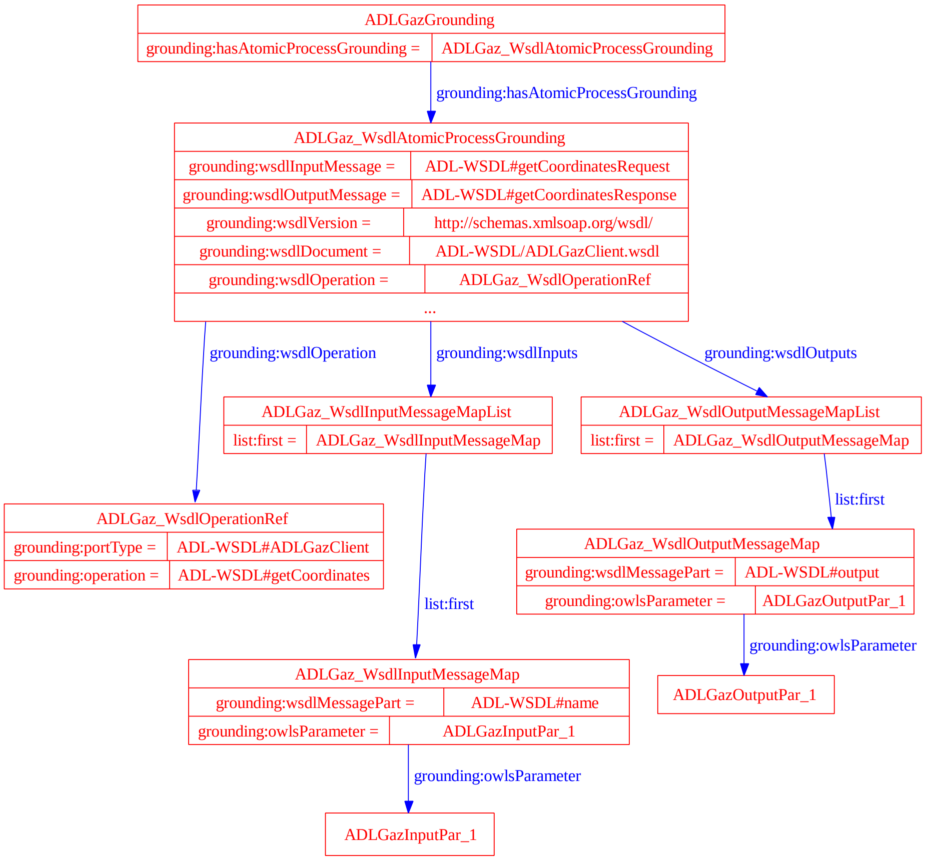

ADLGazGrounding
grounding:hasAtomicProcessGrounding = ADLGaz_WsdlAtomicProcessGrounding
grounding:hasAtomicProcessGrounding
ADLGaz_WsdlAtomicProcessGrounding
grounding:wsdlInputMessage = ADL-WSDL#getCoordinatesRequest
grounding:wsdlOutputMessage = ADL-WSDL#getCoordinatesResponse
grounding:wsdlVersion = http://schemas.xmlsoap.org/wsdl/
grounding:wsdlDocument = ADL-WSDL/ADLGazClient.wsdl
grounding:wsdlOperation = ADLGaz_WsdlOperationRef
...
grounding:wsdlOperation
grounding:wsdlInputs
grounding:wsdlOutputs
ADLGaz_WsdlInputMessageMapList
list:first = ADLGaz_WsdlInputMessageMap
ADLGaz_WsdlOutputMessageMapList
list:first = ADLGaz_WsdlOutputMessageMap
list:first
ADLGaz_WsdlOperationRef
grounding:portType = ADL-WSDL#ADLGazClient
grounding:operation = ADL-WSDL#getCoordinates
ADLGaz_WsdlOutputMessageMap
grounding:wsdlMessagePart = ADL-WSDL#output
grounding:owlsParameter = ADLGazOutputPar_1
list:first
grounding:owlsParameter
ADLGazOutputPar_1
ADLGaz_WsdlInputMessageMap
grounding:wsdlMessagePart = ADL-WSDL#name
grounding:owlsParameter = ADLGazInputPar_1
grounding:owlsParameter
ADLGazInputPar_1

Figure 5.16: Ontology capture diagram showing the OWL-S grounding of the Alexandria Digital Library Gazetteer service. Note that, in contrast to other ontology capture diagrams used in this thesis, this diagram shows expanded class boxes with their attributes.

5.6.2 Creating hybrid descriptions: grounding

A semantic description of service concepts must be complemented with a syntax-/structure-providing description of service access protocols once service execution is needed. The following paragraphs elaborate upon alternative approaches for creating *hybrid* descriptions which support syntactical, structural *and* semantic interoperability. They will be applied to the ADL Gazetteer service example (introduced in Section 1.1 and also used in Section 5.6.1).

OWL-S grounding

In OWL-S, the annotation of concepts with syntax/structure-based parameters is done in the OWL-S grounding, as discussed in Section 4.7.1. The ADL Gazetteer service grounding is performed with its WSDL file (see Appendix F). The resulting OWL-S grounding is depicted in Figure 5.16. For display reasons, the original URI prefix of the WSDL file 'http://localhost:8080/axis/services/ADLGazClient' is represented in the figure by the shorter prefix 'ADL-WSDL'.

Such grounding makes it possible to integrally perform service parameter inference and invocation.

Annotation with WSDL-S

Annotating syntax/structure-providing service descriptions, such as WSDL files, with ontology concepts is done according to the WSDL-S approach (see Section 4.7.2). The following namespaces have been added to header of the WSDL file:

```
xmlns:wssem="http://www.ibm.com/xmlns/WebServices/WSSemantics"
xmlns:Ontology0="http://geoserver.itc.nl/lemmens/owl/ontogeo.owl"
```

The message part body of the WSDL file has been annotated with WSDL-S constructs (with 'wssem' prefix):

```
<wsdl:message name="getCoordinatesRequest">
    <wsdl:part name="name" element="xsd1:RequestType"
    wssem:modelReference="Ontology0#OP_Address"/>
</wsdl:message>

<wsdl:message name="getCoordinatesResponse">
    <wsdl:part name="output" element="xsd1:ResponseType"
    wssem:modelReference="Ontology0#OP_Point"/>
</wsdl:message>
```

The result is comparable with the aforementioned OWL-S grounding approach. However, the annotation involves less effort and it fits more invocation approaches, because currently, WDSL file handling is more common practice (e.g., in WS-BPEL) than the handling of OWL-S documents.

Especially the construct *wssem:modelReference* is generally applicable (it associates an XML schema element directly with an ontology concept) and is thought to be useful for the semantic annotation of information sources in general. Other annotation approaches for information sources have been proposed in [146, 227, 247, 277]. This thesis proposes to use *wssem:modelReference* as a basic construct, due to its well defined semantics, compared to the others. An example of its use in the context of this thesis is provided in the use case described in Section7.2.1.

5.7 Summary and reflection

This chapter has provided the basic elements of a semantic interoperability framework for geo-services, based on the principles of semantic interoperability frameworks, discussed in Section 4.6.

An important starting point has been the well-established conceptual model for geographic features of ISO. The generic constructs of this model are implemented in this research as concepts in ontologies. By doing so, the ISO model can be used for asserting and inferring relationships between specific geographic domain models. The ISO model forms a solid basis for concept formalisation. However, some issues complicate the realisation of machine ontologies:

- **Integrated data structure representation.** The ISO General Feature Model (ISO 19109) and Coverage Model (ISO 19123) are quite detached, and although in the same models, coverages are seen as features, it is difficult to integrate them for operation typification.

- **Metaclass implementation.** The paradigm of metaclasses would demand for an ontology implementation where class instantiations at the meta-level can be treated as classes in the level below (application level). In practice, this can be achieved only with OWL-Full. Because OWL-Full is undecidable, this does not give us the possibilities of reasoning that we strive for. The basic ISO model has been implemented in OWL-DL, but not without a careful treatment of the so-called 'meta-bridges'. These meta-bridges appear at two places (see Figure 5.1):

    1. Between the meta-level and the application level of the information model. Traversing these meta-bridges relies on the use of a naming convention, i.e., the use of similar names for class and corresponding meta-individual. No naming convention-based mechanism can be built into the ontology itself; it has to be deployed by an external algorithm. The latter has not yet been implemented in the prototype design of this research.

    2. Between the meta-level of the information model and the operation type level of the operation model. Intuitively, we might draw direct role relationships between parameter types in the operation model and classes

> in the information model, for example the role *hasParType* between *Parameter* and *Point*. However, the problem that we face here is that an instance of *Point* is an actual point with specific coordinates (representing a real-world object). As a workaround, this research has opted for an approach that uses operation parameter classes that are 'clone' classes of the information model. These clone classes apply a naming convention by adding the prefix 'OP_' to the names of the information model classes. Because there exist only one-to-one relationships between the information model classes and OP_ classes, a type bijection has been implemented in the operation ontology. This makes the traversing of the meta-bridge possible from within the ontology and thus it can be used for reasoning.

The ontologies have been developed according to the method proposed in Section 4.3.

Ontology mappings Examples of mappings between geo-information ontologies have been provided. The presented ontologies have a relatively simple structure and manual mapping by a domain expert seems to be feasible. Moreover, the NEN3610 and TOP10NL ontologies have been developed in cooperation, so that some conceptual relationships are already apparent in their design. However, with the growing demand for the (sometimes ad hoc) integration of geo-information within other disciplines, it is not unlikely that mappings are also needed with ontologies having a totally different and more complex structure. Semi-automatic mapping methods as discussed in Section 4.6.1 will then be needed.

Operations and OWL-S In addition to utilising feature concepts in data models, they are also used to specify the input and output parameters of geo-operation types. By deploying the process model of OWL-S, we are able to create and reason with combinations of operation type descriptions. Although we have applied the basic principles of OWL-S, our model provides additional functionality. First, it allows to associate types of functionality to operations within a service (in OWL-S this is only possible at the service level, i.e., in the *service profile*). Second, it adds two simple constructs for service composition, which will be discussed in the next chapter.

Concepts versus individuals The discussion on concept vs. individual modelling is both a fundamental and a practical one. Concept modelling allows us to define subsumption relationships, disjointness, etc. and will be always at the basis of describing sets of individuals. An essential choice has to be made whether to define the leafs of the ontological tree as concepts or individuals. Rector [236] provides a rule of thumb as follows: *Can it have kinds - if so, it is a class* . In a practical sense, concepts have the advantage of being able to comprise accurate conditions, referring to other concept definitions. In addition, concepts have to be used in case of mappings with other ontologies. On the other hand, individuals provide an easier service description entry by a service provider or service requester and they allow us to fully deploy query languages such as OWL-QL. Obviously, the description and ontology design in terms of concept vs. individual modelling is also directly linked with the reasoning method. The building of the OnToGeo ontology concentrates on the description of data sets and services. To allow for different modes of reasoning, the semantic interoperability framework (according to the definition in Section 4.6, without the data/service descriptions) has been modelled entirely with classes. This is to be able to accommodate all data/service descriptions as instances. This is a requirement of three out of four modes of matchmaking as elaborated in Chapter 6.

Chapter 6

Geo-information matching and service chaining

Geo-service chaining draws much on the service chaining paradigms that have been developed in ICT mainstream. As stated in Section 3.2.3, specific for the GIS domain are the special kind of services, called geo-services, that are implicitly connected by the geographic location of the features they contain. Geo-services support the complex nature of geo-information and enable the user to derive new geo-information and reuse it with the same or other services for further processing. In addition, because of the reuse, integrated spatial and thematic attribute analysis methods are part of the provenance of geodata (in GIS, also the term *lineage* is often used to indicate provenance). This chapter discusses the descriptive relationships that can be created between geo-services, for the purpose of single service discovery and chaining. It starts with the description of an example geo-service chain in Section 6.1. Section 6.2 introduces the basic elements of a service chain as used for reasoning. Section 6.3 discusses how the reference operation descriptions of OPERA-R can be used for derived operation descriptions for both atomic and composite operations in OPERA-D. Section 6.4 presents the reasoning mechanisms that use these description for calculating inferences, for the purpose of matching service requests and advertisements. The chapter ends with a summary and reflection.

6.1 Example: Riskmap chain

For demonstration purposes a simple service chain has been created as a sequence of four services (see Figures 6.1 and 6.2). The aim of this service chain is to generate a map with information about potentially hazardous objects such as ammonia and fireworks depots, centred around a location provided by an end user. Each service in the chain is described by one operation with the same name as the service. The

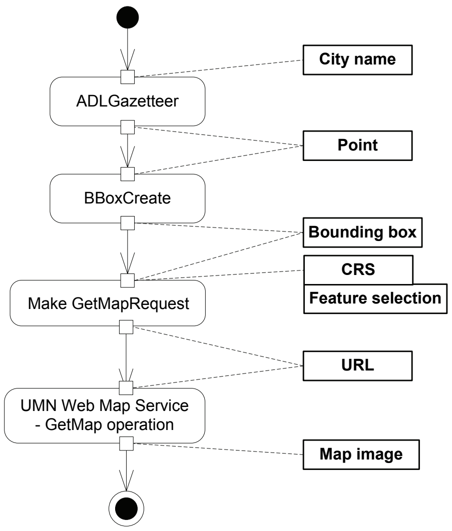


Figure 6.1: UML activity diagram representing a service chain that displays a risk map of an area around a location provided by an end-user. The boxes on the right-hand side represent in/output parameters; CRS means Coordinate Reference System.

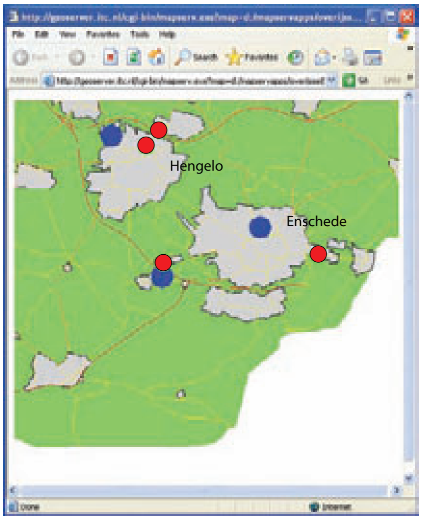


Figure 6.2: Result of risk mapping service chain with location 'Enschede' as input. The map shows the area around the city of Enschede with fireworks depots (small (red) circles) and ammonia storage locations (large (blue) circles).

first part of the chain is implemented by the Alexandria Library Gazetteer service [115], which returns the coordinates of a point, based on an input location name. The second service (*BBoxCreate*) creates a bounding box around this point and a third (*MakeGetMapRequest*) creates a URL that contains the necessary parameters for the fourth service, an OGC-compliant Web Map Server, implemented with the University of Minnesota WMS software. Note that the result in Figure 6.2 is shown in a web browser, which acts as a client application. This client application is not part of the service chain itself. More on the creation of this chain can be found in [165].

6.2 Semantic modelling of geo-service chains

The purpose of modelling a service chain is twofold. First, it supports discovery through inferencing on its component services, its operations and its control flow. Second, it supports syntax/structure-providing service descriptions in concretely composing the service chain. There exist various possibilities to model a service chain within the semantic framework of Chapter 5. Their suitability depends on

(1) the particular needs for the above-mentioned discovery of the chain and its component services, and (2) the needs to communicate the chain's characteristics to execution mechanisms, different from the one which performs the discovery and composition.

Data flow

The anatomy of a service chain has been discussed in Section 2.1.1 and has been illustrated in Figure 2.3. Each single feature processing operation of a geo-service chain reads and returns particular characteristics of features. In our model, these characteristics are represented by the parameter types as specified in Section 5.1.1. During invocation, the formal parameters are substituted with parameter values; in a service chain they are passed between service instances and constitute a data flow. In a service chain, there are many places where the input of a service is obtained from the output of a preceding service [181]. We will use this type of data flow for reasoning in Section 6.4.2.

Control flow

The control flow of a composite service specifies the chaining pattern of its component services. A control flow may describe this pattern using the service operations as components. The Risk map chain of Section 6.1 is written in shorthand notation (introduced in Section 4.7.1) as:

```
<sequence>
    ADLGazetteer : opera:LocSpat
    BBoxCreate : opera:BoundingBox
    MakeGetMapRequest : opera:BuildRequest
    UMN_WMS : opera:ExtractMap
</sequence>
```

The names with prefix *opera:* indicate the operation *types* that appear in the OPERA ontology. In the above sequence, no data flow is specified. This is done in more specific data flow definitions, which are, for example, available in OWL-S [67]. How such control flow may be described in DL is discussed in Section 6.4.5.

Simple OWL sequence

A simple sequence construct can be defined in OWL as follows:

$$\begin{aligned} &\mathit{opera{:}GeoOperation} \sqsubseteq \\ &\quad (\forall\, \mathit{opera{:}hasSequenceNext.opera{:}GeoOperation}) \sqcap \\ &\quad (=1\ \mathit{opera{:}hasSequenceNext}) \end{aligned} \tag{6.1}$$

Note that an operation has only one subsequent operation in the case of a specific instance of a service chain. Such a construct makes it possible to

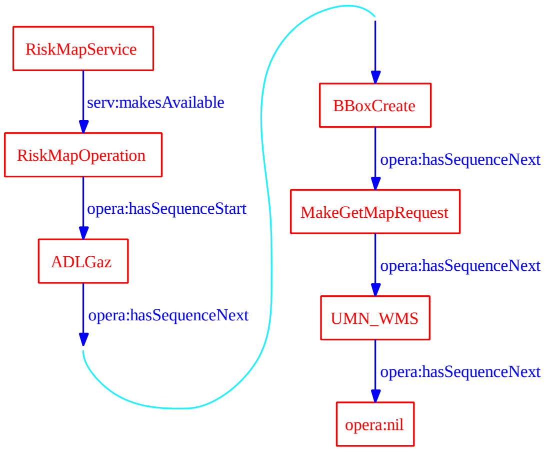


Figure 6.3: Ontology capture diagram showing a risk mapping service chain modelled with a simple OWL sequence construct.

chain instances as a sequence. The end of a chain is marked by instantiating *opera:GeoOperation* with *opera:nil*. The Riskmap service chain is instantiated as in Figure 6.3.

Its parameter binding mechanism is shown in Figure 6.4 for its two operations *ADLGaz* and *BBoxCreate*.

The parameter typing mechanism has been discussed in Section 5.1.1. The parameter identifiers, like *InputPar_18*, are internal identifiers, generated by the ontology editor.

An even more simple construct is formed by the *isComposedOf* role, which is useful for highly abstracted service chain requests and advertisements. It specifies nothing else than that an operation is composed of other operations, without stating any control flow pattern (this reflects also the principle of the Service Organiser Folder (SOF) in ISO 19119 (Services), as described in Section 3.2.3):

$$\begin{aligned}&\textit{opera:GeoOperation} \sqsubseteq \\ &\quad (\forall\ \textit{opera:isComposedOf.opera:GeoOperation})\end{aligned}$$

With this construct, the Risk map service chain can be instantiated as in Figure 6.5.

OWL-S

OWL-S provides more powerful control flow constructs than the aforementioned simple OWL sequences. Besides sequence, it allows for split, iterate, etc. (see Section 4.7.1). The structure of the service chain, modelled in OWL-S is depicted

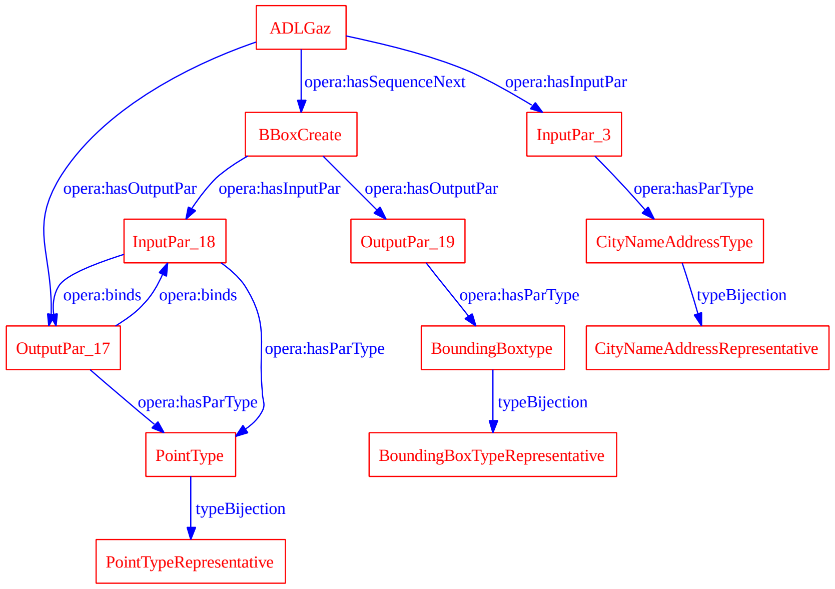


Figure 6.4: Ontology capture diagram showing the parameter binding of two subsequent operations of the risk mapping service chain of Figure 6.3.

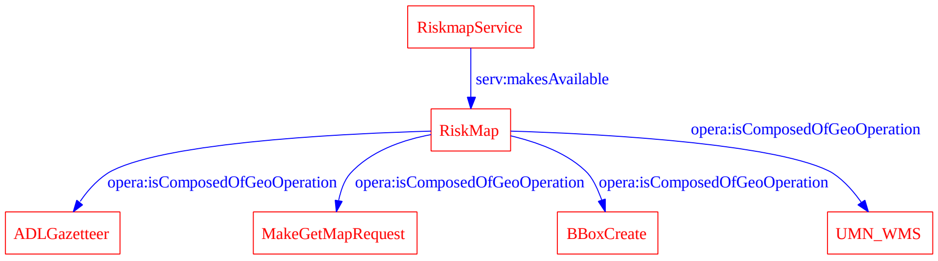

RiskmapService
serv:makesAvailable
RiskMap
opera:isComposedOfGeoOperation
opera:isComposedOfGeoOperation
opera:isComposedOfGeoOperation
opera:isComposedOfGeoOperation
ADLGazetteer
MakeGetMapRequest
BBoxCreate
UMN_WMS

Figure 6.5: Ontology capture diagram showing a risk mapping service chain modelled with a 'isComposedOf' construct. It does not specify a control flow.

in Figure 6.6. The boxes represent instances of OWL-S process concepts. Amongst

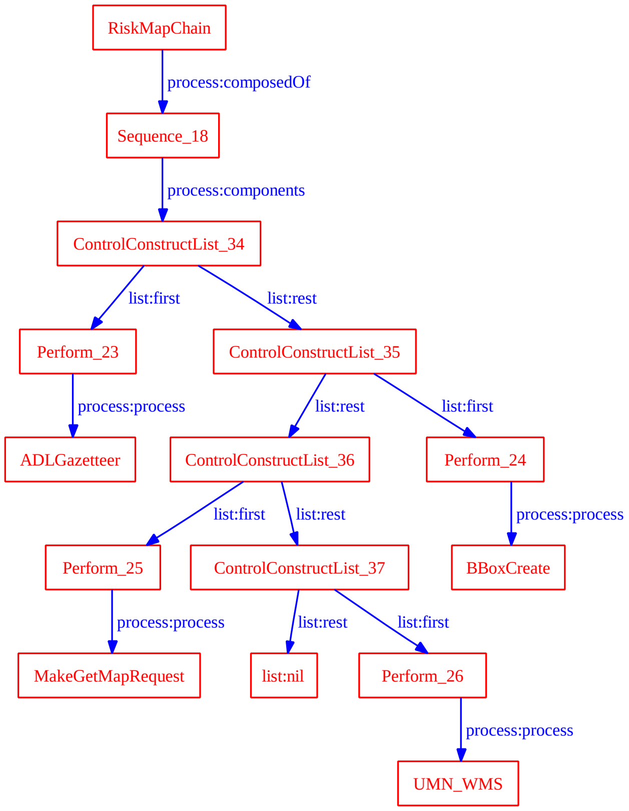


Figure 6.6: Ontology capture diagram showing a risk mapping service chain modelled in OWL-S. In addition to the sequence control flow pattern shown in this example, other patterns such as split and iterate are supported.

them are the geo-operations (ADLGazetteer, BBoxCreate, MakeGetMapRequest) and supporting control constructs (Sequence, Perform, etc.). The postfix numbers are automatically generated by the ontology editor. The sequence pattern can be recognised by following the 'first-rest' control flow as portrayed as a UML activity in Figure 6.1. The 'first-rest' control constructs have been illustrated in Figure 4.7.

6.3 Derived operations and ontology mappings

Derived operations are atomic or composite operations that are described in terms of an OPERA-R class or other derived operations. They are contained in the

OPERA-D ontology in a way that the ontology class *opera:DerivedGeoOperation* serves as a placeholder for them. Derived operations may involve:

- Classes of another model (such as ISO19119 (Services)). These are established by *mappings* of the type that is also used for the symbol and concept ontologies in Section 5.3. Examples of such mappings are provided in Sections 6.3.1 through 6.3.3.
- Class definitions of actual operation implementations, established by *service descriptions* as described in Section 5.6.1.

6.3.1 ISO 19119 mapping

A full mapping has been made between OPERA and the ISO 19119 (Services) taxonomy of services (see Appendix G). In these mappings it is assumed that each ISO 19119 service makes available one, possibly composite, operation. Some examples are now explained. The mapping below has to be interpreted as: The (composite) operation made available by the ISO19119 *ImageGeometryModelConversionService* is subsumed by the *Resampling* class in OPERA.

iso19119:ImageGeometryModelConversionService $\sqsubseteq$ *opera:Resampling*

Note that the ISO19119 service may have other subsumption relationships with concepts in other ontologies. The use of a mapping of this kind for reasoning is discussed in Section 7.4.3. Mappings may also include composite operations:

iso19119:SamplingServiceSpatial $\sqsubseteq C$

in which C is an ontological concept that describes the *iterate* control flow pattern below (written here with the shorthand notation introduced in Section 6.2). How such control flow may be described in DL is discussed in Section 6.4.5.

```
<Iterate>
    opera:ExtractGeoInfoFromStream
    opera:SelectByTopo
</Iterate>
```

The use of a mapping of this kind for reasoning is discussed in Section 7.3. In some cases, the ISO 19119 services are defined with a mapping to another ISO 19119 service class, which has been mapped to an OPERA class already.

6.3.2 Multiple ontology mapping

Mapping between multiple ontologies is an important asset for service discovery, because:

- it allows to find matching services across models, and
- models become more rigidly defined, when they are interlinked.

Mappings with descriptions of commonly used operation types increases the reach of the discovery. This section provides mapping examples of two domains with commonly used operations, i.e., OGC services and mainstream proprietary GIS products.

Mappings with OGC services

In addition to the mappings with ISO 19119 (Services), the OPERA ontology can also be mapped to the OGC's services model (OWS). As a design strategy, the OGC services are mapped as subclasses of ISO 19119 classes and not as direct subclasses of OPERA classes. After all, ISO19119 classes have been already mapped as subclasses of OPERA classes. Such a 'subsumption chain' is considered to be a stronger ontological construction for reasoning purposes than a sibling construction (ISO19119 and OGC both subsumed by OPERA). The choice is justified, because OGC's service model is strongly related to the ISO model and is therefore better represented by direct subsumption relationships. Another option would be to make certain OGC classes equivalent to ISO19119 classes. However, by design, ISO19119 are of a more general nature.

OGC Web Map Service OGC's WMS can be mapped as follows:

ogc:WebMapService $\sqsubseteq$ *iso19119:MapAccessService*

OGC Web Feature Service OGC's WFS can be mapped as follows:

ogc:WebFeatureService $\sqsubseteq$ *iso19119:FeatureAccessService*

OGC OpenLS The OpenLS specification is an effort to standardise the interfaces between the various types of LBS services and LBS clients. It defines abstract data types (e.g., Address ADT) and defines the five core LBS service types together with the request and response parameters for each service type. Below, the service types are listed with a short description in italics, taken from [216] and a mapping to ISO 19119 (Services), based on the more elaborated definitions in the OpenLS specification:

- Directory Service. *An online directory to find the location of a specific or nearest place, product or service.*

$$\textit{ogc:OpenLSDirectoryService} \sqsubseteq \textit{iso19119:ProductAccessService} \tag{6.2}$$

- Gateway Service. *A network-accessible service that fetches the position of a known mobile terminal.*

$$\textit{ogc:OpenLSGatewayService} \sqsubseteq \textit{iso19119:PositioningService} \tag{6.3}$$

- Location Utility Service (Geocoder/Reverse Geocoder). *A network-accessible service that transforms a description of a location, such as a place name, street address or postal code, into a normalised description of the location with a point geometry.* The Location Utility Service does not have a direct correspondence with a ISO19119 (Services) leaf class. Therefore a more detailed mapping is needed. In the OGC ontology, the Location Utility Service subsumes a Geocoder and reverse Geocoder:

$$\textit{ogc:OpenLSGeocoder} \sqsubseteq \textit{ogc:OpenLSLocationUtilityService} \tag{6.4}$$

$$\begin{aligned}&\textit{ogc:OpenLSReverseGeocoder} \sqsubseteq \\ &\qquad \textit{ogc:OpenLSLocationUtilityService}\end{aligned} \tag{6.5}$$

The first has a corresponding leaf class in ISO19119 (Services), but the latter does not, so it may be allocated at a higher hierarchy level in the ISO 19119 class hierarchy:

$$\textit{ogc:OpenLSGeocoder} \sqsubseteq \textit{iso19119:GazetteerService} \tag{6.6}$$

$$\begin{aligned}&\textit{ogc:OpenLSReverseGeocoder} \sqsubseteq \\ &\qquad \textit{iso19119:GeoModelInfoManagementService}\end{aligned} \tag{6.7}$$

An alternative mapping can be created with OPERA:

$$\textit{ogc:OpenLSGeocoder} \sqsubseteq \textit{opera:LocSpat} \tag{6.8}$$

$$\textit{ogc:OpenLSReverseGeocoder} \sqsubseteq \textit{opera:SpatLoc} \tag{6.9}$$

In fact, the mapping represented by axiom 6.8 was already accomplished by Axiom 6.6 and the Axiom ($\textit{iso19119:GazetteerService} \sqsubseteq \textit{opera:LocSpat}$) in Appendix G.

- Presentation Service. *A network-accessible service that portrays a map made up of a base map derived from any geospatial data and a set of ADTs as overlays.*

$$\textit{ogc:OpenLSPresentationService} \sqsubseteq \textit{iso19119:MapAccessService} \qquad (6.10)$$

- Route Service. *A network-accessible service that determines travel routes and navigation information between two or more points.*

$$\begin{aligned}&\textit{ogc:OpenLSRouteService} \sqsubseteq \\ &\quad \textit{iso19119:RouteDeterminationService}\end{aligned} \qquad (6.11)$$

Mappings with functionality of proprietary GIS products Mappings between the ontologies described above and operations, made available in proprietary GIS products, some of which are listed in Section 2.5, can also rigidify the existing ontologies and improve service discovery. The blueprints of a GIS product (e.g., ESRI's ArcObject UML model diagrams) and software documentation may serve as a basis for these mappings. As an example, ESRI's ArcWeb Services provides a route finder service API, which can be mapped as subclass of *opera:RouteOptimisation* (*opera:RouteOptimisation* is defined in Appendix C). In addition, it provides an interface for OpenLS, so it also subsumed by *ogc:OpenLSRouteService.* A more detailed mapping is needed if one needs to describe the options that are provided by this service, i.e., the choice of shortest or quickest route.

6.3.3 Definitions of actual operation implementations

In addition to service *models* such as OPERA and ISO 19119 (Services) we can define service operation implementations as derived operation, similar to the ADL Gazetteer description in Section 5.6.1 (Description 5.9). In the discovery process, such derived operation serves to further tie existing ontology definitions. Further, it can be used as example for users to specify their query in service discovery, similar to the idea of query-by-example [294].

6.4 Matchmaking

Geo-service discovery involves the identification of service advertisements that match a service request. Apart from behavioural aspects, matches are sought between requested and advertised input/ouput parameters. Matchmaking may involve service parameter concepts as well as data set concepts. Further, matchmaking, as described here, is done in a semantic framework. The ontological concepts and individuals involved are *abstract* representatives of the actual service parameters and/or database entities. Despite the fact that ontologies are capable

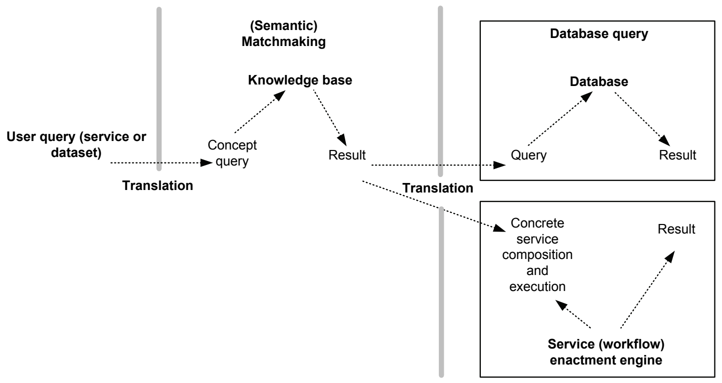


Figure 6.7: Semantic matchmaking provides conceptual links between information models and may form an intermediate between a user query and a database query or service execution. It does not query data directly in the database nor does it invoke a service directly.

of also containing the *concrete* parameters and entities, they are not used for this purpose here. Figure 6.7 shows that there needs to be a translation between a user query and a query, posed to the knowledge base, and a translation of the result of the latter to either a database query or a service invocation. The top-right box and bottom-right box represent respectively the information and operation part described in Section 5.1.1.

The following sections describe the methods that support the matchmaking process. After introducing a notational convention in Section 6.4.1, different matchmaking modes are introduced in Section 6.4.2. Methods for ontology reasoning are explained in the context of our semantic interoperability framework in Sections 6.4.3 through 6.4.6.

6.4.1 The service descriptor, a shorthand notation

For further discussion, a shorthand notation is introduced for the *descriptor* of a service, which comes in two forms:

Term 6.1 *A* concept-based service descriptor *is an ontological concept indicated with a capital C which (partially) describes characteristics of a service, such as the input and output parameters of its operations and its workflow.*

Partially means that the descriptor may not contain *all* characteristics of a service. For example, C may be defined solely by Description 5.9 or by also including the Role restriction 5.10.

Term 6.2 *An* individuals-based service descriptor *is an ontological individual indicated with a lower case c which represents characteristics of a service, such as the input and output parameters of its operations and its workflow.*

Note that an individuals-based descriptor is indicated with a lower case c. c represents the characteristics of the service by being linked to a set of individuals, through role assertions. An example of such a set of linked individuals can be found in Figure 5.15 (individuals are depicted by boxes with red edges).

If the service descriptor involves only input parameters or output parameters, then the notation C_{in} respectively C_{out} is used for concept-based service descriptors and c_{in} respectively c_{out} for individuals-based service descriptors.

6.4.2 Matchmaking modes

This section elaborates on the concept of matchmaking, applied to isolated services or as part of a service chain. The shorthand notation introduced in Section 6.4.1 is used. In matchmaking, we distinguish between a *requesting* service descriptor and *advertised* service descriptors. They are denoted as R respectively A for concept-based service descriptors and r respectively a for individuals-based service descriptors. During the matchmaking process, a requesting service descriptor is matched with each of the advertised service descriptors.

The following four degrees of freedom for matchmaking are considered relevant in the context of this thesis work:

- Matching with concept-based descriptors, with individuals-based descriptors or with both.
- Matching by means of splitting concept intersections.
- Requesting for an isolated service, versus requesting a service as part of a chain.
- Matching specific service characteristics (I/O parameters, control flow, or overall class).

Their variations are orthogonal and a concept match is characterised by an option out of each of the four degrees of freedom. The variations are described below.

Concepts versus individuals

Depending on whether we use concept-based service descriptors or individuals-based service descriptors, there are four possible match types:

$$\begin{array}{lll} match\,(R, A) & \text{type I} & \text{(between concepts)} \\ match\,(R, a)) & \text{type II} & \text{(between a concept and individuals)} \\ match\,(r, A)) & \text{type III} & \text{(between an individual and concepts)} \\ match\,(r, a) & \text{type IV} & \text{(between individuals)} \end{array} \tag{6.12}$$

They are defined below.

- Match type I

  match (R, A) is a function which takes as input a TBox (terminological part of a knowledge base, see explanation in Section 4.2.1), a requesting concept R and an advertised concept A, and returns one out of the following five result types: Exact, PlugIn, Subsume, Intersection, Disjoint. These result types are further discussed in Section 6.4.3.

- Match type II

  match (R, a) is an instance checking function which takes as input a requesting concept R and an ABox (assertional part of a knowledge base, see explanation in Section 4.2.1), and returns all the individuals a that are instances of R.

- Match type III

  match (r, A) is an instance checking function which takes as input an individual r (r represents a requesting service) and an ABox, and returns all concepts A of which r is an instance.

- Match type IV

  match (r, a) is a similarity function which takes as input an ABox, an individual r (which represents a requested service) and an individual a (which represents an advertised service), and returns a value as measure of similarity between the two individuals.

Matches of type I are performed by Tbox reasoning. The other three match types are performed by Abox reasoning. Differences between TBox and ABox reasoning have been elaborated upon in Section 4.5. The practical aspect of creating definitions, in this case for services, plays an important role in the realisation of (prototype) implementations. The creation of individual-based service definitions makes use of an already defined structure of class definitions (including role assertions) and is therefore less flexible, but principally more a matter of 'filling in' the concepts with instances. Such experiences are described in Chapter 8. The specific application of TBox and ABox reasoning is described in respectively Sections 6.4.3 and 6.4.4.

Splitting of concept intersections

Service descriptions are typically consisting of intersections of concepts, each of which represents specific characteristics of a service, like its input parameters. As we will see in Section 6.4.3, for matchmaking it makes sense to split such an intersection in its concept parts and match them individually. This approach has been identified by [257] as *query relaxation*, in which specific variables of a conjunctive query are removed from the query to target a specific match.

In this respect there are four types of matching. We denote a description containing concept intersections as $R_{\sqcap}$ or $A_{\sqcap}$ for respectively request and advertisement. A partial description (not containing intersections) is denoted as R_{part} respectively A_{part}:

$$\begin{array}{l} match\ (R_{\sqcap}, A_{\sqcap}) \\ match\ (R_{part}, A_{\sqcap}) \\ match\ (R_{\sqcap}, A_{part}) \\ match\ (R_{part}, A_{part}) \end{array} \tag{6.13}$$

The above match types are all of type I. Similarly they can be written for concept-based descriptions in match types II and III. For example, for a match type II we can distinguish:

$$\begin{array}{l} match\ (R_{\sqcap}, a) \\ match\ (R_{part}, a) \end{array} \tag{6.14}$$

Isolated service versus chain

A distinction is made between (1) the evaluation of a single service in isolation (from here on referred to as 'isolated service') and (2) a service that is part of a chain (see a depiction of both in Figure 2.3). For searching an isolated service, a requesting concept R should contain a role *hasInputPar* to specify the requested input parameters and a role *hasOutputPar* to specify the requested output parameters. For searching a service that is part of a chain, the parameters of candidate *preceding* or *following* services are matched with those of the target service. In this case, a requesting concept R should contain a role *hasInputPar* to specify the requested input parameters of a following service in the chain and a role *hasOutputPar* to specify the requested output parameters of a preceding service in the chain. The requesting concept may make use of a partial description R_{part} (as introduced above) that contains only input or output parameters. As a notational convention, we write a requesting concept R_{part} with only input parameters as R_{in} and one with only output parameters as R_{out}.

Matching specific service characteristics

The concept match may involve the testing of I/O parameters, control flow, or ontological relations with an overall service class description. Control flow testing is described in Section 6.4.5.

6.4.3 TBox reasoning

A match of type I (between concepts) can basically have five result types. The match is:

- **Exact** if R and A are equivalent concepts, formally R $\equiv$ A. In terms of our request, all individuals in advertised service concept A satisfy the requested service concept R.
- **PlugIn** if R is a subconcept of A, formally R $\sqsubseteq$ A. In terms of our request, there may be some (but not all) individuals in advertised service concept A that satisfy the requested service concept R. For example, if our request is a *house*, then an advertisement of a *building* is a PlugIn match.
- **Subsume** if R is a superconcept of A, formally A $\sqsubseteq$ R. In terms of our request, all individuals in advertised service concept A satisfy the requested service concept R. The difference with an exact match is that A is less general (or: more specific) than R. For example, if our request is a *building*, then an advertisement of a *house* is a Subsume match.
- **Intersection** if the intersection of R and A is satisfiable (or in Protégé terms, consistent), formally $\neg(A \sqcap R \sqsubseteq \perp)$ For example, if our request is a *house*, then an advertisement of an *office* is an Intersection match (if in the ontology, the concepts of *house* and *office* are not disjoint).
- **Disjoint** if the intersection of R and A is not satisfiable (or in Protégé terms, not consistent), formally $(A \sqcap R \sqsubseteq \perp)$. In other words, there are no individuals in advertised service concept A that satisfy the requested service concept R.

Reasoner tests

In the Protégé-Racer reasoner environment (software details can be found in Chapter 8) match tests were conducted with a range of descriptions, each demonstrating a particular interpretation of the reasoner. The followed approach describes geo-information entities, similarly to what efforts in DL research [91, 168, 264] have applied in other domains.

Test case 1: Matching partial descriptions.
A request R is created for searching a feature represented with a geographic coordinate reference system (using latitude and longitude coordinates). As an example, R is matched with an advertisement A of features represented by addresses:

$$\begin{aligned} &match\,(R_{part}, A_{part}) \text{ with} \\ \\ &R_{part} \equiv (\exists symbol{:}hasReference.symbol{:}SC\_\ CRS\_LatLon) \\ &A_{part} \equiv (symbol{:}Address) \end{aligned} \tag{6.15}$$

The reasoner infers that R is a sub class of *symbol:GeometricObject*. As *symbol:GeometricObject* and *symbol:Address* are defined in the ontology as disjoint, the reasoner classifies R and A as disjoint. In other words, there are no individuals in the advertised concept A that satisfy the requested concept R.

Test case 2: Matching with concept intersections.

$$\begin{aligned}
&\mathit{match}\,(R_{\sqcap}, A_{\sqcap}) \text{ with}\\
&R_{\sqcap} \equiv (\mathit{symbol{:}Point}) \sqcap\\
&\qquad (\exists\forall \mathit{symbol{:}hasReference.symbol{:}Projected})\\
&A_{\sqcap} \equiv (\mathit{symbol{:}GeometricObject}) \sqcap\\
&\quad (\exists\forall \mathit{symbol{:}hasReference.symbol{:}UTM})
\end{aligned} \tag{6.16}$$

In this example, R has a coordinate reference system characteristic *symbol:Projected* that subsumes the *symbol:UTM* concept of A. Vice versa, the *symbol:GeometricObject* of A subsumes *symbol:Point* of R. In a match with partial descriptions we would discover these subsumption relationships but in a match with descriptions containing intersections, we do not identify these subsumption relationships, we only observe that R and A intersect. A similar problem occurs with one partial description and one description containing intersections:

$$\begin{aligned}
&\mathit{match}\,(R_{part}, A_{\sqcap}) \text{ with}\\
&R_{part} \equiv (\mathit{symbol{:}Point})\\
&A_{\sqcap} \equiv (\mathit{symbol{:}Point}) \sqcap\\
&\qquad (\exists\forall \mathit{symbol{:}hasReference.symbol{:}Projected})
\end{aligned} \tag{6.17}$$

In this case, a test will result in a subsumption $A \sqsubseteq R$. Although a match with solely partial descriptions would display the equivalence relation between the 'Point' role restriction of R and A, the above match 'hides' this equivalence.

6.4.4 ABox reasoning

ABox reasoning involves concept matches of types II, III or IV (see Formulas 6.12 in Section 6.4.2).

Type II (between a concept and individuals): Matches of this type have been implemented in the prototype. The matching is performed by the RacerPro reasoner with the command *(concept-instances R)*. The following example match searches for a subsequent operation to the ADLGazetteer operation, as discussed in Section 6.1:

$$match\ (R_{part}, a) \tag{6.18}$$

where R represents the output parameters of the ADLGazetteer operation and a is sought. R is defined as the query concept:

$$\begin{aligned} &ADLGazetteerOut \equiv \\ &\quad (\exists\ opera{:}hasInputPar.(\exists\ opera{:}hasParType. \\ &\qquad (\exists\ opera{:}typeBijection.opera{:}OP\_Point))) \sqcap \\ &\quad (\exists\ opera{:}hasInputPar.(\exists\ opera{:}hasParType. \\ &\qquad (\exists\ opera{:}typeBijection. \\ &\qquad (\exists\ opera{:}hasReference.symbol{:}SC\_CRS\_LatLon)))) \end{aligned} \tag{6.19}$$

An example of an individual a that matches the query is *BBoxCreate*, and is depicted in Figure 6.4.

Type III (between an individual and concepts): Matchmaking type III requires all advertisements A to be specified as concept definitions and the request r to be one of a set of linked individuals. A match of this type is performed with the RacerPro reasoner command *(individual-types r)*.

Type IV (between individuals): Matchmaking type IV can be performed by comparing two individuals r and a as part of a set of linked individuals. This type of matching has not been implemented in this research, due to time constraints. However, some ideas for applying this match type are given below. A simple example is provided in Figure 6.8.

In the figure, r is *Point_7* as part of *Point_7—LatLon_8* and a is *Point_6* as part of *Point_6—LatLon_1*. *Point_6* is found to be similar to *Point_7*, because they are instances of the same class (*symbol:Point*). A similar correspondence exists for their coordinate reference systems (*LatLon_1* and *LatLon_8*). The example match applies to geographic information entities, but can also be applied to input and output parameters of geo-operations. More complex similarity matches have to take into account *similarity functions*, such as proposed in [178] and (by including the instance types) [241].

6.4.5 Reasoning with control flow

This section describes how control flow is used in reasoning. We adopt the process model constructs of OWL-S, as described in Section 4.7.1. A control flow is assumed to be constituted of one or more of the following constructs: *Sequence, Split, Split+Join, Choice, Any-order, If-then-else, Iterate, Repeat-while, Repeat-until*. Knowledge about control flow can be valuable in service discovery. For example, composite advertised services that expose their control flow, allow for more specific concept matches than advertisements that do not expose their control flow; ditto for service requests. Control flow can be expressed in many different ways. The two options of Section 6.2 (A simple OWL sequence (not based on OWL-S) and one modelled with OWL-S) are elaborated below with the risk mapping chain of Section 6.1.

Simple OWL sequence

Consider the Riskmap chain as a composite operation. Its control flow can be written as a DL axiom as follows (only valid for sequence):

$$\begin{aligned}
&\mathit{RiskmapOperation} \sqsubseteq \\
&(\exists\ \mathit{opera{:}hasSequenceStart.opera{:}ADLGazetteer}) \sqcap \\
&(\exists\ \mathit{opera{:}hasSequenceStart.}(\\
&\quad \exists\ \mathit{opera{:}hasSequenceNext.opera{:}BBoxCreate})) \sqcap \\
&(\exists\ \mathit{opera{:}hasSequenceStart.}(\\
&\quad \exists\ \mathit{opera{:}hasSequenceNext.}(\\
&\qquad \exists\ \mathit{opera{:}hasSequenceNext.opera{:}MakeGetMapRequest}))) \sqcap \\
&(\exists\ \mathit{opera{:}hasSequenceStart.}(\\
&\quad \exists\ \mathit{opera{:}hasSequenceNext.}(\\
&\qquad \exists\ \mathit{opera{:}hasSequenceNext.}(\\
&\qquad\quad \exists\ \mathit{opera{:}hasSequenceNext.opera{:}UMN\_WMS})))) \sqcap \\
&(\exists\ \mathit{opera{:}hasSequenceStart.}(\\
&\quad \exists\ \mathit{opera{:}hasSequenceNext.}(\\
&\qquad \exists\ \mathit{opera{:}hasSequenceNext.}(\\
&\qquad\quad \exists\ \mathit{opera{:}hasSequenceNext.}(\\
&\qquad\qquad \exists\ \mathit{opera{:}hasSequenceNext.opera{:}nil}))))
\end{aligned} \tag{6.20}$$

The description does not contain value restrictions, because the role *hasSequenceNext* has a value restriction ‘=1’ (see its definition in Section 6.2). This means that an operation can only participate once in a *hasSequenceNext* relationship and the use of a ‘forall’ restriction in the above axiom would not make sense.

The description can be used as advertisement or request concept (in the case of a request, written with $\equiv$ instead of $\sqsubseteq$) in concept matches of type I, II or III. Individuals in these matches are of the form as presented in Figure 6.3 and 6.4.

Note that each of the five parts of the above description represents a subsequent operation. Obviously, a sequence construct of this kind is simple in nature, but

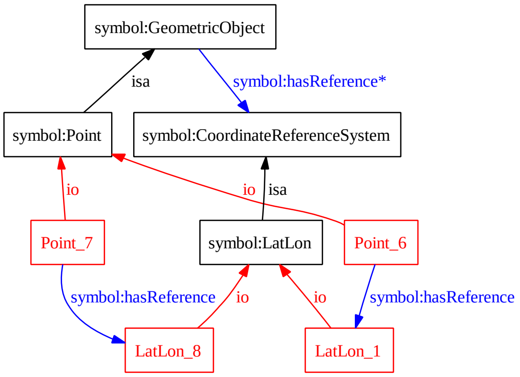


Figure 6.8: Ontology capture diagram showing matchmaking type IV: connecting individuals representing two points.

has a verbose representation. However, a requesting concept does not necessarily contain all five parts. For example, if our aim is to find a service with a gazetteer as first operation, than only the first part is needed. A similar case occurs when we search for an OGC Web Map Server at the end of the chain, then only the last part of the description is needed.

Further, note that Description 6.20 assumes that each operation is defined as a *class* in the ontology. If we want to search for similar operations as components of the chain, then each requested operation can be described by its superclass, e.g., *opera:ADLGazetteer* becomes *opera:LocSpat*. A third option is to target individuals, e.g., the first lines of Description 6.20 become

$$(\mathit{opera{:}hasSequenceStart} \ni \mathit{ADLGazetteer})$$

and the second axiom becomes

$$(\exists\ \mathit{opera{:}hasSequenceStart}.(\mathit{opera{:}hasSequenceNext} \ni \mathit{BBoxCreate})$$

Composed of The simple construct *opera:isComposedOf* can be used for matchmaking if only the component operations of a control flow are relevant in a request. Again, one can specify single or multiple components. A single component request is for example:

$$(\exists\ \mathit{opera{:}isComposedOf.opera{:}ADLGazetteer})$$

OWL-S

In contrast to the simple control flow structures above, the more complex OWL-S control flow structures provide more possibilities for reasoning, but also make the service description procedure and the reasoning process more complicated. OWL-S expresses its control flow with a set of linked individuals (see Figure 6.6). A requesting concept may search the whole or parts of the chain. As an example, we will search for a partial sequence of an operation of type *LocSpat* (typically a gazetteer operation), immediately followed by a bounding box creator operation. This request can be formulated as follows:

$$R \equiv$$

$$(\exists \textit{ list:first } (\exists \textit{ process:process.opera:LocSpat})) \sqcap$$
$$(\exists \textit{ list:rest } (\exists \textit{ list:first } (\exists \textit{ process:process.opera:BoundingBox})))$$

When concept matching type II is performed, the following instance will be found (test performed with the RacerPro reasoner): *ControlConstructList_34* (see Figure 6.6). Obviously, we want to know to which service this instance belongs. This can be done by tracing it to the instance a in $R(a, b)$, with R being the OWL-S role *process:composedOf*. Another possibility is to simplify the composition information by neglecting the control flow and evaluating only the collection of component operations, as in the *opera:isComposedOf* construct above. However, in OWL-S such a construct is not available and to extract the component operations, one has to navigate to them through the first-rest relations in the set of linked individuals.

Single service discovery A single service, which may be a component of a service chain, can always be discovered with a simple query (type II) of the form:

$$R \equiv \textit{opera:LocSpat}$$

This will result in the discovery of, for example, the *ADLGazetteer* operation. When an operation, for example *ADLGazetteer*, is part of a chain, than it is modelled both as an instance of *opera:LocSpat* and of *process:process.* This ensures that further inspection of the service reveals it is part of a chain. Such inspection can be performed by querying the graph structure of which *ADLGazetteer* is a part (see Figure 6.6).

6.4.6 Exploitation of ontology mappings

Mappings between ontologies can be used to discover a resource advertisement defined in one ontology, with a request defined in another. It also enables the discovery of advertisements over multiple ontologies. Mappings can involve information concepts (see Section 5.3) and operation concepts (see Section 6.3).

Examples of the use of mappings in reasoning can be found in Sections 7.1 and 7.4 for respectively geo-information and geo-operations.

6.5 Summary and reflection

This chapter has discussed the application of the semantic interoperability framework of Chapter 5 for the purpose of describing services and service types and reasoning with them.

When translating textual definitions of service types into ontological concepts, some deficiencies in these definitions became apparent. Some examples were shown for the OGC and ISO 19119 service taxonomies. Although the geo-service taxonomy, contained in the ISO19119 (Services) standard provides a useful generic breakdown of service classes, it falls short as a single basis for a geo-service ontology, due to the following reasons:

1. The semantics of several classes are not well-defined.
2. Several subsumption relations between classes are not identified.
3. Several classes show a large overlap.
4. For some classes, its granularity is too coarse to distinguish common sub-operations sufficiently, for example with respect to spatial overlay and measurement.

Reasoning and design patterns of service descriptions

Reasoning is a combined play between a request and advertisements and there are many choices to be made to construct them in a way that makes sense for a reasoner. One example is the choice between existential quantifiers and value restrictions. The existential quantifier is the most common role restriction that is used in OWL ontologies [118]. In some cases it does not give enough ‘restrictive power’ and therefore, another common pattern is the conjunction of an existential quantifier and a value restriction (∃∀...).

Matchmaking modes

Matchmaking can be performed in different modes, as discussed in Section 6.4.2. Class-based requests and advertisements can be described with either concept intersections or with partial descriptions (split concepts with no intersections). Partial descriptions allow for a more specific matching of data/service characteristics, but it requires the splitting of data/service descriptions into parts, which is, if done manually, a tedious process. Further, in the ontology editor, used in this research, the entry of individual-based descriptions is much easier than creating class-based descriptions. A summarising typification of the the different matchmaking modes is provided below:

- Type I (between requesting and advertised classes): The match results are represented by the class relations ‘exact’, ‘disjoint’, ‘subsume’, ’plug-in’ and 'overlap'. In case of descriptions containing concept intersections, the latter three result types have to handled with care, because they 'shield' positive matches of atoms.

- Type II (between requesting class and advertising individuals): Matchmaking is done by requesting the reasoner to find all individuals that are instances of the requesting class. In the current prototype setup, requests are defined as partial descriptions. As advertisements outnumber the requests, this is a convenient mode for entering descriptions (see the last remark in the introduction of this bullet list)
- Type III (between requesting individual and advertised classes): This type is similar to type II, but has an inconvenience in the tedious entering of class-based data/service advertisements.
- Type IV (between requesting and advertising individuals): Allows for convenient advertisement and request entry, but is limited with respect to the possibilities of the current reasoning implementation. The current reasoning process does not provide options to compare graphs of linked individuals. This may be solved by using combinations of reasoner commands and 'navigate' the graph in such way that detailed comparisons are possible. This has not been implemented in the current prototype.

Relaxed matching

The fact that ontologies deploy class subsumption and intersection, facilitates the use of relaxed matching. Information and services may be described imprecisely by using rather general concepts instead of specific ones. This is useful when requests and/or advertisements cannot be formulated in detail. In this context also simplified control flow is possible with the help of the constructs *hasSequenceNext* and *isComposedOf*, compared to the more complex OWL-S control flow structure. However, the mixed use of the three types of control flow models requires a translation between the descriptions formulated in either model.

Application value

Matchmaking has shown to be applicable for input/output parameter type matching and for matching of control flow patterns. Example applications are further worked out in the use cases of Chapter 7.

Chapter 7

Use case implementations

In this Chapter, the use cases, as presented in Section 2.6, are further elaborated upon by applying the methods described in Chapter 5 and 6. Each of the use cases discusses different aspects of the developed semantic interoperability framework and the reasoning about its concepts. Use is made of the GeoMatchMaker prototype, which is integrated with the Protégé ontology editor and the RacerPro reasoner and described in Chapter 8. For all uses cases, the assumption is made that users, who interrogate the ontologies, are either able to write DL-axioms or they are equipped with a query client that translates query formats to DL-axioms. Obviously, the use cases are implemented with a significant involvement of human decision making (a high degree of *human-in-the-loop*). In this context, the emphasis of this chapter is to demonstrate where semantic service descriptions can play a role in the process of semantic service chaining. The sections 7.1 through 7.4 correspond to the introductions given in respectively the Sections 2.6.1 through 2.6.4.

7.1 Riskmap NL

The overall goal of this use case is to improve the interoperability of the risk map to allow service extensions and integrated analysis over multiple information models. As introduced in Section 2.6.1, the key person in this use case scenario is Oscar, a member of the Riskmap team, responsible for carrying out a feasibility study.

7.1.1 Extension of the Riskmap chain[1]

First, the risk map is converted into an OGC Web Map Service (see a screenshot of the implemented prototype service in Figure 6.2). Now consider a scenario where this is the only service available and we want to extent it with the functionality

[1]Parts of this section also appear in [165].

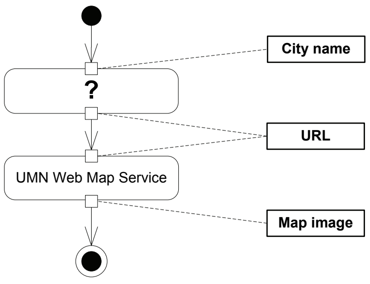


Figure 7.1: UML activity diagram representing the Riskmap service chain with the first part to be discovered.

of a gazetteer service as presented in Section 6.1. However, the preceding services are not directly available and have to be searched for. The elements of the service chain and the aimed output are depicted in Figure 7.1. Oscar proceeds as follows. He requests for one service that converts a city name address to a bounding box with the following request:

$$Request \equiv R_{in} \sqcap R_{out} \tag{7.1}$$

with

$$\begin{aligned} R_{in} \equiv & (\exists \textit{ opera:hasInputPar.}(\exists \textit{ opera:hasParType.} \\ & \quad (\exists \textit{ opera:typeBijection.opera:OP\_CityNameAddress}))) \\ R_{out} \equiv & (\exists \textit{ opera:hasOutputPar.}(\exists \textit{ opera:hasParType.} \\ & \quad (\exists \textit{ opera:typeBijection.opera:OP\_BoundingBox}))) \end{aligned} \tag{7.2}$$

As the above request does not give any result, Oscar tries again with a less restrictive request that will give a result on either an operation's input or output parameter:

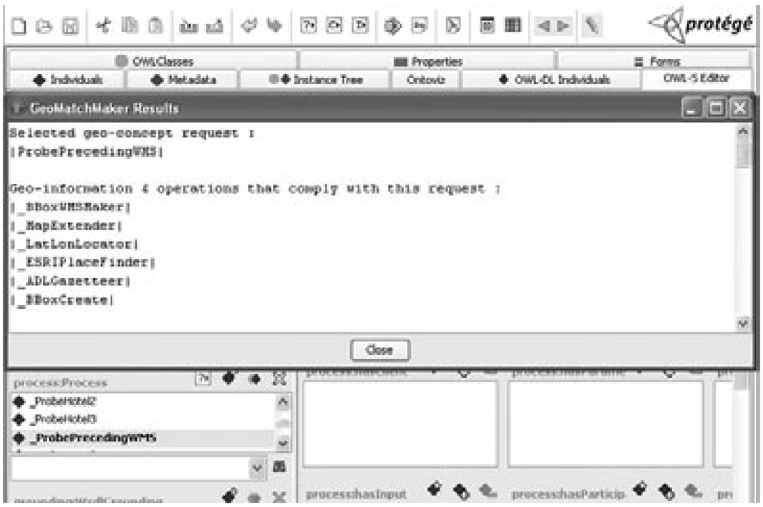
$$Request \equiv R_{in} \sqcup R_{out} \tag{7.3}$$

This query results in a number of gazetteer operations and a number of operations that create a bounding box around a point, see Figure 7.2. None of these single operations can convert a city name address to a bounding box, but a combination of two can, because the output of any of the three gazetteers fits the input of any of the bounding box creators. The gazetteer operations are ESRIPlaceFinder[2], LatLonLocator [3] and ADLGazetteer[4] (see also Section 5.6.1). A choice between

[2]http://www.arcwebservices.com/v2006/livesamples/placefinder/index.jsp

[3]http://www.xmethods.net/ve2/ViewListing.po?key=uuid:F90B2ACA-8C99-9F04-21E7-A455DEC05F1E

[4]http://middleware.alexandria.ucsb.edu/client/gaz/adl/index.jsp



```
Selected geo-concept request :
|ProbePrecedingWMS|

Geo-information & operations that comply with this request :
|_BBoxWMSMaker|
|_MapExtender|
|_LatLonLocator|
|_ESRIPlaceFinder|
|_ADLGazetteer|
|_BBoxCreate|
```

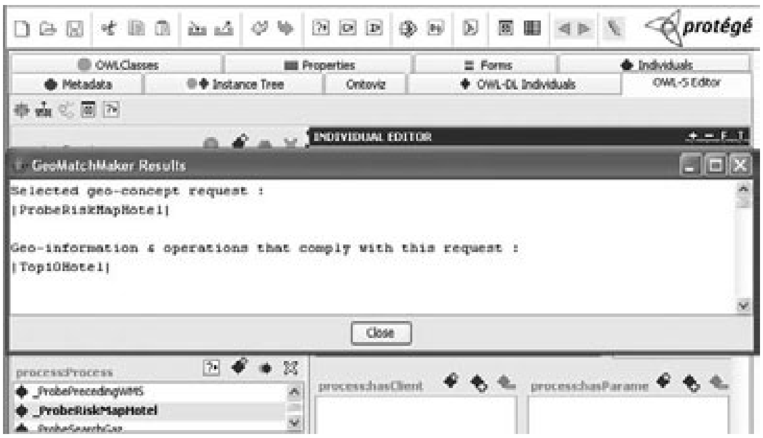
Figure 7.2: Output of the GeoMatchMaker prototype: All service operations are found that have an address as input *or* a point geometry as output.

the three gazetteer services can be made by a match refinement or by evaluating the actual services through their demo URL's. The next step involves the evaluation of the bounding box creator, which is not elaborated here. After selecting the BBoxCreate service, there is one service left to complete the chain. This is a service that must build a GetMapRequest from the bounding box. Information, such as feature selection and coordinate system metadata, which are needed by the GetMapRequest, are also needed as input to this service. The chain is now finished and looks as the one already discussed in Section 6.1. As WSDL documents are available for each of the services in the chain, it is possible to invoke the complete chain of services via a WSBPEL document, that contains the concrete composition. The execution of the WSBPEL document is done by the Oracle BPEL manager[5]. The actual implementation of concrete composition and execution of the chain has been described in [165] and is based on work of Granell [97].

7.1.2 Exploitation of ontology mappings

In a next stage, Oscar is going to test how the actual mappings between the NEN3610, TOP10NL and Riskmap ontologies can support the integrated analysis over multiple geo-information models. In the light of the Riskmap NL use case, suppose a service has to be created that supports the checking of land use plans against safety regulations. Such service may show a map with risk information

[5]http://www.oracle.com/technology/products/ias/bpel/index.html



```
Selected geo-concept request :
|ProbeRiskMapHotel|

Geo-information & operations that comply with this request :
|Top10Hotel|
```

Figure 7.3: Output of the GeoMatchMaker prototype: Top10Hotel is found as a match with the RiskMapHotel query.

and topographic information, with more detail than the public risk maps described earlier. An end-user may now request to see hotels in both Riskmap and TOP10NL models. A direct 'hotel' mapping between the models does not exist. However, two other mappings, i.e., TOP10NL-NEN3610 (see Section 5.3.2) and Riskmap-NEN3610 (see Section 5.3.3) *have* been created and indirectly map Riskmap with TOP10NL. The query below expresses a request in the Riskmap model:

$R \equiv \textit{riskmap:Hotel}$

Due to the mappings between the models, GeoMatchMaker finds all the individuals of top10nl:Hotel as a query result (see Figure 7.3; in the prototype only one individual (*TOP10Hotel*) has been stored and is therefor the only one found). Note that the query result is a result of the knowledge base and not a direct database query (see the explanation of Figure 6.7).

7.2 Emergency 112

The use case, described in Section 2.6.2, is worked out as follows. During the emergency, information is coming in via sensors and messages that people send via their mobile phone. An emergency information system (EMIS) is used by Kora, the information engineer, for storing and retrieving such information. A knowledge base is used to infer relationships between registered information items (which can involve data and services).

7.2.1 Annotation

A connection between EMIS and the knowledge base is created through semantic annotations of incoming information items. This is performed in three ways (see their principles in Section 4.6.3):

- **Annotation method 1:** Adding semantic tags to the incoming information item. This is done when the items are likely to be reused in another semantic framework.
- **Annotation method 2:** Registering information item references in the ontology. This is done for performing ontology reasoning instantly.
- **Annotation method 3:** Registration mapping; if an annotation is stored in a third source, separate from, but holding identifiers for ontology and information source.

In this use case implementation, method 1 is preferred over the third option of registration mapping, because the first can be implemented instantly with currently available tools. The latter would necessitate an additional interface for the ontology to read the mappings. The followed annotation procedure is described below with two examples, one for a sensor message and one for a human observation. They are first described with method 1 and then with method 2.

An example of an incoming sensor message schema is given below (taken from [217]).

```
<om:observedProperty>
  <swe:CompositePhenomenon gml:id="UserDefinedID" dimension="4">
    <gml:name>UserDefinedComposite</gml:name>
    <swe:component xlink:href="urn:ogc:data:time:iso860"/>
    <swe:component xlink:href="urn:ogc:phenomenon:location:EPSG:4326:longitude" />
    <swe:component xlink:href="urn:ogc:phenomenon:location:EPSG:4326:latitude"/>
    <swe:component xlink:href="urn:ogc:def:phenomenon:OGC:1.0.30:WindSpeed"/>
  </swe:CompositePhenomenon>
</om:observedProperty>
```

The above sensor information snippet is a serialisation as specified in the OGC Sensor Observation Service specification [217]. The example is part of the response of a sensor to the GetObservation request as described in the same specification. Kora annotates the sensor information with the *wssem* tags of WSDL-S, described in Section 5.6.2. The annotation of method 1 contains a declaration of ontology namespaces and a set of semantic tags in the body of the sensor message.

Annotation method 1: Adding semantic tags to the information item

First, the following ontology namespace declarations are placed in the header of the sensor source file among the already existing namespaces:

```
xmlns:wssem="http://www.ibm.com/xmlns/WebServices/WSSemantics"
xmlns:Ontology0="http://geoserver.itc.nl/lemmens/owl/symbol.owl"
xmlns:Ontology1="http://geoserver.itc.nl/lemmens/owl/riskmap.owl"
```

Second, semantic tags are placed in the sensor message body. The tags are marked below with boxes:

```
<om:observedProperty>
  <swe:CompositePhenomenon gml:id="UserDefinedID" dimension="4">
    <gml:name>UserDefinedComposite</gml:name>
    <swe:component xlink:href="urn:ogc:data:time:iso860"/>
      wssem:modelReference="Ontology0#DateTimeCode" />
    <swe:component xlink:href="urn:ogc:phenomenon:location:EPSG:4326:longitude"
      wssem:modelReference="Ontology0#Point","Ontology1#LatLon" />
    <swe:component xlink:href="urn:ogc:phenomenon:location:EPSG:4326:latitude"
      wssem:modelReference="Ontology0#Point","Ontology1#LatLon" />
    <swe:component xlink:href="urn:ogc:def:phenomenon:OGC:1.0.30:WindSpeed"
      wssem:modelReference="Ontology1#WindSpeed" />
  </swe:CompositePhenomenon>
</om:observedProperty>
```

The following spoken message is received from an human observer and entered as written text in the emergency information system:

'I am at the Stationstraat and I see smoke coming from East direction'.

The annotation below shows the use of the *wssem:modelReference* construct in a form that slightly differs from the form defined in the WSDL-S specification in [4]. We apply it here as an XML *element* instead of an XML *element attribute.*

```
xmlns:wssem="http://www.ibm.com/xmlns/WebServices/WSSemantics"
xmlns:Ontology0="http://geoserver.itc.nl/lemmens/owl/symbol.owl"
xmlns:Ontology1="http://geoserver.itc.nl/lemmens/owl/riskmap.owl"

<message>
  <wssem:modelReference="Ontology1#Observer"> I </wssem> am at
  <wssem:modelReference="Ontology0#Address"> Stationstraat </wssem> and I see
  <wssem:modelReference="Ontology1#HazardousSubstanceMoveble"> smoke </wssem>
  coming from
  <wssem:modelReference="Ontology1#WindDirection"> East direction </wssem>
</message>
```

Annotation method 2: Registering source references in the ontology

The Riskmap ontology contains place holders for incoming emergency information. For example, each sensor observation is stored as an instance of the *riskmap:-SensorObservation* class (see Figure 7.4). The figure shows that *SensorObserva-*

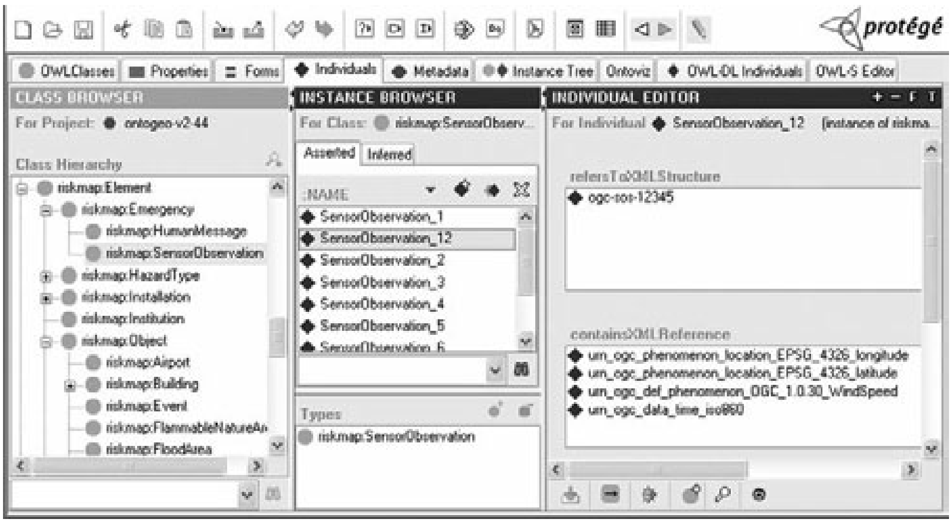


Figure 7.4: Annotation of sensor observations in the Riskmap ontology. The figure shows the XML references of one observation in the Protégé ontology editor.

tion_12 contains XML references on the sensor's location (latitude, longitude), its measured quantity (WindSpeed) and the timestamp of measurement (time_iso860). These references are based on OGC URNs that appear in the GetObservation request for this sensor. Figure 7.5 shows for one reference ('latitude') how it is registered to the ontology concepts *Point*, *CoordinateSet* and *LatLon*; all instances of concepts in the SYMBOL ontology. Finally, Kora can see from the reference to the XML structure (see Figure 7.6) where the XML structure can be found.

7.2.2 Information retrieval

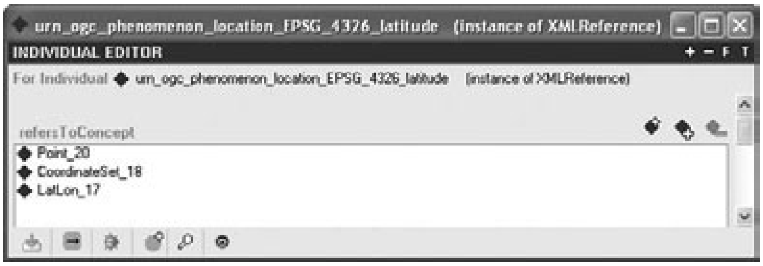
Kora uses the knowledge base of registered emergency information to infer new concept relations:

- She searches for specific information that has been registered to concepts in the knowledge base with a concept match type II, for example:

  $$R(T) \sqsubseteq (\exists\ \mathit{containsXMLReference}.(\exists\ \mathit{refersToConcept}.\mathit{riskmap{:}WindSpeed}))$$

- She searches the web for other sensors with similar information to the ones already registered, based on their semantic annotation.

- She searches for a service that simulates the progression of the gas plume, based on the sensor information. This service is then chained with traffic

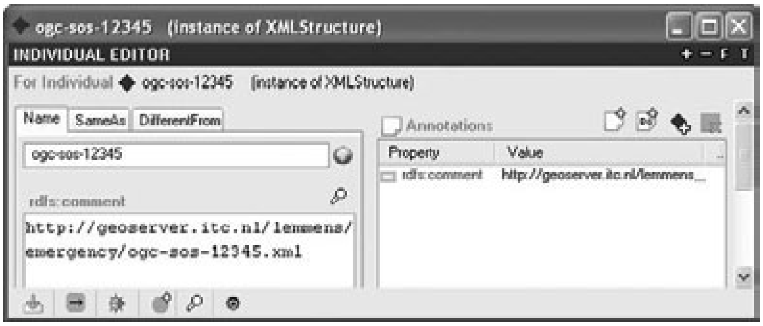


urn_ogc_phenomenon_location_EPSG_4326_latitude (instance of XMLReference)

For Individual urn_ogc_phenomenon_location_EPSG_4326_latitude (instance of XMLReference)

Point_20
CoordinateSet_18
LatLon_17

Figure 7.5: Annotation of sensor observations in the Riskmap ontology — detail 1. The figure shows the ontology concepts to which the XML reference 'latitude' refers. The postfix numbers are id-numbers, generated by the Protégé ontology editor.



ogc-sos-12345 (instance of XMLStructure)

For Individual ogc-sos-12345 (instance of XMLStructure)

ogc-sos-12345

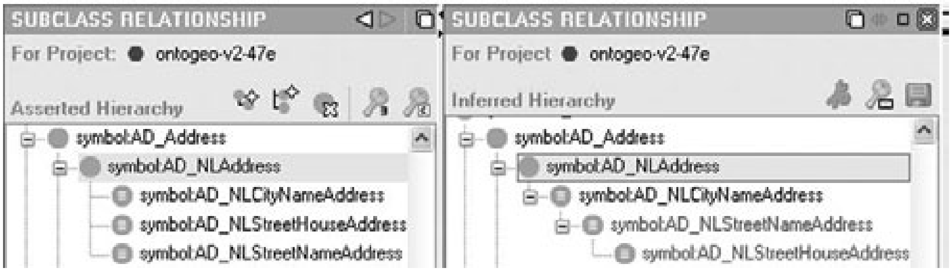
rdfs:comment
```
http://geoserver.itc.nl/lemmens/
emergency/ogc-sos-12345.xml
```

Property	Value
rdfs:comment	http://geoserver.itc.nl/lemmens_

Figure 7.6: Annotation of sensor observations in the Riskmap ontology — detail 2. The figure shows the reference to the XML structure that defines the sensor content information. It is stored in the ontology as a URL in a *rdf:comment* field.

services that provide detour information to in-car navigation systems. The service discovery and composition is done through the method, described in the use case in Section 7.1.1.

- For the dispatch of aid workers, the accuracy of locations is an essential property. Kora uses the reasoner to classify all registered emergency information in the knowledge base, based on the accuracy of their address labels. For example, some addresses contain street name and house number, others contain only street name. The classification is done through the classification of concepts, which is depicted in Figure 7.7. The address elements for the four most specific address types are indicated in Table 7.1. The address type at the bottom is the most specific. The reasoning applied above can be considered a kind of 'spatial reasoning'.



```
SUBCLASS RELATIONSHIP                      SUBCLASS RELATIONSHIP
For Project: ● ontogeo-v2-47e              For Project: ● ontogeo-v2-47e
Asserted Hierarchy                         Inferred Hierarchy
symbol:AD_Address                          symbol:AD_Address
  symbol:AD_NLAddress                        symbol:AD_NLAddress
    symbol:AD_NLCityNameAddress                symbol:AD_NLCityNameAddress
    symbol:AD_NLStreetHouseAddress               symbol:AD_NLStreetNameAddress
    symbol:AD_NLStreetNameAddress                  symbol:AD_NLStreetHouseAddress
```

Figure 7.7: Classification of address classes, performed with the RacerPro reasoner classification procedure. Left: asserted hierarchy (before classification). Right: inferred hierarchy (result of classification).

Address concept	Elements
symbol:AD_NLCityNameAddress	City name
symbol:AD_NLStreetNameAddress	City name, street name
symbol:AD_NLStreetHouseAddress	City name, street name, house number

Table 7.1: Address elements for three types of Dutch addresses. The *AD* bi-alpha prefixed concepts refer to prescribed address classes, specified in ISO 19133 (LBSNav) [138].

Because all incoming information is registered with a time stamp, a full play-back of knowledge registration is possible when an aftermath analysis is performed. This considerably contributes to the establishment of lineage for this application.

7.3 Research Net

This section describes the implementation of the use case as a continuation of Section 2.6.3. The use case involves a researcher named Jody who creates a service chain for the purpose of satellite image processing and a researcher named Jeff who reuses this service chain.

7.3.1 Phase 1: Service provision

In this first phase, Jody has the role of service developer, as depicted in Figures 2.2 and 3.9. For creating the services, she uses three SPOT images, representing respectively the periods during the dry season, a moderately severe flood and a severe flood (see Figures 7.8 and 7.9). In this phase, it is assumed that the processing services have been identified and that they are directly available. As Jody wants to describe her method and wants to make it available to others,

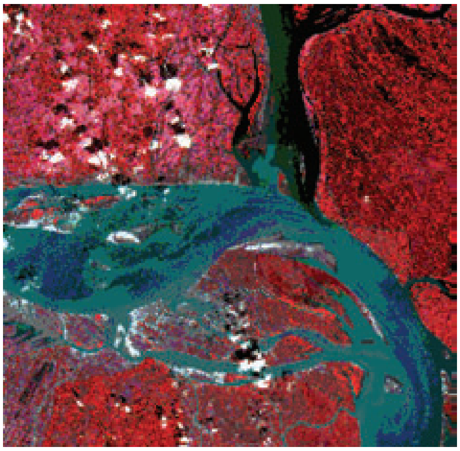


Figure 7.8: SPOT color composite of moderately severe flood, taken from [175]

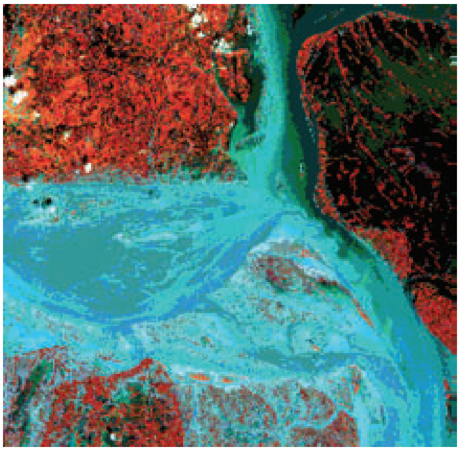


Figure 7.9: SPOT color composite of severe flood, taken from [175]

she creates meta-information for each component service and for the composite service (a sequence). As each service contains only one operation, this is done by referencing each service to a class in the OPERA ontology. The result is an OWL-S document that describes the service chain semantically. The subsequent services are described below:

1. Jody uses a *band rationing service* to distinguish land-water boundaries. Dividing SPOT band 3 by band 1 is a known method that accomplishes this. This service can be classified as *opera:CrossCalculate* in the OPERA ontology.

2. A *slicing service* is used to classify the water and land pixels in each SPOT image. The result for the moderately severe flooding period is shown in Figure 7.10. This service is a subclass of OPERA's *opera:Classify*.

3. With a *cross service*, the three land-water grid coverages are now combined into one, by creating a class for each combination of land and water attributes (see Figure 7.11). The cross service is classified as *opera:CrossCalculate* in OPERA.

4. Finally, Jody uses a service called *ConvertToClasses* to show the impact areas of two flooding periods. She selects the class with attribute *landdry*-watermfl*waterfl* to represent a moderately severe flooding period and the class with attribute *landdry*landmfl*waterfl* to represent a severe flooding period (respectively the second and third class from the top of the legend in Figure 7.11). The service is a subclass of *opera:Group*

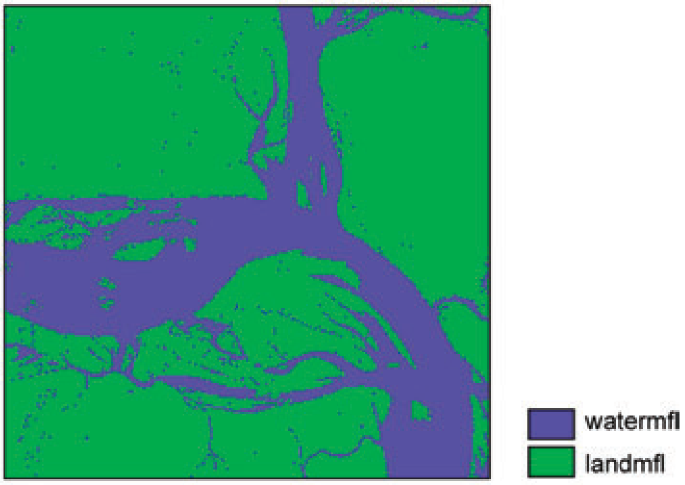


Figure 7.10: Result of the *slicing* service for the moderately severe flooding period, taken from [175].

With an annotation tool (in this case the OWL-S editor, described in Chapter 7), she imports the OPERA ontology and enters the service chain definition below:

$$EvaluateFlood \equiv C$$

in which C is an ontological concept that describes the control flow pattern below:

```
<sequence>
      BandRationing : opera:CrossCalculate
      Slicing : opera:Classify
      Cross : opera:CrossConcatenate
      ConvertToClasses : opera:Group
</sequence>
```

Depending on the publication option (see the three options in Section 2.6.3), Jody creates service groundings as well. So far, Jody has applied the principle of *transparent* service chaining, as described in Section 3.3.2. Now, she is going to make the flood analysis available to another user in different modes, i.e., allowing a user to (1) see a single map, (2) perform opaque chaining and (3) perform translucent chaining. These three options are implemented as follows.

Option 1: An OGC Web Map Service (WMS) showing the end result.

- Discovery: The meta-information, describing this WMS, is materialised as an OWL document with the following items:

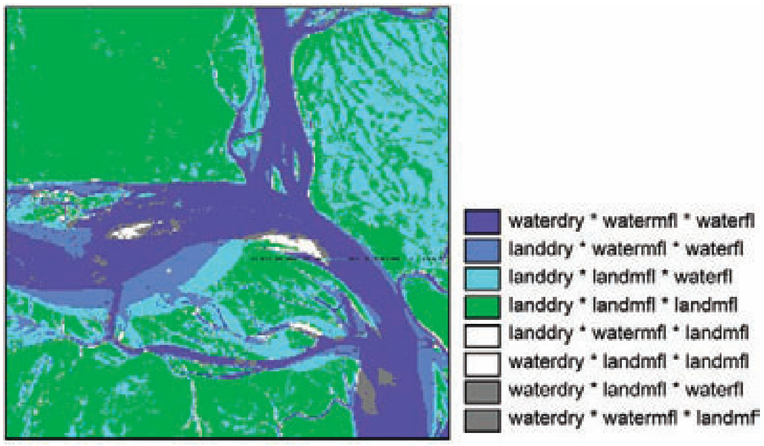

waterdry * watermfl * waterfl
landdry * watermfl * waterfl
landdry * landmfl * waterfl
landdry * landmfl * landmfl
landdry * watermfl * landmfl
waterdry * landmfl * landmfl
waterdry * landmfl * waterfl
waterdry * watermfl * landmf

Figure 7.11: Result of the *cross* service. An attribute class is formed for each combination of land and water attributes of the three periods (dry, moderate flood, severe flood). Figure taken from [175].

  - End result meta-information (OWL constructs with references to OPERA, SYMBOL and concept ontologies)
  - Lineage meta-information (OWL-S control flow constructs)

- Execution: Link to the WMS in the form of a URL (GetMap request).

Option 2: A link to the satellite images, implemented by an OGC Web Coverage Service (WCS) [208] and a link to the (composite) *EvaluateFlood* service.

- Discovery: The meta-information, describing these services, is materialised with an OWL document for each service with references to OPERA, SYMBOL and concept ontologies. In addition, the *EvaluateFlood* service is described by OWL-S control flow constructs.

- Execution:

  - A link to the WCS in the form of a URL (GetCoverage request)
  - A link to the *EvaluateFlood* service in the form of a URL, that invokes the service chain. This mode is referred to as *opaque* service chaining, see Section 3.3.2.

Option 3: A link to the satellite images and meta-information to create a service chain.

- Discovery: The meta-information is the same as in option 2.

- Execution:
  - A link to the WCS in the form of a URL (GetCoverage request)
  - A link to a WSBPEL document, containing the control flow of the service chain and WSDL descriptions for each of the component services in the chain. This mode allows for *translucent* chaining, see Section 3.3.2.

7.3.2 Phase 2: Service consumption

Jeff is a user that would like to use a flood evaluation service. In this second phase of the use case scenario, he has the role of application user, as depicted in Figures 2.2 and 3.9. To find the composite service, Jeff first formulates the following query that holds a key characterisation of its functionality:

$$\begin{aligned} R \equiv \quad & \mathit{opera{:}CrossCalculate} \sqcap \\ & (\exists\ \mathit{opera{:}hasInputPar.}(\exists\ \mathit{opera{:}hasParType.} \\ & \quad (\exists\ \mathit{opera{:}typeBijection.opera{:}OP\_ISO19115\_MDBand}))) \\ & (\exists\ \mathit{opera{:}hasOutputPar.}(\exists\ \mathit{opera{:}hasParType.} \\ & \quad (\exists\ \mathit{opera{:}typeBijection.opera{:}OP\_ISO19115\_MDBand}))) \end{aligned} \tag{7.4}$$

Jeff is able to discover this service, independent of the option that Jody has chosen. After all, the OWL-S document with chain information is made available in each option. By investigating the meta-information in more detail (following the *single service discovery* procedure, described in Section 6.4.5), Jeff discovers that the band rationing service is part of a service chain, called *EvaluateFlood* (see Figure 7.12). Depending on Jody's publication option, Jeff is now able to view her end result (option 1) or to perform the chaining himself with other data (options 2 and 3).

7.4 Travel Google

This section describes the implementation of the use case, as discussed in Section 2.6.4. The use case makes the following assumptions:

- A considerable number of geo-information resources (data sets and services) have been described, based on one or more ontologies, and these descriptions have been published in a registry.
- Ontologies that contain travel concepts, similar to the ones described in Section 5.3.4 are available on the web.
- Ontologies are discoverable through their registered meta-information. A Semantic Web search engine facilitates their discovery and enables the inspection of their ontological structure and capabilities.

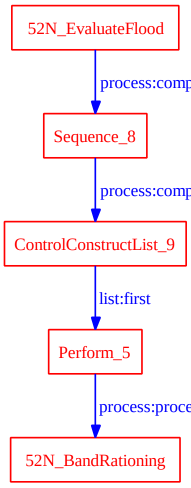


Figure 7.12: Part of the OWL-S service chain that shows the band rationing service as first part of the composite flood evaluation service.

The key person in this use case is Eddie who has a meeting in the United Nations headquarters in New York. He would like to stay in a hotel in southern Manhattan and wants to travel with public transport between his hotel and UN as preparation for his trip. He wants to know the restaurants in the neighbourhood of his hotel and wants to have a printed map with this information for walking around downtown.

Eddie first searches for a suitable ontology that facilitates the description of geo-information and services in the travel domain. He uses a Semantic Web search engine (see Section 4.6.4) to find ontologies and service descriptions containing the concepts *hotel*, *restaurant* and *travel*. In a first round he selects four ontologies :

- TravelOntology[6]
- OnToGeo ontology[7]
- SUMO[8]
- Wordnet[9]

which he is going to evaluate on their suitability for the task at hand.

[6]http://learn.tsinghua.edu.cn/homepage/2003214945/travelontology.owl
[7]http://geoserver.itc.nl/lemmens/owl/ontogeo.owl
[8]http://reliant.teknowledge.com/DAML/Mid-level-ontology.owl
[9]http://xmlns.com/wordnet/1.6/

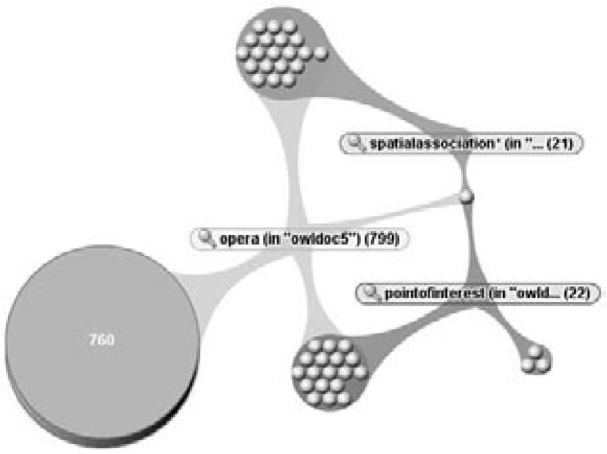

spatialassociation* (in "... (21)
opera (in "owldoc5") (799)
pointofinterest (in "owld... (22)
760

Figure 7.13: Cluster map, generated by the AutoFocus software, that shows the relationships between three search terms (rounded boxes) that appear in the OnToGeo ontology. The spheres in each cluster represent the ontology concepts and properties that match one or more of the search terms. A number indicates the number of items found.

7.4.1 Ontology inspection

During a further inspection of the ontologies with the AutoFocus tool see Chapter 7), Eddie finds out that the SUMO and Wordnet ontologies are generic ontologies that do not provide relationships between the hotel and restaurant concepts in a travel domain. The Travel ontology turns out to be designed for describing travel reservations in the travel agency domain and is too limited to describe potential geo-resources. He selects the OnToGeo ontology, because it provides constructs to describe features and their spatial relationships in travel information and services. This means that the ontology facilitates resource descriptions, semantically rich enough to hold meaningful advertisements to a more detailed service request. The cluster map, generated by the AutoFocus software[10] in Figure 7.13, shows the essential relationships between three key terms in the OnToGeo ontology, that clarify the above mentioned ontology property. In the figure, each search term is indicated by a rounded box, ontology concepts and properties are represented by spheres. These spheres are clustered in subsets that match the specific search term. In this particular ontology representation, a term match means that an OnToGeo concept or property contains the (textual) term in its definition. The figure shows that in the OnToGeo ontology there are:

- Concepts that contain operation elements *and* spatial association elements.

[10]AutoFocus is discussed in Section 8.2.9

- Concepts that contain operation elements *and* point-of-interest elements.

- One concept that contains operation elements *and* spatial association elements *and* point-of-interest elements. However, this concept is a probe class for testing purposes that does not actually belong to the core model of the ontology.

Ruling out this last concept, even the first two items give an indication of the potential capability of the ontology.

7.4.2 Creation of service request

Using the OnToGeo ontology, Eddie creates a service request that seeks for services, showing the spatial relationships between hotels, restaurants and public transport hubs. The query is translated into a request (type II) of the form:

$$
\begin{aligned}
R \equiv \\
&((\exists\ \mathit{opera{:}hasInputPar}. \\
&\qquad (\exists\ \mathit{opera{:}hasParType.symbol{:}SpatialAssociationType})) \sqcup \\
&(\exists\ \mathit{opera{:}hasOutputPar}. \\
&\qquad (\exists\ \mathit{opera{:}hasParType.symbol{:}SpatialAssociationType}))) \sqcap \\
&(\exists\ \mathit{opera{:}hasInputPar}.(\exists\ \mathit{opera{:}hasParType}. \\
&\qquad (\exists\ \mathit{opera{:}typeBijection.opera{:}OP\_Travel\_Hotel}))) \sqcap \\
&(\exists\ \mathit{opera{:}hasInputPar}.(\exists\ \mathit{opera{:}hasParType}. \\
&\qquad (\exists\ \mathit{opera{:}typeBijection.opera{:}OP\_Travel\_Restaurant}))) \sqcap \\
&(\exists\ \mathit{opera{:}hasInputPar}.(\exists\ \mathit{opera{:}hasParType}. \\
&\qquad (\exists\ \mathit{opera{:}typeBijection.opera{:}OP\_Travel\_TransportHub})))
\end{aligned}
\tag{7.5}
$$

Request 7.5 makes use of the following assumption: Operations that either use or create a spatial association role have the capability of spatially relating geographic features. Examples of such operations are those in the categories *opera:DistanceBased*, *opera:Topology*, etc. (see Appendix D).

To relax the request 7.5, a more general request is formulated as well, to 'catch' services that may not have categorised their points of interest. The last three axioms of Request 7.5 are then interchanged with:

$$
\begin{aligned}
&(\exists\ \mathit{opera{:}hasInputPar}.(\exists\ \mathit{opera{:}hasParType}. \\
&\qquad (\exists\ \mathit{opera{:}typeBijection.opera{:}OP\_Travel\_PointOfInterest})))
\end{aligned}
\tag{7.6}
$$

7.4.3 Exploitation of ontology mappings

To be more specific, the request is refined by adding a partial request for a route planner. Eddie is familiar with the OGC OpenLS specification and expresses his request accordingly. This is translated by the query client into the following axiom:

$$R \sqsubseteq \textit{ogc:OpenLSRouteService} \tag{7.7}$$

Relaxing the request Assuming that the advertised services are all defined in the ISO 19119 ontology (Services), the mapping between the OpenLS and 19119 ontology enables the discovery of services, 'close' to Eddie's request. Note that it requires to relax the request first to the superclass of *ogc:OpenLSRouteService*, otherwise it will not target *all* advertisements in the ontology. Figure 7.14 shows the mappings that serve this purpose. The 'unrelaxed' request 7.7 will only find *ESRIArcWebRoute*. Once we relax the request to the superclass *iso19119:Route-DeterminationService* as follows

$$R \sqsubseteq \textit{iso19119:RouteDeterminationService} \tag{7.8}$$

the reasoner will also find the other advertisements shown in Figure 7.14. This also includes the advertisement a that is defined in yet another ontology O which is mapped to the ISO 19119 (Services) ontology. The advertising instances are elaborated in the next paragraph.

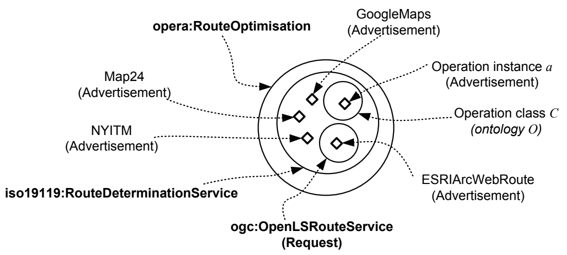

opera:RouteOptimisation
GoogleMaps (Advertisement)
Map24 (Advertisement)
Operation instance a (Advertisement)
Operation class C (ontology O)
NYITM (Advertisement)
ESRIArcWebRoute (Advertisement)
iso19119:RouteDeterminationService
ogc:OpenLSRouteService (Request)

Figure 7.14: Venn diagram representing the ontology mapping context, used in the discovery of a route service. The figure shows the containment of advertisements for the ontology classes of OPERA, ISO 19119 and OGC. The detailed mappings between these classes are described in Section 6.3.2.

Another use of ontology mapping had in fact been used in request 7.5. The role *opera:hasInput.symbol:SpatialAssociationRole* is defined in OPERA-R for a subset of its operations. Due to the mapping between ISO 19119 (Services) and OPERA-R, a subset of ISO 19119 operation concepts inherit this role (see Figure 7.15). As a consequence, the role can be used in matchmaking between concepts in either ontology.

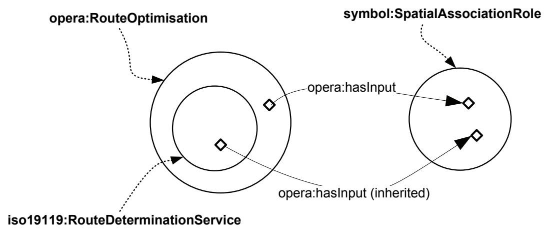

opera:RouteOptimisation
symbol:SpatialAssociationRole
opera:hasInput
opera:hasInput (inherited)
iso19119:RouteDeterminationService

Figure 7.15: Venn diagram representing the ontology mapping context, used in the discovery of a service that has the capability of spatially relating geographic features. The example in the figure shows that the class *RouteDeterminationService*, defined in the ISO 19119 taxonomy, inherits this characteristic from its mapped superclass in the OPERA ontology.

7.4.4 Service description analysis and service selection

With the request 7.8 in the previous section, Eddie discovers four services:

1. GoogleLocal[11] (including the GoogleMaps functionality)
2. ESRI ArcWeb Services-Routing (ESRIArcWebRoute) finder[12]
3. Map24[13]
4. New York Interactive Transit Map (NYITM)[14]

Eddie derives from the service descriptions that the ArcWebRoute service complies to the OpenLS standard. However, it is available as an API (with most of the requested options) and as a limited demo application. As Eddie wants quick results, the API is not suitable. The demo has no point-of-interest input option and suffices neither. The GoogleLocal and Map24 services are quite similar in functionality. However, the GoogleLocal service description shows two options that Eddie likes: (1) map output can be overlaid with satellite images and (2) point coordinates are accepted as input to display on the map.

From the advertisements, Eddie finds out that service 1 and 3 show subway stations, but do not provide route planning on the subway network (they provide *street* route planning only). Service 4 provides subway route planning, but only shows a map background, no other points of interest, such as the hotel of departure. Service 4 cannot be chained with 1 or 3 through data streaming. However, from the same advertisement it is clear that they all contain a map background, so they

[11]http://local.google.com/
[12]http://mapapps.esri.com/aws_routing/index.cfm
[13]http://www.us.map24.com/
[14]http://www.brail.org/transit/nycgoogle.html

can be compared and 'chained' visually. Eddie would like to compare all three services and performs his task with as follows:

1. In the 'point-of-interest' option of Map24 he selects the category 'Lodging' and city 'New York'. He selects the Cosmopolitan hotel (see Figure 7.16).
2. He uses Map24's 'proximity search' option to locate surrounding restaurants.
3. In Google Local he zooms in on Southern Manhattan and searches for 'hotel'. He selects the Cosmopolitan hotel (see Figure 7.17).
4. He searches for surrounding restaurants with the key words 'cosmopolitan hotel restaurants'.
5. He uses Google Local to locate the United Nations head quarters.
6. He uses the NYITM application to find the right subway line, see Figures 7.18 and 7.19. The application requires the user to enter start location and destination by mouse-clicking on the map. Eddie estimates his locations by looking at the locations resulting form the previous applications. In Figure 7.18, the start location is indicated by a triangle-in-balloon, the destination by a square-in-balloon. The in-between balloons indicate subway stations. The service also provides textual information with subway line identifiers (not displayed in the figures).

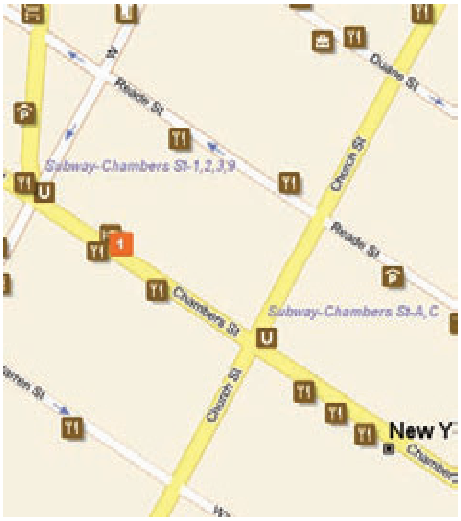
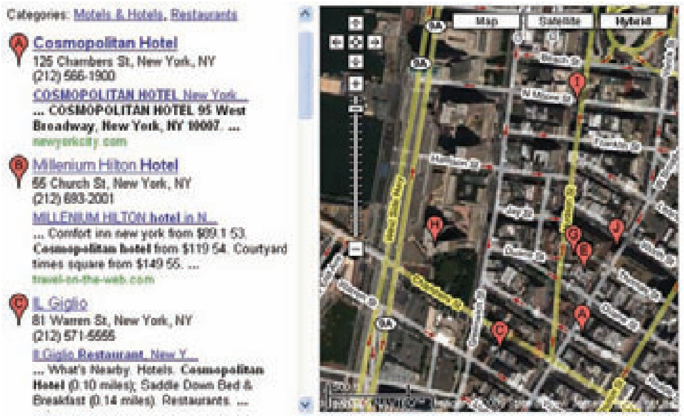


Figure 7.16: Map features of the Map24 application. Box 1 is the selected hotel.


Categories: Motels & Hotels, Restaurants

A Cosmopolitan Hotel
125 Chambers St, New York, NY
(212) 566-1900
COSMOPOLITAN HOTEL New York ...
... COSMOPOLITAN HOTEL 95 West Broadway, New York, NY 10007. ...
newyorkcity.com

B Millenium Hilton Hotel
55 Church St, New York, NY
(212) 693-2001
MILLENIUM HILTON hotel in N...
... Comfort inn new york from $89.1 53. Cosmopolitan hotel from $119 54. Courtyard times square from $149 55. ...
travel-on-the-web.com

C IL Giglio
81 Warren St, New York, NY
(212) 571-5555
Il Giglio Restaurant, New Y...
... What's Nearby. Hotels. Cosmopolitan Hotel (0.10 miles); Saddle Down Bed & Breakfast (0.14 miles). Restaurants. ...

Figure 7.17: Map features of the Google maps application. The Cosmopolitan is the selected hotel.

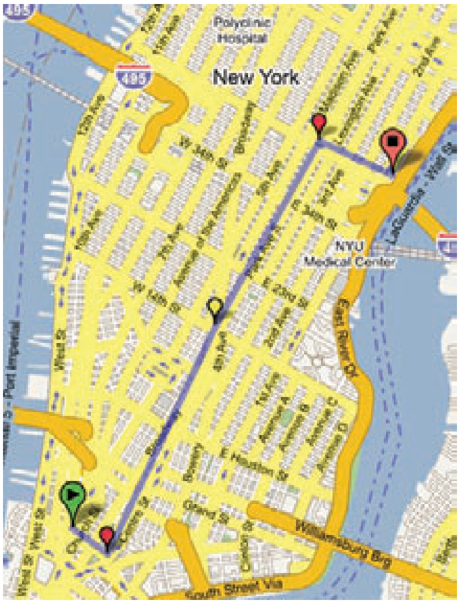


Figure 7.18: Subway route between hotel and conference centre, calculated by the NYITM application.

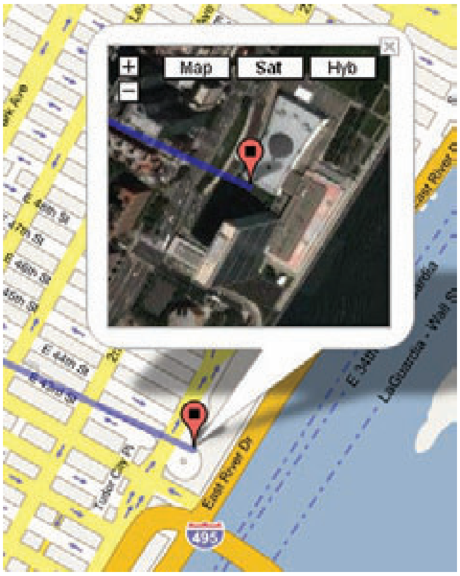


Figure 7.19: Detail of the NYITM application, that borrows functionality from Google Maps. The insert shows the United Nations head quarters.

7.5 Summary and reflection

This chapter has provided relevant examples for the typical use of the semantic interoperability framework as presented in Chapter 5 and the reasoning described in Chapter 6. The examples have demonstrated the following:

- Based on the tests described in Section 7.1.2 and the discussion in Section 6.4 (section introduction) we observe that machine ontologies can be seen as useful extensions to XML schema (on which for example TOP10NL and NEN3610 are based) when it comes to concept integration. Ontologies can also be used as data repositories, but, for simple data carrier purposes XML documents and schema will suffice (see also a related discussion in [256]).

- Semantic service descriptions can successfully assist a human user in discovering a geo-service in isolation or as part of a service chain. This discovery is based on the services' type (as part of a taxonomy), their input/output parameters, their validity region/application domain (imposed by tightly-coupled geodata), and their internal workflow. A limitation in the discovery process is formed by the fact that it still significantly depends on the human-in-the-loop.

- The applied reasoning is based on the constructs, provided by OWL-DL. No additional spatial hierarchies have been created to facilitate spatial reasoning, except for the structuring of address types as locations. This implies that the *spatial* matching of feature instances has not been possible. As this research assumes that instances of real-world objects (e.g., *Louvre* and *Paris*, see the research scope in Section 1.2) are not part of the ontology, this does not pose a problem. However, for the case of matching service validity regions —imposed by tightly-coupled geodata— spatial reasoning would be useful. The inability to do so forms a limitation in the presented framework and prototype.

- In a combined research effort which has been described in Section 7.1.1 it is shown[15] that the results of the GeoMatchMaker prototype in the form of abstract compositions can be successfully used as input to the process of concrete composition and execution. The 'glue' between abstract and concrete composition is provided by semantic annotation.

- Ontology mappings are crucial for the discovery and interoperability of services across different application domains. The creation of these mappings is not trivial and needs expert knowledge. Semi-automatic methods are required to assist domain experts in creating consistent and complete mappings between large ontologies.

[15]The actual results are described in [165].

Chapter 8

Implementation of prototypes: OnToGeo and GeoMatchMaker

The interoperability framework described in Chapter 5 has been implemented in an OWL-ontology that is from here referred to as the *OnToGeo* prototype ontology. OnToGeo integrates the namespaced concepts of the SYMBOL, feature concept and OPERA ontologies.

OnToGeo is created with a selection of currently available software tools, which is referred to as *the workbench*. These tools are also used for ontology analysis and reasoning. Some of the software tools were adapted and integrated for building the *GeoMatchMaker* prototype, which was developed in this thesis work for matchmaking purposes. The combination of the OnToGeo prototype, the workbench tools and the GeoMatchMaker prototype are from here referred to as *the prototype environment*.

Figure 8.1 shows a general overview of the components of the prototype environment. The dashed boxes indicate the major usage of the components:

- Semantic framework creation, updating and inspection; introduced in Chapter 5.

- Geo-information matching and service chaining (discovery and abstract composition); introduced in Chapter 6.

Both usages were also elaborated upon in the use case implementation of Chapter 7.

Conceptual design considerations of OnToGeo have been discussed in Chapter 5. The current chapter discusses practical design and implementation considerations (Section 8.1). Central in the software part of the prototype environment is

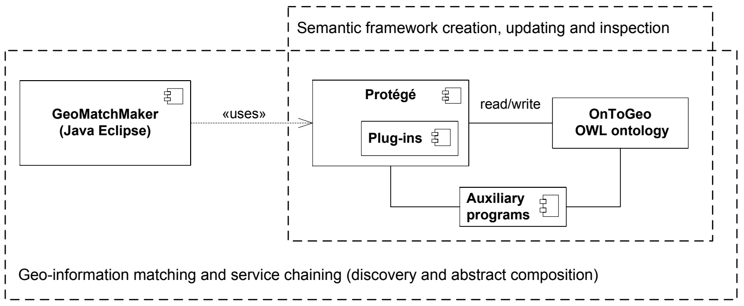

Semantic framework creation, updating and inspection
GeoMatchMaker (Java Eclipse)
«uses»
Protégé
Plug-ins
read/write
OnToGeo OWL ontology
Auxiliary programs
Geo-information matching and service chaining (discovery and abstract composition)

Figure 8.1: UML component diagram showing an overview of the main prototype components.

the Protégé ontology platform. It is used with several of its plug-in software modules and auxiliary programs. An example of the latter is the RacerPro reasoner. An overview of the software tools is given in Section 8.2; the GeoMatchMaker prototype is discussed in Section 8.3. Procedures followed for the creation of service descriptions in the prototype environment are discussed in Section 8.4.

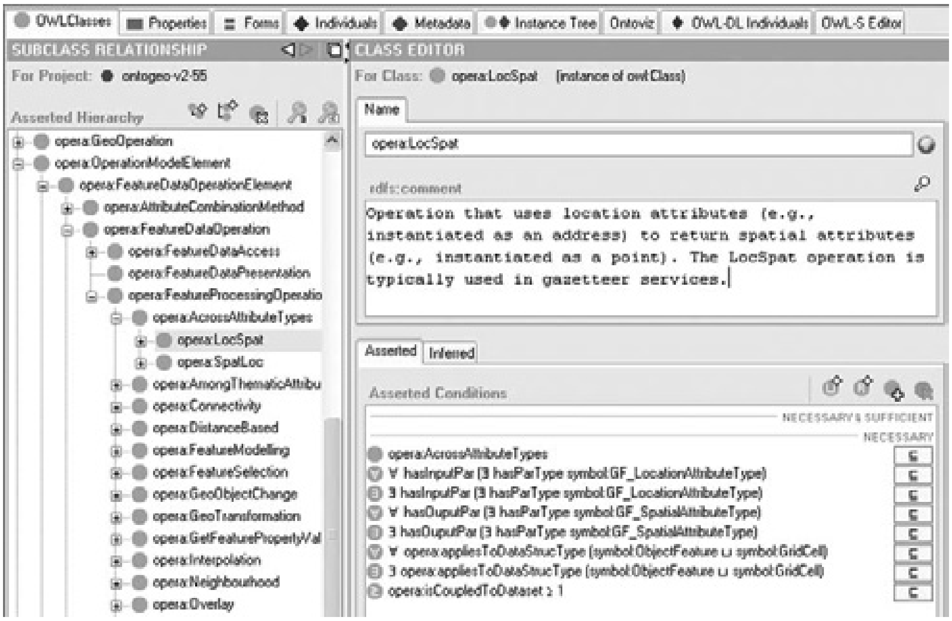
OWL-S OWL-S is considered to be an important asset to the OnToGeo and GeoMatchMaker prototypes. However, the prototypes have been designed in such a way that they can also function independently of OWL-S and their ties with OWL-S are clearly visible. These ties are discussed in Sections 8.1 and 8.3.

8.1 Practical design and implementation issues of OnToGeo

The OnToGeo ontology has been designed according to the *Manchester House Style* [237] (see Section 4.3). Operations are implemented as primitive classes (see the remarks in Section 5.5.3). Figure 8.2 shows part of the asserted hierarchy of OPERA with the asserted conditions for its LocSpat operation, which is the implementation of Description 5.4 in Section 5.5.3.

8.1.1 Implementation of OWL-S

OWL-S has been integrated with OnToGeo through the linking of operation identifiers and operation input/output parameters, as described below.



For Project: ontogeo-v2-55

For Class: opera:LocSpat (instance of owl:Class)

```
opera:GeoOperation
opera:OperationModelElement
  opera:FeatureDataOperationElement
    opera:AttributeCombinationMethod
    opera:FeatureDataOperation
      opera:FeatureDataAccess
      opera:FeatureDataPresentation
      opera:FeatureProcessingOperatio
        opera:AcrossAttributeTypes
          opera:LocSpat
          opera:SpatLoc
        opera:AmongThematicAttribu
        opera:Connectivity
        opera:DistanceBased
        opera:FeatureModelling
        opera:FeatureSelection
        opera:GeoObjectChange
        opera:GeoTransformation
        opera:GetFeaturePropertyVal
        opera:Interpolation
        opera:Neighbourhood
        opera:Overlay
```

opera:LocSpat

```
Operation that uses location attributes (e.g.,
instantiated as an address) to return spatial attributes
(e.g., instantiated as a point). The LocSpat operation is
typically used in gazetteer services.
```

```
opera:AcrossAttributeTypes                                                    ⊑
∀ hasInputPar (∃ hasParType symbol:GF_LocationAttributeType)                  ⊑
∃ hasInputPar (∃ hasParType symbol:GF_LocationAttributeType)                  ⊑
∀ hasOuputPar (∃ hasParType symbol:GF_SpatialAttributeType)                   ⊑
∃ hasOuputPar (∃ hasParType symbol:GF_SpatialAttributeType)                   ⊑
∀ opera:appliesToDataStrucType (symbol:ObjectFeature ⊔ symbol:GridCell)       ⊑
∃ opera:appliesToDataStrucType (symbol:ObjectFeature ⊔ symbol:GridCell)       ⊑
opera:isCoupledToDataset ≥ 1                                                  ⊑
```

Figure 8.2: Screenshot of the Protégé ontology editor, showing the asserted class hierarchy of OPERA (left window), the asserted conditions of the LocSpat operation (bottom-right window) and a short explanation of its functionality (top-right window).

Operation identifiers

Operation identifiers have been constructed both in *opera:GeoOperation* (in OPERA) and *process:Process* (in OWL-S). If the operation is modelled as a class, than it is modelled as subclass of both

- a named OPERA operation type from Appendix C or a derived operation type.
- a subclass of the OWL-S *process:Process* class: either *process:AtomicProcess*, *process:CompositeProcess* or *process:SimpleProcess*.

If the operation is modelled as an individual, than this individual has two class types as listed above, at the same time. Dual class types are created in the OWL-DL individuals tab by the *Add named type* option.

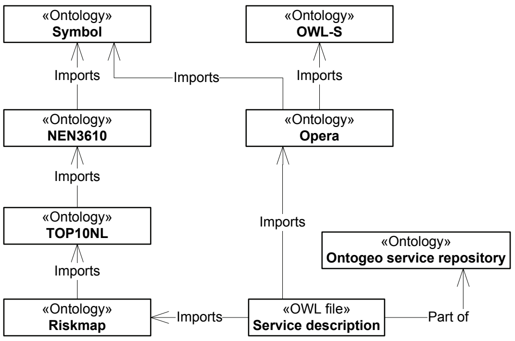


Figure 8.3: Import pattern of separately-stored ontologies in the prototype. The example shows how Riskmap service descriptions make use the ontologies. Service descriptions of another type than Riskmap may use another import pattern, e.g., by importing NEN3610 directly and bypassing TOP10NL.

Operation input/output parameters

Another link between OPERA and OWL-S is established through the dual class typing of their input and output parameters. An individual that represents an input parameter instantiates both *opera:InputSymbol* in OPERA and *process:Input* in OWL-S. This is done similarly for output parameters. In this way other OWL-S constructs, such as the grounding, can use the individuals, created in OPERA.

8.1.2 Distributed ontologies and namespaces

There are two basic ways to store and manage the ontologies used in the prototype. First, they can be stored in a single OWL file, as is currently done. Preferably, the ontologies have different namespaces, so that their concepts remain unique. Once this file is loaded into a knowledge base (e.g., Protégé), the concepts of all ontologies are fully editable. The advantage of this approach is that changes in each of the concepts is performed in the knowledge base and relationships with other concepts are updated instantly. Second, the ontologies can be stored separately. In an environment where the ontologies are shared among different applications, this separated storage allows for ontology maintenance at the source and clearly shows their dependencies. Each application that wants to use an ontology has to import it. Often, imported ontologies import ontologies themselves, so they are imported in chains (see Figure 8.3). This may imply that some ontologies are imported more than once (e.g., the symbol ontology in the figure). In practice this should not be a problem, as long as these ontologies are identical.

To avoid some of the problems involved with versioning, the Protégé ontology environment does not allow changes to the concepts of imported ontologies nor to the relations that they already participate in. Actions that *are* allowed with imported concepts are the *addition* of subclasses, superclasses, role creation, etc.

Storing multiple ontologies in a single OWL file is useful when the ontologies are developed integrally and subject to frequent change. Once they are more stable, and their independency is an issue, they may be stored separately. Ontology separation is still a tedious job and is not well supported by current ontology tools. The inverse process, ontology conjunction, is easier, for example by editing/merging the OWL files directly in the XML code.

Either way, the use of prefixes for ontology 'domains' is considered a good practice. In OnToGeo, all derived service classes get the default prefix *opera*, unless they belong to a specific domain, such as *openls* or *ogc*. Individuals normally do not get a prefix.

8.2 Workbench tools

This section discusses the tools that have been used as part of the prototype environment. Sections 8.2.1 through 8.2.3 describe the Protégé ontology editor and its plug-ins. Sections 8.2.5 through 8.2.8 comprise tools involved in the reasoning process. The tool described in Section 8.2.9 is used for ontology exploration.

8.2.1 Protégé core and OWL plug-in

Protégé is an open source ontology editor that supports OWL-based ontology development and inferencing. Protégé is java-based and is extensible via plug-ins [149]. Around seventy plug-ins have been developed and published for a wide variety of purposes, e.g., ontology export formats, visualisation, reasoning, etc. Some of them have been included in the standard installation procedure of Protégé. In fact, OWL support itself is provided through a plug-in, see Figure 8.4; Protégé has its own internal representation mechanism for ontologies and knowledge bases, based on a metamodel, which is comparable to object-oriented and frame-based systems [149]. In the prototype environment discussed in this thesis, Protégé version 3.1 Build 195 and Build 207 were used, with respectively OWL plug-ins version 2.1 beta Build 275 and version 2.1 rc Build 283. Protégé and its *OWL*, *Ontoviz* and *OWL-S editor* plug-ins have been extensively used in this thesis project. The most relevant details of their functionality is discussed in the current section. Examples of their usage are highlighted in Chapter 7 and in the remainder of the current chapter. For further documentation on Protégé, the reader is referred to [118, 149] and the reference materials at the CO-ODE website[1].

[1]http://www.co-ode.org/resources/reference/

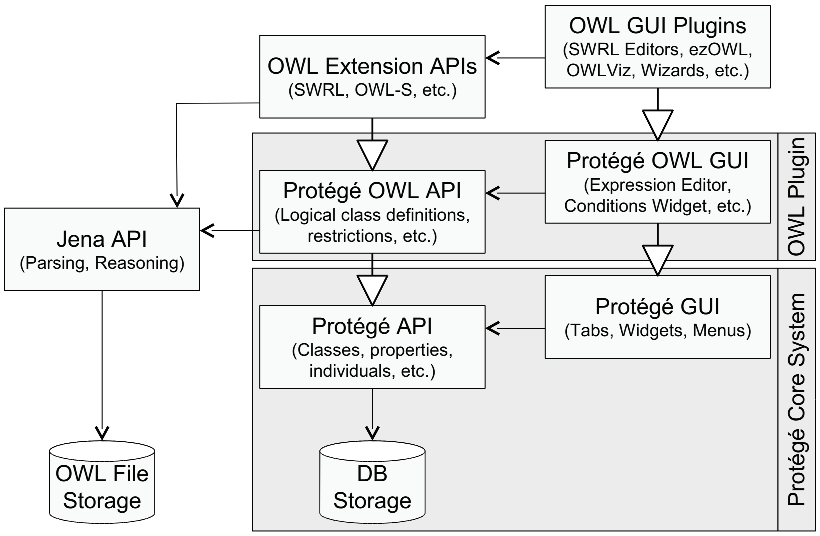


Figure 8.4: The extension mechanism of Protégé. Figure taken from [149].

8.2.2 Ontoviz plug-in

Ontoviz [252] is one of the Protégé ontology visualisation plug-ins. It has been used extensively in this project due to its possibility to visualise classes together with individuals and properties (DL-roles) and the flexibility of class and individual selection. This makes it a powerful tool for ontology inspection and documentation. The figures that have been produced with OntoViz in this thesis are identified as 'ontology capture diagrams'. An example of such a diagram is Figure 4.4.

Ontoviz has been developed by Michael Sintek at the German Research Center for Artificial Intelligence (DFKI).

8.2.3 OWL-S editor plug-in

The OWL-S editor is designed for easy and intuitive OWL-S service development [67]. The control flow of services, described in the OWL-S process model, can have a rather complex structure. The OWL-S editor provides a graphic editor to easily construct control flows. In addition, the plug-in provides an interface to create OWL-S groundings and import WSDL files. In the prototype environment, the OWL-S editor is considered to be a powerful tool for abstract service composition. It was this functionality and the wish to create an integrated discovery-composition environment that led to the integration of the OWL-S editor and the GeoMatch-Maker prototype. The details of this integration are discussed in Section 8.3.

8.2.4 OWLDoc plug-in

OWLDoc is a Protégé plug-in, distributed by the CO-ODE project[2], that creates Java Doc[3] style documentation for OWL ontologies. OWLDoc creates for each ontology element (class, individual and role) a small HTML file with the characteristics of that element. For example, for each class it shows its local class hierarchy. All elements are hyperlinked to the ones they have an ontological relationship with. This facilitates the exploration of the ontology through a web browser as a stand-alone client, which is very useful for sharing its meta-information over the web. Figure 8.5 shows an example output of OWLDoc.

8.2.5 Reasoning through DIG interface

Protégé can be connected with a reasoner through the DIG interface, which is a standardised XML interface to Description Logics systems developed by the DL Implementation Group (DIG) [21]. Protégé has several commands it can send to through the DIG interface. This functionality is here referred to as the Protégé *DIG reasoning commander*. It entails the following functions for checking the entire ontology (function group 1):

- Check consistency (‘`?`▷’ menu option): checks the entire ontology for unsatisfiable concepts. In the Protégé class window, all unsatisfiable concepts are marked red.

- Classify taxonomy (‘`C`▷’ menu option): checks the entire ontology for unsatisfiable concepts and implicit subsumption relationships between concept names. In the Protégé class window, all unsatisfiable concepts are marked red and two additional windows are opened with respectively the inferred class hierarchy and an overview of concepts, moved by the reasoner.

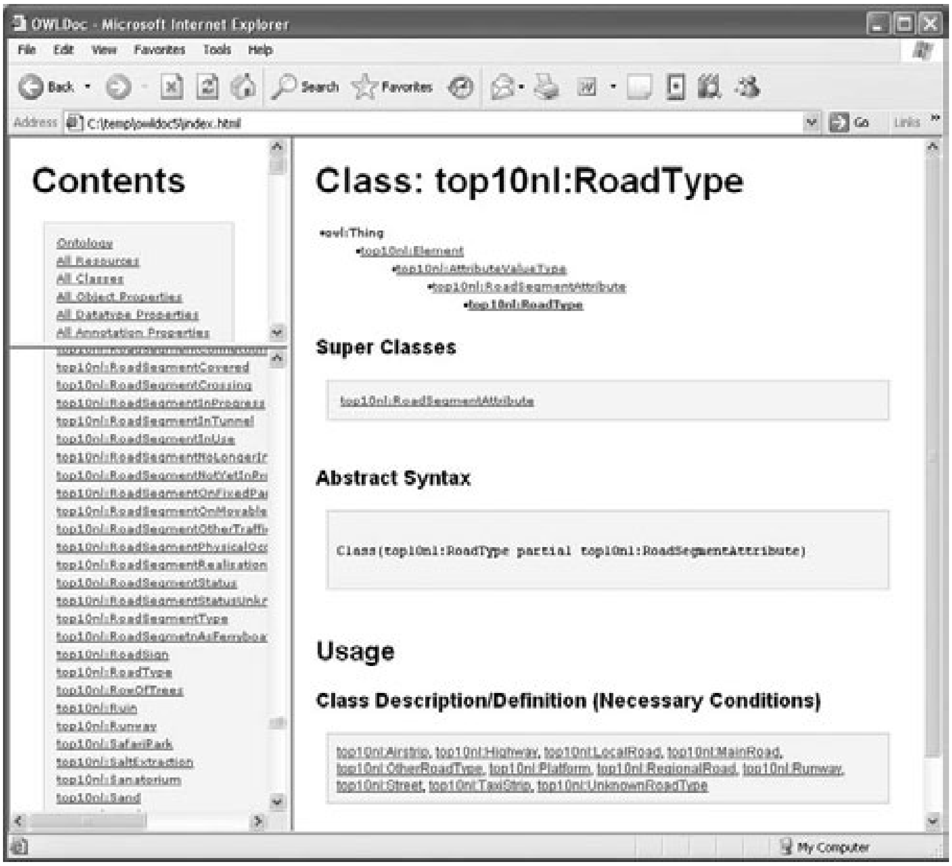
- Compute inferred types (‘`I`▷’ menu option): Computes the inferred types (classes) for the individuals in the ontology. The results are displayed in the OWL-DL individuals tab, distributed by the CO-ODE project[4]

It provides functions for each concept (function group 2):

- Check concept consistency (‘`?`▷’ menu option): see similar function in function group 1. The results are displayed in a temporary result window.

- Compute individuals belonging to class (‘`I`▷’ menu option). Note that this function is not the same as the one with the same icon in function group 1. The results are displayed in a temporary result window.

[2]http://www.co-ode.org/downloads/
[3]http://java.sun.com/j2se/javadoc/
[4]http://www.co-ode.org/downloads/


OWLDoc - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\temp\owldoc5\index.html
Go Links
My Computer

Contents

Ontology
All Resources
All Classes
All Object Properties
All Datatype Properties
All Annotation Properties
top10nl:RoadSegmentCovered
top10nl:RoadSegmentCrossing
top10nl:RoadSegmentInProgress
top10nl:RoadSegmentInTunnel
top10nl:RoadSegmentInUse
top10nl:RoadSegmentNoLongerIr
top10nl:RoadSegmentNotYetInPr
top10nl:RoadSegmentOnFixedPa
top10nl:RoadSegmentOnMovable
top10nl:RoadSegmentOtherTraffi
top10nl:RoadSegmentPhysicalOcc
top10nl:RoadSegmentRealisation
top10nl:RoadSegmentStatus
top10nl:RoadSegmentStatusUnkr
top10nl:RoadSegmentType
top10nl:RoadSegmetnAsFerryboa
top10nl:RoadSign
top10nl:RoadType
top10nl:RowOfTrees
top10nl:Ruin
top10nl:Runway
top10nl:SafariPark
top10nl:SaltExtraction
top10nl:Sanatorium
top10nl:Sand

Class: top10nl:RoadType

- owl:Thing
  - top10nl:Element
    - top10nl:AttributeValueType
      - top10nl:RoadSegmentAttribute
        - top10nl:RoadType

Super Classes

top10nl:RoadSegmentAttribute

Abstract Syntax

```
Class(top10nl:RoadType partial top10nl:RoadSegmentAttribute)
```

Usage

Class Description/Definition (Necessary Conditions)

top10nl:Airstrip, top10nl:Highway, top10nl:LocalRoad, top10nl:MainRoad, top10nl:OtherRoadType, top10nl:Platform, top10nl:RegionalRoad, top10nl:Runway, top10nl:Street, top10nl:TaxiStrip, top10nl:UnknownRoadType

Figure 8.5: Output of OWLDoc for the TOP10NL ontology (see Section 5.3.2). Exploration of the ontology in a web browser is facilitated through hyperlinked ontology elements.

- Get inferred superclasses ('$\sqsubseteq \triangleright$' menu option): computes the classes that subsume this class. The results are displayed in a temporary result window.

and a function for each individual in the ontology (function group 3):

- Compute types ('T $\triangleright$' menu option): computes the inferred types (classes) for the individuals in the ontology (similar to the I $\triangleright$ menu option in function group 1). The results are displayed in a temporary result window.

8.2.6 RacerPro

RacerPro [5] is a knowledge representation system that can be used for reasoning with ontologies. RacerPro and alternative reasoners have been introduced in Section 4.5.1, under 'Tools'.

RacerPro implements the description logic $\mathcal{ALCQHI}_{R^+}$, also known as $\mathcal{SHIQ}$ [233]. RacerPro can directly read OWL-Lite and OWL-DL documents and represent them as TBoxes and ABoxes in DL knowledge bases [108, 287]. The only restriction for OWL-DL is that RacerPro does not support nominals (individual names expressed in class descriptions). RacerPro provides numerous functions for managing the knowledge base and reasoning with its TBoxes and ABoxes. These functions are classified as follows [232]:

- Knowledge base management functions: initialisation of TBoxes and ABoxes, OWL document import, etc.
- Knowledge base declarations: creation of concepts, individuals, roles, disjoint concepts, etc.
- Reasoning modes: functions for changing the reasoning mode, for example, to set unique name assumption[6].
- Evaluation functions
  - Queries for concept terms: checking of class subsumption, disjointness, etc.
  - Role queries: checking of role subsumption, transitivity, domain, range, etc.
  - TBox evaluation functions: classify TBox, check coherency, etc.
  - ABox evaluation functions: checking of Abox consistency, etc.
  - ABox queries: Checking role relations between individuals, checking instantiation relationship between concept and individual, etc.

[5] RacerPro stands for *Renamed ABox and Concept Expression Reasoner Professional*

[6] By default RacerPro assumes no unique name for each individual, which means that two named individuals could be the same or different, unless explicitly stated by the model.

- Retrieval
  - TBox retrieval of: taxonomy, all sub/super classes of a concept, all sub/super roles of a role, etc.
  - ABox retrieval of: all concepts of which an individual is an instance, all individuals that are instances of a concept, role assertions in which an individual participates, etc.

All these functions can be called through a LISP interface but RacerPro also acts as a server, providing these functions through a TCP interface and an HTTP-based standard DIG interface (see Section 8.2.5) for connecting client programs.

Further, Racerpro provides a recently developed API for expressing specific queries on the knowledge base. This API is called nRQL, not to confuse with an RDF query language. nRQL can be seen as an expressive ABox query language that implements a subset of the OWL-QL query language[287]. nRQL's functionality shows an overlap with that of the reasoning functions listed above. For example, the RacerPro function *(concept-instances $R(T)$)* (see Section 6.4.4) returns the same result as the nRQL query (retrieve (?x) (?x $R(T)$)). nRQL has not been implemented in the prototype. Although not investigated, it is expected that inclusion of nRQL functionality in the prototype will increase its expressiveness. The version of RacerPro used in this thesis project is 1.8.1 2005-06-29.

8.2.7 JRacer

JRacer is a Java software library that provides Java methods for creating client applications to the RacerPro server TCP socket interface [107]. Each JRacer method represents a RacerPro function as discussed in 8.2.6. The RacerPro function requests and responses are sent as booleans or strings through the socket interface. In this thesis project, the JRacer API version 2 has been used to implement reasoning functionality in the OWL-S editor plug-in of Protégé (see Section 8.3). This version of JRacer is the version, originally developed by Jordi Alvarez[7].

8.2.8 RICE

RICE (RACER Interactive Client Environment) [54] is a software program that provides a graphical user interface to RacerPro. It lets a user to invoke all RacerPro functions by typing or pasting them in a request window and displays RacerPro's reply in a separate window. It makes use of the JRacer API. Although RICE is not capable of programming multiple RacerPro requests and was not designed for integration in other software, it is very suitable for testing single concept requests. It is used accordingly in the prototype environment (Version 1.2 Build 36: April 11, 2004). RICE has been developed by Ronald Cornet at the department of Medical Informatics, Academic Medical Center, University of Amsterdam.

[7]A newer version makes now part of the RacerPro software distribution and is named JRacer 1.8.

8.2.9 AutoFocus

AutoFocus [2] is a software tool for the exploration and search of information stored in electronic documents . A search is performed by entering search terms similar to common web search engines. The software produces a list of relevant documents and for each one it lists the most significant terms found, which can be used for a refinement of the search. In addition it creates a *cluster map*, that shows graphical clusters of documents that contain the search term entered. When more than one search term is entered, the documents are also clustered for each combination of terms. In this so called *guided exploration*, a user can interpret the significance of multiple terms in the document repository. AutoFocus supports several file formats, such as MS-Word and HTML, both on-line and locally stored. In the prototype environment, AutoFocus has been deployed together with OWLDoc (see Section 8.2.4) to explore 'foreign' ontologies. OWLDoc has been used to generate HTML files for each class and property of such ontology. Then AutoFocus was used to earmark particular ontology elements and visualise their relationships in order to get an overview of the *scope*[8] of the ontology. This method is also proposed for the inspection of the OnToGeo ontology by third parties, as demonstrated in the Travel Google use case in Section 7.4. Figure 7.13 shows an example of the cluster map generated in this use case.

8.3 GeoMatchMaker, an integrated prototype

The GeoMatchMaker prototype has been developed to perform ontology editing and service chaining in one, integrated software application. It has been developed in the Eclipse Java developing environment. Eclipse[9] is an open source software development platform, which is extensible through a plug-in mechanism [123]. It is particularly renown for its excellent Java development tools. The development of GeoMatchMaker entailed the modification of the OWL-S editor of SRI International by providing it with a connection to the RacerPro reasoner and a simple user interface to interact with the reasoner. The prototype application is built in Eclipse from modified source code of the OWL-S editor and Java jar files of the rest of the Protégé software. By compiling the source files in this way, GeoMatchMaker behaves like Protégé, but with additional functionality.

8.3.1 Functional parts

Figure 8.6 shows the components of the prototype. The functional parts of Protégé are indicated with boxes in the Protégé component box. Protégé uses an OWL parser to import and export OWL documents. Communication with the RacerPro reasoner is done in two ways:

[8]The *scope* of an ontology has been defined in Section 4.1
[9]http://www.eclipse.org/

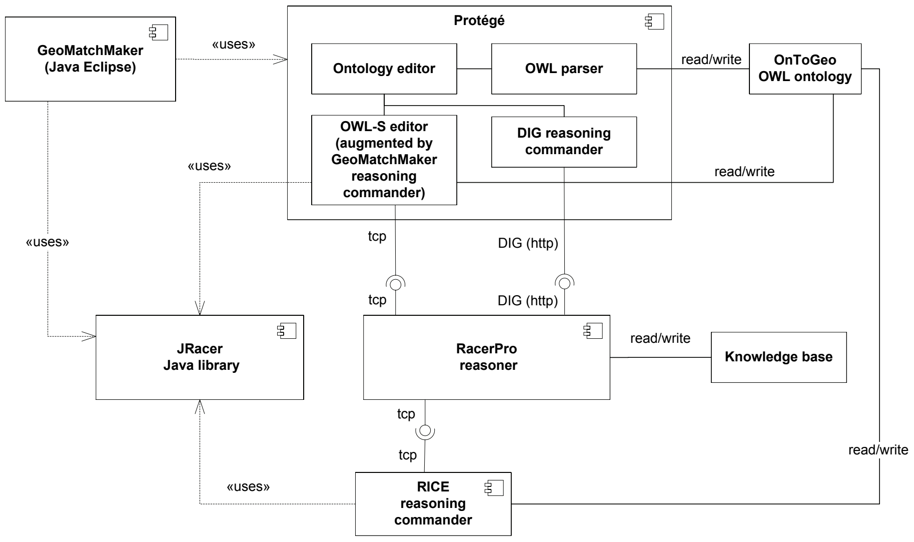


Figure 8.6: UML component diagram showing the GeoMatchMaker prototype with implemented software components.

1. Via the HTTP-based DIG interface.
2. Via TCP-based JRacer calls in the augmented OWL-S editor.

The actual reasoning is performed on a knowledge base, that is temporarily created by RacerPro and filled by Protégé's reasoning commander (see Section 8.2). The RICE software is used for checking purposes and manual testing of the reasoner.

Figure 8.7 shows the relationships between the augmented OWL-S editor and JRacer at java class and package level. The methods *Check* and *SearchSimilar* of the java class *CheckWithRacer* are invoked by buttons in the OWL-S editor GUI. They form placeholders for the implementation of algorithms that call JRacer Methods (that for their part call RacerPro functions). For all the experiments reported in this thesis, the method *conceptInstances* is used in type II ABox queries. The option of a relaxed query (see its discussion in Section 8.3.3) has been implemented with the *conceptParents* method. Figure 8.8 shows the Eclipse workbench used for the implementation and executions of the java classes. The left-hand window of the workbench contains the source packages and *Java Archive (jar) files* files used in the application. The top-middle window contains part of the source code of the *CheckWithRacer.java* class. The bottom-middle window shows a test run with the OnToGeo ontology and the right-hand side window shows the methods and properties of the *CheckWithRacer.java* class.

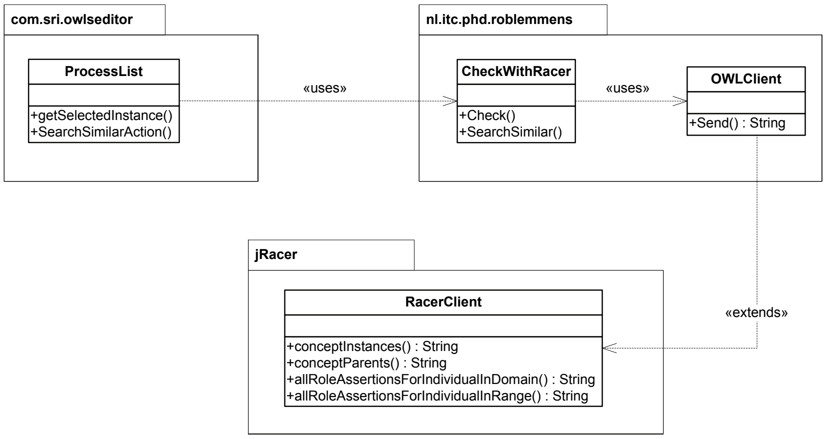
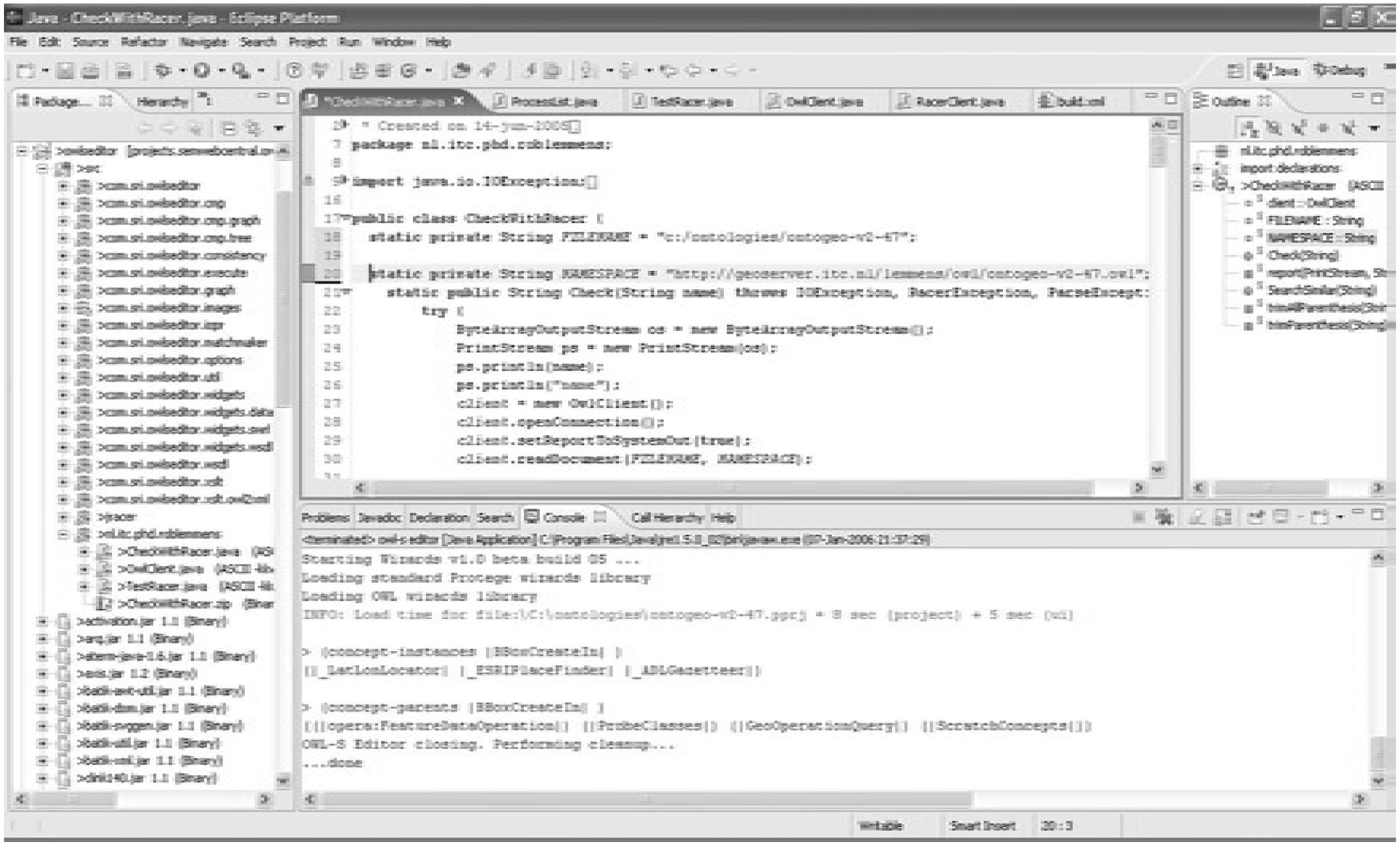


Figure 8.7: UML class diagram showing the essential class and package relations that augment the graphic user interface of the OWL-S editor by customisable reasoning capabilities. Only the most important Java classes and methods are displayed in the diagram.


Java - CheckWithRacer.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Package... Hierarchy
*CheckWithRacer.java ProcessList.java TestRacer.java OwlClient.java RacerClient.java build.xml
Outline
Problems Javadoc Declaration Search Console Call Hierarchy Help
Writable Smart Insert 20 : 3

```
 * Created on 14-jun-2005
package nl.itc.phd.roblemmens;

import java.io.IOException;

public class CheckWithRacer {
  static private String FILENAME = "c:/ontologies/ontogeo-v2-47";

  static private String NAMESPACE = "http://geoserver.itc.nl/lemmens/owl/ontogeo-v2-47.owl";
    static public String Check(String name) throws IOException, RacerException, ParseExcept:
        try {
            ByteArrayOutputStream os = new ByteArrayOutputStream();
            PrintStream ps = new PrintStream(os);
            ps.println(name);
            ps.println("name");
            client = new OwlClient();
            client.openConnection();
            client.setReportToSystemOut(true);
            client.readDocument(FILENAME, NAMESPACE);
```

```
<terminated> owl-s editor [Java Application] C:\Program Files\Java\jre1.5.0_02\bin\javaw.exe (07-Jan-2006 21:37:29)
Starting Wizards v1.0 beta build 05 ...
Loading standard Protege wizards library
Loading OWL wizards library
INFO: Load time for file:\C:\ontologies\ontogeo-v2-47.pprj = 8 sec (project) + 5 sec (ui)

> (concept-instances |BBoxCreateIn| )
(|i_LatLonLocator| |_ESRIPlaceFinder| |_ADLGazetteer|)

> (concept-parents |BBoxCreateIn| )
((|opera:FeatureDataOperation|) (|ProbeClasses|) (|GeoOperationQuery|) (|ScratchConcepts|))
OWL-S Editor closing. Performing cleanup...
....done
```

Figure 8.8: Eclipse workbench showing part of the GeoMatchMaker implementation.

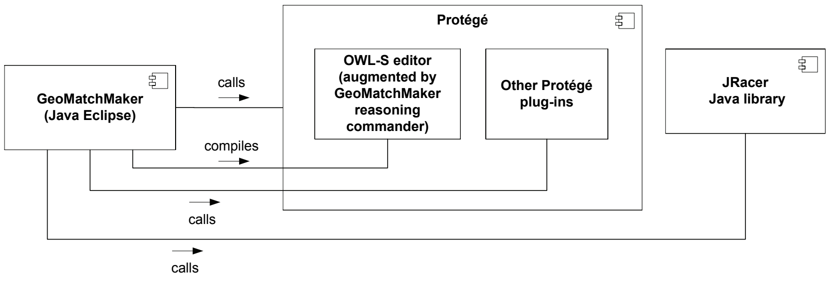

Protégé
GeoMatchMaker (Java Eclipse)
OWL-S editor (augmented by GeoMatchMaker reasoning commander)
Other Protégé plug-ins
JRacer Java library
calls
compiles
calls
calls

Figure 8.9: UML communication diagram showing the compilation process of the GeoMatchMaker prototype software. It includes in the build-path: (1) the adapted *source code* of the OWL-S editor and (2) the *Java Archive (jar) files* of the Protégé software and its other plug-ins.

8.3.2 Compilation of Java code

An executable of GeoMatchMaker is built with a combination of source files and *Java Archive (jar) files*, as indicated in Figure 8.9. The executable is a java jar file (*Protege.jar*) that is used in the experiments discussed in Chapter 7.

8.3.3 Usage

The workflow of a typical experiment is presented in Figure 8.10. Note that the components in the figure are the same as in Figure 8.6. After reading the OWL document *ontogeo.owl* (step 1), a semantic query is formulated as a requesting concept R in step 2. This involves two substeps:

1. Creating a probe class with appropriate conditions
2. Creating an individual with the same name, as instance of the *process:Query* class. The latter is a class that is added to the OWL-S service model as depicted below:

```
service:ServiceModel
    process:Process
        process:AtomicProcess
        process:CompositeProcess
        process:SimpleProcess
        process:Query
```

In step 3, the ontology is loaded in the RacerPro knowledge base by invoking any of the 'group 1' commands of the Protégé *DIG reasoning commander*. In step 4, the query is run by selecting the query instance of step 2.(see the red-boxed

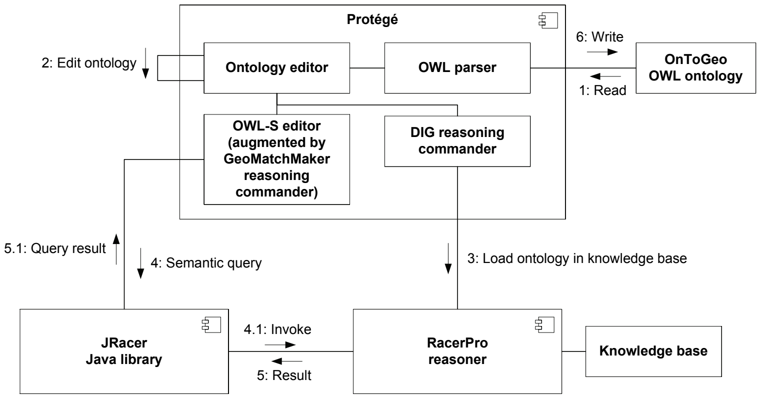
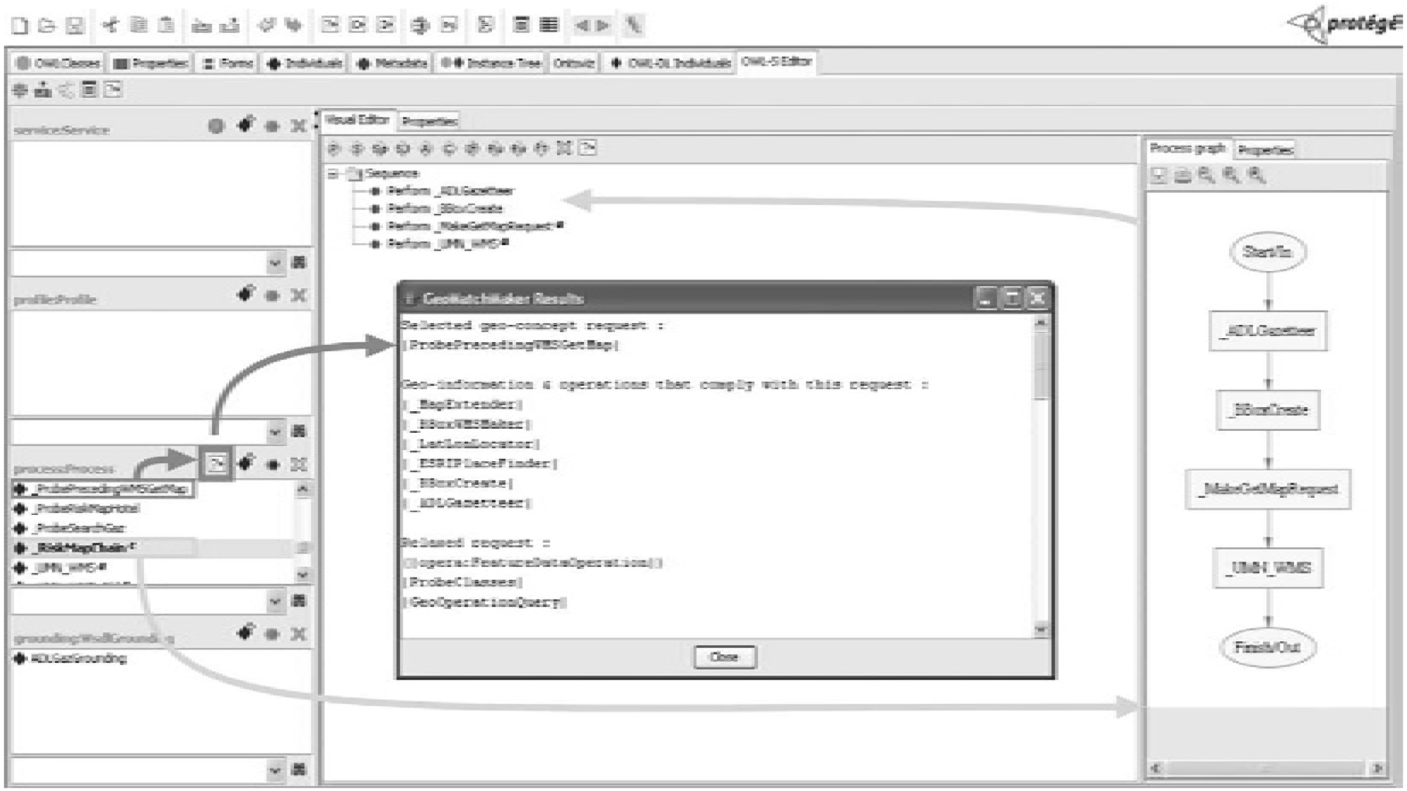


Figure 8.10: UML communication diagram showing the workflow of a semantic query in the prototype.

process name in Figure 8.11) and subsequently selecting the newly created **?** ▷ menu option in the OWL-S editor's process window (see the red-boxed button in Figure 8.11). The JRacer command call invokes the corresponding RacerPro function (step 4.1) and RacerPro's response is captured (step 5) as a string (containing one or more ontology elements, in this case individuals). In step 5.1, the query result is displayed in a separate window (the central window in Figure 8.11). The window shows the following items

- The selected request. In fact, this can be any request specified as a concept in the ontology, but considered the purpose of the prototype, it will involve in practice a request for geo-information or for a geo-operation.
- The ontology elements (in this case individuals) found as instances of the requesting concept.
- Suggestions for a relaxed request. This is a rudimentary implementation by providing the superclasses of the originally selected request, which can be used to specify a new request in step 4.

In step 6, the ontology (with additional probe class) may be stored again as OWL document. In addition, Figure 8.11 shows the rest of the OWL-S editor GUI. When selected, a composite service is shown by the editor as graph (right-side window). The menu bar of the visual editor's middle window contains diamond-shaped buttons that are used for the creation of the OWL-S control flow elements, such as sequence (S) and Repeat-until (Ru).



```
Selected geo-concept request :
|ProbePrecedingWMSGetMap|

Geo-information & operations that comply with this request :
|_MapExtender|
|_BBoxWMSMaker|
|_LatLonLocator|
|_ESRIPlaceFinder|
|_BBoxCreate|
|_ADLGazetteer|

Relaxed request :
|opera:FeatureDataOperation|
|ProbeClasses|
|GeoOperationQuery|
```

Figure 8.11: Screenshot of the OWL-S editor GUI, augmented by the GeoMatchMaker functionality. The thick dark grey arrows represent the activation of the request for services that can be chained with the GetMap request of the Minnesota Web Map Service, as presented in Section 7.1. The thick light grey arrows point to the representation of the Riskmap service chain in the native GUI of the OWL-S editor.

8.3.4 Exploitation and embedding

The OnToGeo and GeoMatchMaker prototypes have been embedded in an integrated approach, by combining them with another prototype, under development by Granell [97, 98]. This prototype, referred from here as *Integrated Component Designer*, supports concrete composition and execution of services. Figure 8.12 shows the integrated architecture of the combined prototypes as also described in [165]. It combines the user contexts of Figures 2.2 and 3.9 and implements the service chaining process as depicted in Figure 3.10. OnToGeo serves the interoperability between different processes as it is used for (1) WSDL annotation, (2) discovery and abstract composition and (3) concrete composition. Workflow documents form links between GeoMatchMaker, Integrated Component Designer and the workflow engine. New composite services that are created in the Integrated Component Designer, are fed into the registry, ready for use in newly defined tasks. The thick arrows in the figure indicate the workflow performed by an application user. In this case it shows the result of the workflow discussed in Section 6.1.

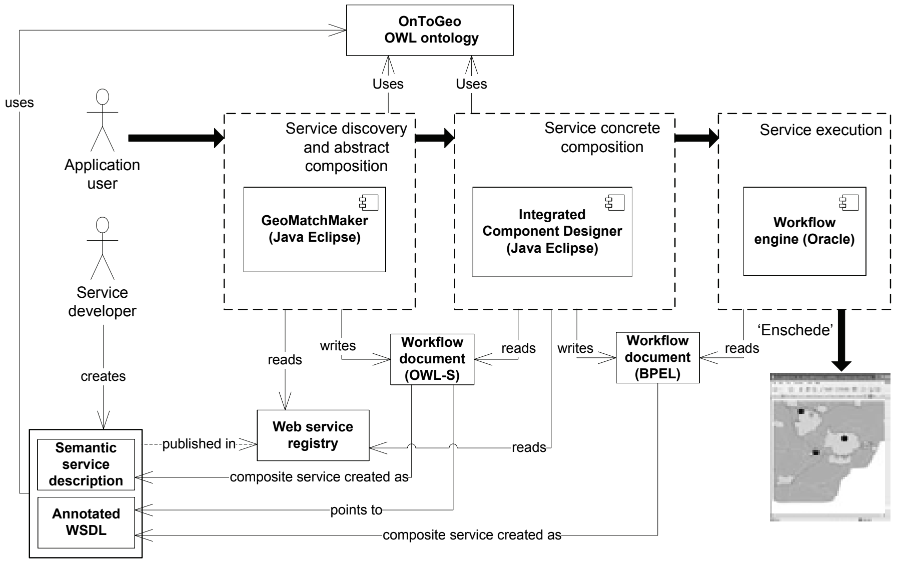
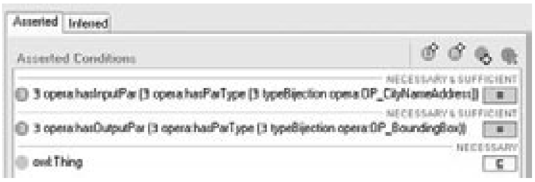

OnToGeo OWL ontology
uses
Uses
Uses
Application user
Service developer
creates
Service discovery and abstract composition
GeoMatchMaker (Java Eclipse)
Service concrete composition
Integrated Component Designer (Java Eclipse)
Service execution
Workflow engine (Oracle)
reads
writes
Workflow document (OWL-S)
reads
writes
Workflow document (BPEL)
reads
'Enschede'
Web service registry
Semantic service description
Annotated WSDL
published in
reads
composite service created as
points to
composite service created as

Figure 8.12: Integrated architecture for service chaining. Figure adapted from [165].

8.3.5 Limitations

The JRacer methods used in the prototype are limited to *conceptInstances* and *conceptParents*. Algorithms that implement these and other JRacer calls in a



NECESSARY & SUFFICIENT
∃ opera:hasInputPar (∃ opera:hasParType (∃ typeBijection opera:OP_CityNameAddress))
NECESSARY & SUFFICIENT
∃ opera:hasOutputPar (∃ opera:hasParType (∃ typeBijection opera:OP_BoundingBox))
NECESSARY
owl:Thing

Figure 8.13: Implementation in Protégé of the concept request of Section 7.1 (Description 7.3).

sequence or other control flow are deemed very useful and their realisation is fairly straightforward in the prototype architecture that has been realised in this thesis work. Such algorithms may involve methods to solve more advanced queries and allow for 'ontology navigation'. However, they have not been implemented yet in the prototype.

8.4 Creating service descriptions

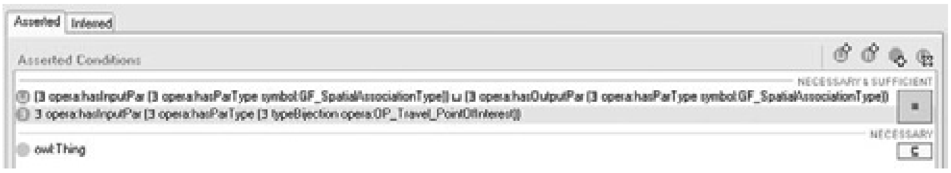
8.4.1 Individual-based descriptions

The creation of instances is done in the instance browser of Protégé. Protégé provides a mechanism that lets the user enter only an appropriate role assertion, constraint by the range of the role. It allows to enter 'nested' role assertions, by 'chaining' individuals. This nesting is effectuated by nested windows (one for each individual) in the user-interface.

8.4.2 Class-based descriptions

A concept that is intended as request for a concept match is created as subclass of (1) *ontogeo:GeoOperationQuery* if it models the parameters of a geo-operation or (2) *ontogeo:ProbeClass* in all other situations. Figure 8.13 shows the representation of a concept request in Protégé with two necessary & sufficient blocks. Individuals are returned that satisfy either of the two blocks. The first block shows the implementation of a nested concept condition. Figure 8.14 shows the use of a union construct in a concept request with one necessary & sufficient block.

In Protégé, individual-based descriptions are more easy to create than class-based descriptions, because the user is directed by forms, based on the corresponding class definitions. Obviously, it depends on the mode of matchmaking (type I, II, II or IV) which kind of description is appropriate.



(∃ opera:hasInputPar (∃ opera:hasParType symbol:GF_SpatialAssociationType)) ⊔ (∃ opera:hasOutputPar (∃ opera:hasParType symbol:GF_SpatialAssociationType))
∃ opera:hasInputPar (∃ opera:hasParType (∃ typeBijection opera:OP_Travel_PointOfInterest))
owl:Thing

Figure 8.14: implementation in Protégé of the concept request of Section 7.4 (Description 7.5 with 7.6).

8.5 Summary and reflection

This chapter has described the implementation aspects of the OnToGeo and GeoMatchMaker prototypes.

The Manchester House style (which is introduced in Section 4.3) provides useful guidelines for consistent ontology creation. However, two remarks are made below with respect to the ontology building process in this research:

1. Disjointness between concepts (which is elaborated upon in Section 5.5.3) was only used where needed. In some cases disjointness does not appropriately represent the intended semantics.

2. Domain and range restrictions (introduced in Section 4.2.1) were considered carefully, but did not pose a problem in the ontologies at hand.

The prototype environment has proved to be sufficient to demonstrate the use cases. However, the tools are not yet mature enough to be deployed instantly in practice. The following limitations of the work bench tools can be identified:

- The lack of support for extracting and copying parts of the ontology to other files (the Prompt plug-in did not provide satisfactory results).

- Protégé uses a Jena parser for importing and exporting OWL files. The order of OWL constructs is different, every time an OWL file is exported. In addition, modules of constructs, combining multiple role restrictions, may be found scattered around the OWL file. This does not pose a problem with Protégé's import, but it *does* with the import in RacerPro. This the reason for using the method of loading the ontology directly from Protégé in the knowledge base of RacerPro.

- The current prototype environment uses Protégé as an ontology editor, which has a user-friendly interface for entering individuals by means of forms. For class-based definitions, Protégé provides syntax checking, but the creation of nested role restrictions cannot considered to be user friendly. In addition, the creation and implications of multiple axioms in a class definition is not intuitive for a typical developer of geo-information sources that wants to enter meta-information. This definitely requires a more user-friendly menu-driven interface. Finally, Protégé is subject to improvement, which could help the development of the prototype development. This concerns amongst others, its OWL import/export method, ontology extraction/copy facilities and ontology visualisation.

The integrated architecture for service chaining as presented in Figure 8.12 (Section 8.3.4), is prototypical for a basic semantic infrastructure (see its definition in Section 4.6) and is proposed as such by this research.

Chapter 9

Conclusions and recommendations

We start this last chapter of the thesis by recapturing the overall objective of this research as stated in Section 1.2:

> *The general objective of this research is to provide solutions for the computer-aided integration of distributed heterogeneous geo-information and geo-services, based on their semantics, to support on demand geo-processing.*

The research puts an emphasis on the formal modelling of semantics, because the reasoning about the meaning of geo-information content and geo-service functionality is essential in geo-information integration and service interoperability. Yet, these semantic modelling techniques have not penetrated in current geo-application development and thus forms a challenge for future software implementations.

With this focus, the remainder of this chapter starts with a summary of the most important observations made in this research. It serves as a bridge between the summary and reflection sections in each chapter and the final conclusions in Section 9.2. The main contributions of this research are summarised in Section 9.3. Section 9.4 addresses key issues that play a role in the deployment of the proposed semantic interoperability framework. The chapter ends with recommendations for further work.

9.1 Summary and reflection

Solutions to overcome heterogeneity conflicts between geo-information sources have shifted from bi-lateral to multi-lateral contracting. Open interface specifications, endorsed by OGC, and a rigid geo-information model in the ISO 19100 series of standards, have contributed to the success of this shift (see Chapter 2).

Current issues that still hamper the interoperability of geo-services lie in the fact that contracts are often only loosely defined at the semantic level by referring informally to the concepts used. This so called semantic heterogeneity can be approached by formalising contracts and by making them machine accessible, so that software-aided processes can help to resolve heterogeneity conflicts. Making services semantically interoperable is an important prerequisite for information sharing in today's networked society.

This research has embarked upon such formalisation, by firstly analysing abstraction models for information and processes (see Chapter 3) and then translating existing submodels of ISO into machine ontologies (see Chapters 4 and 5). The representation was done using the Web Ontology Language (OWL). A satisfactory model still did not exist for modelling geo-*services*. This was developed, based on a set of common operations in GIS, reported in pertinent literature, software manuals and the ISO 19119 (Services) taxonomy. The ontologies were integrated in such a way that they form a *semantic interoperability framework* for expressing geo-information and geo-service descriptions.

The formalisation of a conceptual model can lead to many different OWL representations of the same model. This is due to the freedom we have in OWL (and Description Logics) to represent our concepts. Several literature resources provide design patterns as examples of 'good design practice' (see Section 4.3). The design of ontologies used in this thesis has been following the 'Manchester House Style' as closely as possible (see Section 8.1). The research has given explanations for the various design choices and alternatives.

The semantic interoperability framework, developed in this research, has been used for creating geo-information/service descriptions and to perform matchmaking between them (see Chapter 6). This has been done by applying existing Semantic Web techniques and adapting tools where necessary. Chapter 7 has described the use cases in which all of the above has been tested. The software tools that have been used have been described in Chapter 8.

A final reflection needs to be made on the limitations of the research. Ultimately, the task of a geo-information user is solved in a fully automated process that analyses the task, then matches it with several solutions, ranks them and finally presents them to the user (or provides it to a software agent that takes immediate action). Such a scenario requires adaptive task models and the simulation of combinatory modular solutions (read: services) and seems only implementable on the short/medium term in a closed problem space with not too many parameters. Most of the current approaches (including the method adopted by this research) assume the presence of a human in the loop, who is capable of taking care of parts of the reasoning process.

9.2 Conclusions

This section provides final conclusions, starting with the main and general conclusion, followed by answers to the research questions and conclusions on specific topics.

9.2.1 Main conclusion

The application of Semantic Web technology forms a promising approach for improving the interoperability of geo-services. This research has demonstrated that the geo-information models used (based on existing ISO models and a newly developed geo-operation model) lend themselves well to translation into formal, machine accessible, ontologies. The ontologies have proved to be mappable and suitable as a basis for creating semantically enriched meta-information for geo-information and geo-services in a *semantic interoperability framework*. Finally, basic matchmaking has been successfully applied in use case scenarios involving data set integration and service chaining. The limitations of the current prototype environment are formed by the constraints in (1) the user-interfaces, (2) the flexibility of the reasoning implementation and (3) the completeness of mappings between domain ontologies (e.g., 'Travel' and 'TOP10NL'), all of which are thought to be surmountable.

9.2.2 Answers to research questions

The research questions listed in Section 1.2 appear below with their answers.

1. *What are the requirements of on demand geoprocessing?*

   The background to answer this question has been provided in Chapter 2. In conclusion, on demand geo-processing requires:

   - Software and data to be modular and services (as a realisation of software and data) to be sufficiently generic to support different tasks. The 'size' of the implementation of a module depends on factors in the provider-consumer market, amongst which the demand for reusability of the service.
   - The availability of service meta-information based on commonly agreed rules for service characterisation. Loosely-coupled systems need to be self-descriptive.
   - Machine-supported reasoning about service meta-information.
   - Interoperability at all levels (syntactic, structural and semantic). The available service meta-information and reasoning should make it possible to identify if such interoperability requirement is met.
   - Services to be readily available for execution.

2. *What are the key problems in making geo-services interoperable and what solutions are currently available?*

   These issues have been addressed in chapters 2 through 4. The key problems are summarised below together with their solutions:

   - Heterogeneity. Section 2.2 has shown that the interoperability between services is hampered by heterogeneity issues at the syntactic, structural and semantic level. Heterogeneity can be overcome by contracts (system communication agreements) at each of these levels. Semantic heterogeneities may be apprehended by informal or by formal (machine accessible) contracts. The latter allow for resolving the heterogeneities by machine reasoning. Currently, the common approach to establish multi-lateral contracts is by using OGC specifications and ISO 19100 standards.
   - Informality of contracts. OGC specifications and ISO 19100 standards implement formal contracts on the syntactical and structural level, but they prescribe only informal contracting at the semantic level. They lack a formal, machine accessible model for the specification of semantics for geodata and geo-services. This problem may be overcome by the incorporation of an information model and process model (these issues have been addressed in Chapter 3) to which geodata and geo-services can refer their content. To break down the complexity of these models, it is important that they have a clear structure and incorporate a separation of concerns.
   - Incomplete realisation of contracts. Contracts are often only partly realised by the implementing services (e.g., WSDL files that are incomplete, services that are not fully OGC compliant). This is a rather grave issue that can only be resolved by providing the service with a wrapper, or if possible, to use only the compliant parts of the service.
   - Incomplete description of contracts. A clear distinction has to be made between the contracts themselves and the service meta-information that describes the contracts the service adheres to. Incompleteness of this meta-information reduces the possibility to assess the degree of interoperability between services in a service chaining context.

     Web-based ontologies can play an important role in the discovery and integration of geodata sets (e.g., data set harmonisation) and geo-services (e.g., service chaining) by overcoming the limitations of textual agreements, which are deployed in many of todays geo-information standards. The power of web-based ontologies lie in their interoperable (XML based) representation, the use of unique namespaces and the fact that they allow for automated reasoning (see Chapter 4). A typical niche for web-based ontologies is not in the area of syntactic data formats (which are sufficiently covered by many standards) but rather at the level of conceptual data models where current standards fall short. Web-based ontologies can provide a backbone for service descriptions that can be used for identifying (non-) interoperability issues as addressed in the key problems above.

- Incompleteness and incompatibility of semantic models (ontologies). Under the assumption that semantic models are being used for contracting and service descriptions, it may occur that these models are incomplete or not compatible with other models, used by candidate services. Ontology mappings can be used for the alignment between and the refinement of the concepts used in each semantic model (see Chapter 4). The required content of semantic models is addressed in the next research question.

3. *Up to what extent can and should semantics be modelled to support semantic interoperability (both for geo-information and functionality of geo-services)?*

   The demand for machine-supported reasoning requires the model of an application domain to be formal and machine readable. In Section 7.5 it was stated that machine ontologies can be seen as useful extensions to XML schema when it comes to concept integration. Ontologies can also be used as data repositories, but for simple data carrier purposes XML documents and schema will suffice.

   In the geo-information domain, introduced in Chapter 5, an ontological concept is typically constructed by the basic notion of a *feature type* and its properties (e.g., the geometric object that represents it). These constructs form the common ground (called the *semantic interoperability framework*) to which semantic descriptions of interoperable services should refer. These descriptions are expressed with ontological constructs. For the purpose of service discovery, we distinguish between requesting service descriptions and advertised service descriptions. The proposed semantic framework allows for descriptions with different detail and makes relaxed queries possible.

   The provision of concepts at the level of geographic features (e.g., ‘Building’, ‘Point’) is considered a minimum requirement for a semantic interoperability framework for geo-services. As a backbone, a basic ontology with general constructs is made available, based on the ISO General Feature Model (ISO 19109). The fact that the concept of *feature* can be recursively defined (see Section 5.1.1) provides a flexible solution for modelling phenomena at different levels of generalisation and aggregation.

   Domain ontologies that sufficiently cover specific domains can make use of these basic concepts. Examples have been provided in this thesis for the Dutch basic schema for geo-information NEN3610 and the data model of the Dutch topographic service (TOP10NL).

   The approach of extensibility of concepts that was applied in this research (as described in Section 5.6.1 for single concepts and in Section 8.1.2 for groups of concepts) offers the flexibility to upscale to any level of detail that is needed.

In addition to information modelling constructs, we need to model the characteristics of service functionality to support the exchange of explicit service capabilities. For this purpose, services are described by the operations that they make available. Based on the findings in Chapters 5 through 7 we can state that the elements of the feature concept ontology and the feature symbol ontology have proven to be a solid basis for the description of the input and output parameter types of atomic geo-operation types in the proposed geo-operation ontology.

In conclusion, in this thesis, a basis is provided by an OWL implementation with the following basic elements for a geo-operation description. They are considered to be the essentials that can be modelled semantically and of which at least one should make part of a geo-operation characterisation:

- classification of geo-operation functionality,
- description of operation input and output parameter types,
- description of geodata that is tightly-coupled to the service,
- description of the control flow in (virtual) composite operations.

Each of the above characterisations may be used in isolation to describe an operation type, but in combination they are much more effective.

However, the following main limitations still hamper semantic interoperability in the context of the proposed model:

- **Spatial reasoning.** The geodata that may be tightly-coupled to a geo-service imposes a validity region on the service and may have implications for the meaning of input/output parameters. Service chaining requires to match these regions. As discussed in Section 4.8, there are no built-in constructs in OWL that support such reasoning. Alternatively, spatial relations can be modelled (1) in a separate hierarchy using the property construct of OWL or of another ontology language, or (2) by outsourcing spatial reasoning by conventional geometric computation.
- **Implementation of metaclasses.** The information model and operation model as discussed in Section 5.1.1 apply metaclasses. The inability of current Description Logics (in combination with reasoning algorithms) to provide reasoning support to metaclasses, necessitates the use of naming convention-based mechanisms to create meta-bridges. As discussed in Section 5.7, the creation of meta-bridges between information model and operation model implies the creation of a rather impractical clone structure of classes.

4. *How can we semantically enrich meta-information of geo-information and of geo-services to support service chaining?*

   Geo-information and geo-services may be *annotated* with references to ontological concepts (as demonstrated in Section 7.2). Alternatively, geo-information and geo-services descriptions may reside in a knowledge base, immediately available for reasoning (again, see Section 7.2). As mentioned in the answer to the previous research question, geo-services are characterised by their functionality, input/output, tightly-coupled data and workflow. The annotation of this meta-information in service description is essential for the transition from abstract to concrete composition in a service chaining process.

5. *What are the capabilities and limitations of Semantic Web tools to support geo-semantic interoperability?*

   Currently, several commercial and open source tools are available for the creation, editing and management of RDF-based and OWL-based ontologies. Based on the findings in Chapters 7 and 8, we observe that the tools used (which can be considered mainstream Semantic Web tools) provide basic support for constructing concepts, creating a knowledge base with individuals and connecting to reasoning software. However, these tools are quite immature, compared to what database management tools and programming tools can do in their respective fields. For example, debugging of concept inconsistencies and visual design are in its infancy. In addition, they are lacking intuitive user-interfaces for people without some background in knowledge representation. Typically, the tools support knowledge engineering by the information technologist and they (still) need considerable interfacing to support specific application domains.

6. *What are minimum requirements for a semantic interoperability framework for geo-services?*

   The implementation of the use cases in Chapter 7, taking into account the design issues described in Chapters 4, 5 and 6, has demonstrated that a semantic interoperability framework should at least consist of:

   - A set of ontologies, containing concepts, concept relationships and instances of concepts that represent the domain of discourse sufficiently enough for the intended applications. This thesis work proposes to use the tripartite ontology structure (feature symbol—feature concept—geo-operation) developed in this research. The ontologies may be distributed (maintained at different geographic/organisational locations).
   - A set of mappings between the ontologies that make them consistent as a set. The ontologies should allow for ontology extensions by means of new mappings between new and existing concepts.

- A set of guidelines on how to derive descriptions for the data sets and services, based on the ontologies, and how to use them for matchmaking. Descriptions may be implemented as a set of concepts or individuals, or both.

The following steps are recommended to be taken in building a semantic interoperability *infrastructure*:

(a) Semantic framework building, comprising of ontology building (see Conclusion no. 3 in Section 9.2.3) and prescription of description creation.

(b) Documentation and on-line presentation of the ontologies.

(c) Tool building (preferably by using and/or adapting existing tools):

   i. Creation of dedicated client interface for information providers for the creation of data set/service descriptions, based on ontology concepts.

   ii. Creation of dedicated client interface with examples of queries / reasoning requests for information consumer.

A model architecture for a semantic infrastructure has been proposed in Section 8.5 as a operationalisation of the prototype architecture of Figure 8.12 (Section 8.3.4). A more in-depth discussion on this can be found under the header *Deployment* in Section 9.4.

9.2.3 Specific issues

Specific conclusions that can be drawn from the research are given (per topic) in the subsections below.

Ontology design strategies

1. On the ontology design strategy the following statements can be made:

   (a) The Manchester House style provides useful guidelines for consistent ontology creation (see Section 8.5). The research as discussed herein has successfully applied these guidelines and therefore advocates to use them.

   (b) Regarding the use of classes and individuals in geo-information/service descriptions and the 'core' ontology (without the descriptions), the following can be stated. Classes form the basis of each ontology; individuals instantiate them. Careful design choices have been made at the 'leaves' of the ontology (see the discussion in Section 5.7). For example, the concept *PostOffice* occurs at the leaf of the TOP10NL concept hierarchy, but is it a class or an individual?

In this respect, the OnToGeo core ontology was built according to the requirement of three out of four modes of matchmaking (type II, III and IV, involving individuals), i.e., to include data/service descriptions as individuals: the semantic interoperability framework has been modelled entirely with classes, geo-information/service descriptions have been modelled with individuals (for tests of type II, III and IV). To test type I (between classes) reasoning, some geo-information/service descriptions were also written as classes. The selected strategy was considered to be satisfactory with respect to both testing capacity and the possibility to extend the core ontology with subclasses during further development (with individuals this is impossible).

   (c) During ontology building one has to model as accurate as possible. For example, the concept *nen3610:Forest* has to be modelled as a feature attribute value and not as a feature type, because it as appears as such in the NEN3610 data model (similarly to *BuildingFunctionValue*, see Section 5.3.1). But at the same time it is important to stay pragmatic and avoid unnecessary complicated structures that become hard to understand.

2. Ontology mappings are essential constructs to integrate the content of two or more ontologies. They allow a user to pose a query using the constructs of one ontology and return results based on the inferred constructs of the other. For example, as a result of a specific mapping between the NEN3610 and TOP10NL ontology, a user may request a service that displays 'accommodations' (a NEN3610 ontology concept) and is provided with service advertisements that include hotels (as defined in the TOP10NL ontology). Such mappings have proven to work well across multiple ontologies and for the cross referencing of service functionality (see the mapping between OPERA and ISO 19119 (Services) in Section 6.3.1.

   However, creating ontology mappings is not trivial. It requires in-depth knowledge about the semantics of the domains to be integrated and is preferably managed by domain experts. Attempts, external to this research, have been made to semi-automate the mapping process, with reasonable results (see Section 4.6.1).

3. In general, the steps below have proven to be efficient and sufficient in building the OnToGeo ontologies. In practice, the first four steps have to be done in consultation with domain experts.

   (a) Determine the scope of the ontology.
   (b) Determine how it will be used; if for querying, what constitutes a typical query?
   (c) Create an ontology 'sketch' (on paper or with a software tool for drawing diagrams) containing the most general concepts and associations

between them (see the UML diagrams in Chapter 5).

(d) Create a list of all important concepts (at all detail levels) to be included.

(e) Create the ontology concepts and their relationships with axioms in OWL with an ontology editor. Carefully consider the meaning of using subsumption relations, role restrictions, disjointness, etc. (see Chapters 5 and 6 and the other conclusions in this topic.) Repeat this step for any newly added concepts. Check every new axiom to see if the ontology remains consistent and is well-formed.

(f) Instantiate the concepts with individuals (in case of OnToGeo this involves data set and service descriptions).

Information model

4. Based on the efforts made in Chapter 5, we can state that in general, the ISO 19100 series of geo-information standards are very well documented and provide enough conceptual structure to translate their contents into ontologies. Some exceptions to this observation include the loosely defined relations between features and coverages and the shortcomings of the ISO 19119 service taxonomy (see the conclusion under *Process model*).

5. The ISO 19109 General Feature Model forms the basis of the geo-information ontology as presented in this thesis. Its combined use of classes and metaclasses has problematic implications for the use of according concepts in the ontology that implements this model (see the discussion in Section 5.7). In OWL-DL, classes cannot be treated as instances of other classes and as such, metaclasses cannot be modelled, if we demand decidability (all computations will finish in finite time) of reasoning systems. In this research, meta-bridge mechanisms have been created as a workaround (see the explanation in Section 5.7).

   the form of a bijection between the metaclasses and specific classes in the geo-operation ontology. Materialisation and a naming convention approach have been suggested in literature as alternative workarounds (see Section 5.1.1 for more details), but they have not (yet) been implemented in the prototype.

Process model

6. The ontology mapping efforts discussed in Section 6.3.1 have shown that the geo-service taxonomy, contained in the ISO 19119 (Services) standard provides a useful generic breakdown of service classes. However, they have also demonstrated that ISO 19119 falls short as a single basis for a geo-service ontology, as described in Section 6.5. For this reason, it is recommended

to use the newly developed OPERA ontology in conjunction with the ISO 19119 (Services) taxonomy (a mapping has been provided in the OnToGeo ontology).

7. Sections 6.2 and 6.4.5 have shown that OWL-S has proven to be a valuable basis for the modelling of geo-web services. This is besides its clear model structure, simply because it is based on OWL and lets us define extensions for input/output parameters for services and link them with service categories in a taxonomy. Reasoning with the control flow within OWL-S is possible with matchmaking type II, but class requests can become complicated with multiple operations involved. A more simple (and obviously less powerful) 'bag' type of construct is missing in OWL-S and has therefore been developed in this research as to support relaxed queries for control flow.

Matchmaking

8. Matchmaking between data set/service descriptions can be done in a number of modes, as presented in Section 6.4.2. Firstly, testing with 'split' (without concept intersections) data/service descriptions provides a more refined matchmaking than with aggregated descriptions (containing concept intersections). Secondly, class-based descriptions are more flexible than the ones based on individuals, but with the current user interfaces, individuals are easier to create. Based on the pragmatics of finding a right balance between query expressiveness and usability, the current prototype was implemented both with class-based data/service requests and individual-based advertisements. The advertisements are aggregates and the requests are either aggregates or atoms, based on the refinement needed in the query.

Prototype environment

Based on the experiences reported in Section 8.5, the following conclusions are drawn:

9. The current prototype environment is not yet ready for instant deployment. Major issues are (1) the need for a user-friendly interface for entering service descriptions as class definitions and (2) the need for tools that support ontology comprehension by information engineers.

10. The prototype environment is currently limited by the reasoning mode (basically type II) it can handle. In addition, the method of providing a relaxed match is very rudimentary. However, the prototype has been developed in such a way that it allows for extensions with reasoner functions, other than the ones used, with fairly low effort.

9.3 Main contributions

The thesis has made the following main contributions:

1. The research has given an overview of the potential and impediments of current interoperability models for distributed geodata and geo-services. It has concentrated on the influential standards of the ISO technical committee for Geographic information/Geomatics (ISO/TC 211) and the specifications of the Open Geospatial Consortium (OGC). The developed solutions are therefor compliant with current mainstream geographic standards.

2. The research has provided solutions for modelling the semantics of geo-information and geo-services with the design and implementation of machine ontologies, based on existing geo-information standards by ISO and OGC (which was reported in Chapter 5). The solutions are flexible and extensible. This was realised by the following achievements:

   (a) A theory has been provided (in Section 5.1, and based on Section 3.1.1) for the separation of ontology types ('concept' and 'symbol' world) , based on the layered abstraction of geo-information and their conceptual relationships with a geo-operation ontology ('OPERA').

   (b) A generic geo-operation ontology (described in Sections 5.4 and 5.5) that serves as a reference for service descriptions has been designed and implemented in such a way that it can accommodate common geo-operations as instances. In addition, it allows for the extension with sub-ontologies containing more specific geo-operations.

   (c) Flexibility was achieved by using Semantic Web technology, allowing reasoning with imprecise requests and advertisements of (geo-)information and (geo-)services (see the examples given in Sections 7.1.1 and 7.4.3). This allows users to provide imprecise data/service requests and advertisements, whilst matchmaking will still give results. This would not be possible without the semantic structure as a backbone. The results may not represent a match of all data/service details but it can be accurate enough for the purpose of the information discovery or source integration. This is an important asset for GII's that are typically faced with different levels of metadata detail in the provision of advertisements and requests.

   (d) Extensibility was built into the design by splitting the ontology up into well defined domains for information concepts (e.g., 'travel', 'riskmap') and process concepts ('OPERA-R', 'OPERA-D') and by applying a different namespace to each domain respectively.

   (e) Ontological constructs have been documented in the thesis in Description Logic notation. This helps to better understand the OWL implementation of the prototype and facilitates eventual implementations other than OWL.

3. The taxonomy of GIS-operations which has formed the basis for the design of the geo-operation ontology ('OPERA'), has been documented in such a way that it can also serve in 'textual' form as a reference for informal service descriptions. In addition, a mapping with the ISO 19119 service taxonomy has been documented.

4. The research has resulted in a proposal for a semantic interoperability framework for geo-services, comprising of ontologies and prescriptions for creating (1) ontology-based descriptions of geo-information sources (data sets and services) (discussed in Section 5.6) and (2) inter-ontology relationships (discussed in Section 6.3). It is based on the fundamental ISO 19109 (GFM) standard and accommodates the meta-information standards of ISO 19115 (Metadata) and ISO 19119 (Services). The research has shown by means of use case implementations (Chapter 7) how this framework can be used for at least the following applications:

   (a) Data set integration
   (b) Geo-information and service discovery and abstract composition as part of service chaining.
   (c) Documentation of data set lineage and service chains

   Design alternatives have been proposed for several parts of the framework (e.g., with respect to workflow models and information source descriptions). In addition, alternative uses of the framework have been indicated (e.g., with respect to service matchmaking)

5. Generally speaking, this research has presented the application of Semantic Web technology to existing geo-information models. It has shown its applicability in different geographic application domains, ranging from project-oriented to ad hoc activities and from data to service oriented environments.

6. The research has resulted in a prototype environment which comprises an integrated toolset ('GeoMatchMaker') for the exploitation of ontologies as well as the complete set of developed geo-ontologies ('OnToGeo') (see Chapter 8). This prototype can be used by a data/service provider to describe the offered products as well as by a data/service consumer that wants to discover and integrate these products. In addition, the prototype can be used by an information engineer to extend the existing geo-operation ontology with new operation types and to plug-in new conceptual data models, similar to the examples of TOP10NL (data model of the Dutch Topographic Service) and application domains ('Travel', 'Riskmap'). Except for the reasoner, the prototype environment is entirely built with open source software.

General applicability A note on the general applicability of the research findings. The research results and lessons learned are, in certain aspects, not confined to the field of geo-information and geo-services. This applies especially to the following issues as raised in the thesis: Research question 1 (Section 9.2.2), implementation of metaclasses in research question 3, ontology design strategies (Section 9.2.3), and general aspects of ontology mapping, process modelling and matchmaking.

Other research findings may apply to fields that resemble the geo-information field in specific aspects. For example, an application field that deals with multiple versions of its 'features'[1], will have to deal with ontology mappings in a similar way as presented here. Further, users of applications that are characterised by a regular reuse of services and data, may also refer to the non-geo-specific findings of research question 6 (semantic interoperability framework) and Section 9.4 (deployment).

9.4 Deployment

As the proposed semantic interoperability framework has been developed in an obviously conditioned prototype environment, its immediate deployment in a specific application requires the creation of a semantic infrastructure as described in Section 4.6 and in the answer to the last research question in Section 9.2.2. In such an infrastructure, the presented feature symbol ontology (SYMBOL) and geo-operation ontology (OPERA) as discussed in respectively Sections 5.2 and 5.4 are immediately deployable.

In addition, an application ontology has to be built for the specific application at hand. This application ontology may refine OPERA by adding subclasses of operations that are specific to the application. It may also need to extend existing domain ontologies with respect to feature concepts. Existing domain ontologies may be found in libraries, such as *SchemaWeb* (see Section 4.6.4). Once the proper ontologies are found, a complete set of mappings needs to be established between them. In this form, the framework can function as a basis for semantic service descriptions.

Multi-user environments may be needed in some applications. They are supported by some of the tools used in the prototype, but they have not been tested in this thesis work as such.

The proposed semantic interoperability framework can be applied to a variety of fields, as demonstrated in the use cases. Application fields that are thought to benefit from the developed theory in the short term are:

1. Harmonisation of NEN3610 and its sector models; Creation of ontology-based meta-information and application of matchmaking.

2. Generalisation of geographic features (such as in database/geographic map

[1]See Section 1.1, paragraph 'What distinguishes geo-information services from other services?'

generalisation). Semantics are needed for making the right decisions with respect to (re)moving and grouping of feature instances.

3. Data set and service discovery as part of catalogue mechanisms in Geo-information Infrastructures (GIIs) at all organisational levels.

4. Ontology presentation to users of a specific data model (e.g., TOP10NL). This requires a comprehensive visualisation tool.

5. The matching of user profiles with service advertisements, for example to create *context-awareness* in LBS applications.

6. Quality assessment and improvement of metadata for data sets and services. Reasoners can check the consistency and completeness of ontology-based information source descriptions, by integrally inferring the ontological relationships that exist between them.

Organisational issues

The creation and maintenance of a semantic infrastructure as part of a GII needs the commitment of several organisations. The responsibilities of key organisations is proposed as follows with reference to the model architecture depicted in Figure 8.12 (Section 8.3.4). The feature symbol ontology and the fixed part of the geo-operation ontology (in the prototype environment contained in the OnToGeo ontology as respectively SYMBOL and OPERA-R) should be maintained by OGC. OGC should also issue specifications on the methods for ontology-based description of information sources (services, data sets, etc.). Eventually, ISO TC/211 may recognise the abstract part of these specifications as standards.

The feature concept ontologies (e.g., NEN3610 and RISKMAP in the prototype) and the extensible part of the geo-operation ontology (OPERA-D) should be maintained by so called information communities (see its definition in Section 5.1.2), that have a mandate within the GII to do so. Each information community should be responsible for mapping its ontology to global ontologies and, if applicable, to the local ontologies of other information communities. The coordination of such mapping activities may be steered by a commonly agreed method and representation of mappings, possibly materialised in a standard.

The rest of Figure 8.12 is part of the semantic infrastructure that lies outside the semantic interoperability framework. Crucial elements are the web service registry and the tools for service chaining. A web service registry could be managed by a mandated unit within the GII or by commercial clearing houses. Tools for service chaining may be provided by any software developer, commercial or non-profit.

9.5 Recommendations for further work

Based on the outcomes of this research, the following recommendations are made for further work:

1. As stated in the answer to the third research question (Section 9.2.2), there is a need for spatial reasoning about tightly-coupled data in order to match geo-service validity regions. Further work is needed to incorporate existing approaches that apply spatial hierarchies.

2. The creation of advertisements (by a data set/service provider) and requests (by a consumer) needs the development of more user-friendly graphical user interfaces (GUIs). Although the Protégé ontology editor supports entering individuals with the help of rather user-friendly forms (similar to database entry forms), the implementation of ontology class-based descriptions is rather difficult. Moreover, a normal user cannot be expected to familiarise herself with an ontology editor. An interface is suggested with which a user can enter a query with help of menus, keywords and keyword suggestions that are drawn from the ontology. This could also involve a query-by-example interface (as mentioned in Section 6.3.3).

3. Matchmaking between classes (type I) needs a more thorough interpretation mechanism for its result types 'PlugIn', 'Intersection', etc. This may help this method to be better applicable in the current prototype.

4. More flexible ontology visualisation tools are needed, so that ontologies can be made more understandable to a wider audience and inspection of ontologies becomes easier. Such tools should at least include:

   (a) The option to display any combination of ontology construct types (class, individual, role, role restriction).

   (b) A facility to zoom, based on subsumption levels and role depth.

   (c) Expandable ontology items, driven by mouse clicks.

   (d) An option to toggle labels on and off.

   (e) An export facility to common graphic formats (including vector formats such as SVG).

5. The current GeoMatchMaker prototype has limitations with respect to the service chaining process. The following actions that are currently left to the human user should also be subject to automation:

   (a) Result ranking. A ranking mechanism, based on data set/service descriptions would help the user in choosing the best information source.

(b) Meta-information propagation. The automatic calculation of meta-information propagation and evaluating candidate combinations that constitute a given task (expressed in Description Logic), would help a service consumer to find service chains faster, and to eventually perform a simple simulation. It can also help a service *provider* in creating service chain descriptions.

6. Some more recent Semantic Web techniques have not been taken into account in this research. It is worthwhile to investigate whether they can contribute to the improvement of the prototype. These techniques include rule-based reasoning (e.g., *Semantic Web Rule Language* (SWRL)) and dedicated ontology query languages (e.g., *OWL query language* (OWL-QL) and *new Racer query language* (nRQL)).

7. An issue left open is the one of (semi-)automatic ontology mapping. The manual creation of ontology mappings is a tedious job. Section 4.6.1 has introduced some approaches for the semi-automatic identification of mappings. This research recommends to test such semi-automatic mapping methods to establish links between the ontologies currently in OnToGeo and potential new ones to be integrated.

8. Query relaxation is a useful instrument to increase the recall of search results. Strategies are needed to determine the variables to be removed from the query, unless it is clear which variable is targeted (such as the input parameters of an operation). The approach, reported in [257] relies on rather general heuristics and may not work in our specific situation. However, this is left for further research.

9. Tools are needed for helping a data set/service provider with the annotation of these information sources. First prototypical examples are found [39, 141, 221]), but they are application specific, many of them use proprietary annotation tags and some of them are still buggy. A tool is suggested that is universal, by taking as input (1) the ontology that serves as the annotation base and (2) the *structure* of the document to be annotated, e.g., the XML schema of it.

10. Semantic Web technology is slowly finding its way into the OGC community (see the end of Section 2.4.6). Based on the discussion in Section 9.4, this research recommends OGC and ISO/TC211 to establish a semantic interoperability framework and use the ideas of semantic service annotation (as demonstrated in Section 7.2.1) and ontology based matchmaking (demonstrated throughout Chapter 7) to make more formal the existing geo-service standards (such as the GetCapabilities request/response pair of OGC Web Services (OWS), ISO 19115 (Metadata) and ISO 19119 (Services)) and to allow semantic queries on the content of the resources that comply with these standards.

Appendix A

UML notation

Some of the UML elements used in this thesis are represented with a notation that deviates from the standard UML 2 notation as specified in [202] and described in [11]. This was done in order to enhance the visualisation and avoid diagram congestion. The deviating notation is depicted in the figure below. All other UML elements used in the thesis are represented according to the UML 2 standard.

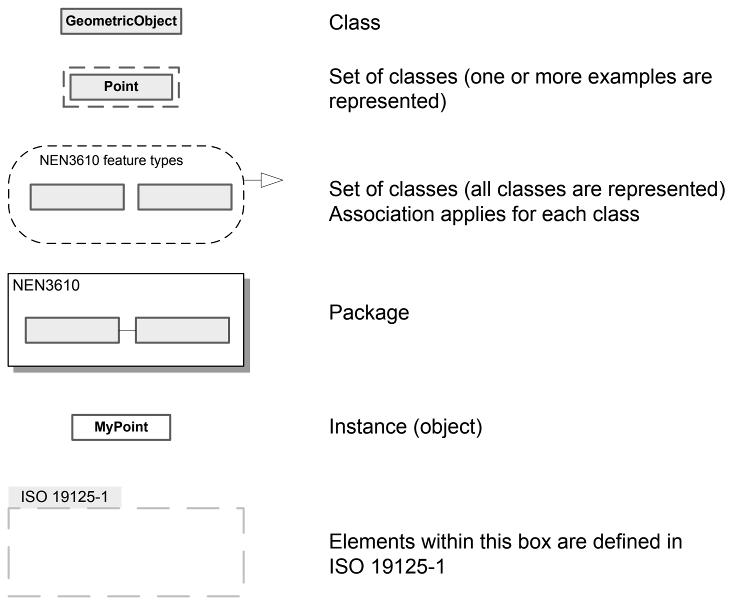

GeometricObject
Class
Point
Set of classes (one or more examples are represented)
NEN3610 feature types
Set of classes (all classes are represented) Association applies for each class
NEN3610
Package
MyPoint
Instance (object)
ISO 19125-1
Elements within this box are defined in ISO 19125-1

Figure A.1: Deviating UML notation used in this thesis.

Appendix B

ISO 19100 overview

This appendix lists all current ISO 19100 standards with their numbers and short names. The ISO 19100 standards are the result of the work of ISO Technical Committee 211 (ISO/TC211) (see an introduction in Section 2.4.7). The short names in the last column of the table below are used throughout the main text of this thesis as reference names. Not all standards have the official status of *International Standard* yet. Current information regarding this status is available on-line at http://www.isotc211.org/.

Standard No.	Standard name	Short name
6709	Standard representation of latitude, longitude and altitude for geographic point locations	LatLon
19101	Reference model	Reference
19101-2	Reference model - Part 2: Imagery	RefImg
19103	Conceptual schema language	Concept
19104	Terminology Introduction	Terms
19105	Conformance and testing	Conform
19106	Profiles	Profiles
19107	Spatial schema	Spatial
19108	Temporal schema	Temporal
19109	Rules for application schema Note: ISO 19109 contains the ISO General Feature Model (GFM)	GFM
19110	Methodology for feature cataloguing	Catalog
19111	Spatial referencing by coordinates 19111 - Revision of ISO 19111:2003	RefCoord
19112	Spatial referencing by geographic identifiers	RefIdent
19113	Quality principles	Quality

19114	Quality evaluation procedures	QualityEval
19115	Metadata	Metadata
19115-2	Metadata - Part 2: Extensions for imagery and gridded data	MetaImg
19116	Positioning services	Position
19117	Portrayal	Portrayal
19118	Encoding	Encoding
19119	Services	Services
19120	Functional standards	Functional
19121	Imagery and gridded data	Imagery
19122	Qualifications and Certification of personnel	Certify
19123	Schema for coverage geometry and functions	Coverage
19124	Imagery and gridded data components	ImgComp
19125-1	Simple feature access - Part 1: Common architecture	SFeature
19125-2	Simple feature access - Part 2: SQL option	SFeatureSQL
19126	Profile - FACC Data Dictionary	FACC
19127	Geodetic codes and parameters	Geodetic
19128	Web Map server interface	WMS
19129	Imagery, gridded and coverage data framework	ImgFrame
19130	Sensor and data models for imagery and gridded data	Sensor
19131	Data product specifications	Product
19132	Location based services - Reference model	LBSRef
19133	Location based services - Tracking and navigation	LBSNav
19134	Multimodal location based services for routing and navigation	LBSMulti
19135	Procedures for registration of geographical information items	RegItem
19136	Geography Markup Language	GML
19137	Generally used profiles of the spatial schema and of similar important other schemas	SchemaProf
19138	Data quality measures	QualMeas
19139	Metadata - Implementation specification	MetaImplem
19140	Technical amendment to the ISO 191** Geographic information series of standards for harmonization and enhancements	Harmonize
19141	Schema for moving	Moving

Appendix C

OPERA-R geo-operation types

This appendix lists the atomic reference types for geo-operations in the OPERA-R ontology and describes their functional semantics (the *R* in OPERA-R stands for 'reference operations'). This appendix is linked to Section 5.4. The list can be considered to be the base taxonomy for OPERA-R. The input/output parameters of each operation are given in Appendix D.

C.1 Human interaction operations

- **InvokeDataOperation**: Parses a user command to activate a geo-operation of the other categories (operations on feature data, operations on services, operations on metadata). For example, it may invoke the *Intersection* operation to overlay two map layers.

- **MapDisplay**: This operation allows a user to interact with geodata through a graphic display. Operations of this type are typically used by map viewer services and may involve the following operations: ShowMap, PanMap, ZoomMap. They trigger the *SelectByContains* operation and the *Extract-GeoInfo* operation (see operations on feature data).

- **FeaturePropertyDisplay**: Based on a feature selection, this operation renders feature properties for displaying in the client, other than a map display, e.g., a table display. This operation triggers the *GetPropertyValues* operation (see operations on feature data).

- **MetadataInteract**: Operation that provide user interaction with a metadata store.

- **ServiceManagementInteract**: Operation that provide user interaction with service (chain) management operations.

C.2 Feature modelling

Feature modelling operations change the feature model of a data set through its feature types and feature property types. Feature *instances* operations are covered under Section C.4.1.

- **ActOnFeatureType**: An operation that creates or deletes a feature type or changes its meta-information (such as its name). An operation that deletes entire feature types also deletes all its instances and geometric objects, if existing.

- **ActOnFeaturePropertyType**: An operation that creates or deletes a feature property type or changes its meta-information (such as its name). An operation that deletes entire feature property types also deletes all its instances, if existing.

The subclasses create, delete and change are subtypes of the above 'ActOn' operation types. They appear in OPERA as respectively as subclasses. For example, the subclasses of ActOnFeatureType are *CreateFeatureType*, *DeleteFeatureType* and *ChangeFeatureType*.

C.3 Feature access

- **DataSourceAccess**: Operations that are needed to access a data source, such as opening a connection to a database or sensor stream and authentication operations.

- **ExtractGeoInfoFromStream**: This operation extracts features from an input stream. It may be used in sensor services such as a positioning service.

- **ExtractGeoInfoFromDB**: Based on a feature type selection and a map extent, this operation extracts features from a database. It (1) makes them directly available or (2) renders a map for displaying in a client application. Subclasses corresponding to (1) and (2) are respectively :
  - ExtractFeature
  - ExtractMap

- **DataSourceManagement**: Operations that involve the management of a data source (e.g., authentication, query optimisation, tiling etc.).

- **ExtractMetaInfo**: Extracts meta-information from features (e.g., data structure informations, data types, number of features, feature names, etc.) This operation does not annotate metadata to a data set; this is done by the *AnnotateMetaInfo* operation (see Section C.8)
- **FormatConversion**: Converts the data exchange format.

C.4 Feature processing

The classes in this section have been chosen to have as little overlap as possible. For this reason, certain operation categories that are commonly known from practice may not occur as named classes. An example is 'terrain analysis'. This class spans several of the classes listed below. In general, it holds the operations that involve geometric objects and thematic attributes, of which the latter represents the terrain height attribute. Note that there are other cases of operations that may not have a named class, but which sub-operations *can* be classified in one or more of the classes listed in OPERA-R.

C.4.1 Feature instantiation

Note that this category deals with feature *instances* only. Operations that change the feature *model* are covered under Section C.2.

- **ActOnFeatureInstance**: An operation that creates or deletes a feature instance (including its geometric object, if existing).
- **ActOnFeaturePropertyInstance**: An operation that creates or deletes a feature property instance.

The subclasses create, delete and change are subtypes of the above 'ActOn' operation types. They appear in OPERA as respectively as subclasses. For example, the subclasses of ActOnFeatureInstance are *CreateFeatureInstance*, *DeleteFeature-Instance* and *ChangeFeatureInstance.*

C.4.2 Across attribute types

These operations use feature attributes of one type to calculate feature attributes of another type. This involves:

- **SpatLoc**: Operations that use spatial attributes to return location attributes (e.g., an address finder). This is done with help of a geographic query on a tightly coupled data set.
- **LocSpat**: Operations that use location attributes to return spatial attributes (e.g., a gazetteer). This is done with help of a lookup table or geographic query on a tightly coupled data set.

C.4.3 Among thematic attributes

These operations change the thematic attributes of features. To be more specific, thematic attributes are distinguished in terms of levels of measurement, see Section 5.2.3. The operations identified, are partly based on the classification by Chrisman [50].

The first group needs as input the values of one feature attribute type of one feature. These operations act on the domain of values (cf. feature symbol *DomainOfValues*) of an attribute and may need additional 'decision' information or external parameters in the form of so called lookup tables to perform successfully. The change in the domain of values also changes the actual attribute values.

- **Group**: The Group operation combines classes into another class. For example, land use classes 'DeciduousForest' and 'ConiferousForest' may be grouped as one class named 'Forest'. The Group operation transfers a nominal classification into another nominal classification.
- **Classify**: The Classify operation creates ordinal classes from interval, ratio, absolute, or count measurements. For example it converts population figures to population classes 'Village', 'SmallTown', 'LargeTown'.
- **Rank**: The Rank operation creates an order in classes. It converts from a nominal scale to an ordinal scale. For example, it may rank restaurant types in terms of a person's preferences.
- **Scale** The Scale operation changes the units of a ratio measurement. For example, it may change the units of a distance measurement from miles to metres.
- **Evaluate**: The Evaluate operation changes measurements in the nominal, ordinal or interval level to the interval level. For example, it may convert the categories 'Old' and 'New' into a calender year. It is obvious that this operation needs new information for each feature instance.
- **Separate**: The separate operation is used to separate attributes that have been concatenated with a concatenate operation (see the *CrossConcatenate* operation below). It creates separate attribute types.

The operation below does not act on the domain of values, but directly on the attribute values.

- **Calculate**: The Calculate operation performs a calculation between attribute values, measured at interval, ratio, count, or absolute levels, resulting in a value at the interval ratio, count or absolute level. Some restrictions exist, for example interval scales do not support multiplication or division and the difference between two interval values results in a ratio value. Calculations may involve any mathematical or statistical function. Examples of the latter are *Average, Count, Deviation, Frequency, Maximum, Minimum.*

The operations below combine values from different attribute types for one feature type. Chrisman's classification is only partly followed, because his classes 'Sum and Difference' and 'Rate and Density' are too restrictive and prevent a structured extension of this category with new operation types.

- **CrossConcatenate**: The CrossConcatenate operation 'glues' the attribute values in a new code, without information loss. The result is at the nominal level. For example, it may combine a land use attribute and elevation zone attribute, for which the original values of a feature instance are 'Urban' and 'BelowSeaLevel' and the resulting value is 'UrbanBelowSeaLevel'. It can be also used to interleave the bands of a satellite image.
- **CrossCalculate**: The CrossCalculate operation performs similarly to the *Calculation* operation described above, with the difference that it needs values from *different* attribute types for one feature type.
- **Proportion**: The Proportion operation is a special case of the CrossCalculate operation. It divides two values, measured at the same scale on the ratio, count or absolute level. The result is a proportion and resides at the absolute level. For example, a proportion operation may divide the population count of a province by the total country population count, which results in a percentage of the total country population (an absolute value).

Note: The 'isolate' operation as identified in this same category by Chrisman appears here in Section C.4.4.1 as 'SelectByThemQuery' operation. This is due to the fact that it has essentially different input/output parameters.

C.4.4 Feature selection

C.4.4.1 Feature selection based on thematic attributes

- **SelectByThemQuery**: The SelectByThemQuery operation selects features, based on a query on thematic attribute types. For example, an SQL query may select the features of type 'House' that have more that three floors. The SelectByThemQuery operation does not involve spatial selections; this is covered by other operations. The SelectByThemQuery operation may be also used for the simple selection of a 'map' layer containing one feature type.

C.4.4.2 Feature selection based on geometric objects

These operations select features based on a topological relationship with an geometric object. They may also output the actual features with all their attributes. The topological relationships of Section C.4.7 are used. The following operation types are identified: **SelectBy**<*Topological relationship*>, where <*Topological relationship*> is one out of the nine relationships defined in Section C.4.7. In case these operations take geometric objects as their input, topological tests are included in the operation mechanism. These tests are not necessary in case the input consists of already existing topological associations between the objects or the input involves topological objects.

C.4.5 Get feature property values

These operations extract the property values of selected features and forward them to an output stream.

- **GetAttributeValues**: Gets the values of GF_AttributeType. For example, this may be 'RoadWidth = 12' from a road feature's GF_ThematicAttributeType 'RoadWidth'.
- **GetAssociations**: Gets the associations that exist for the selected features
- **GetFeatureBehaviour**: Provides the stored behaviour of the features. More specifically in terms of ISO 19109 (GFM), it retrieves the methods defined by GF_Operation for each feature.

C.4.6 Geometric object change

These operations change the spatial attributes of features. ISO 19109 (GFM) considers spatial attributes of features to be represented by a spatial object (geometric or topological). In the ISO Simple Feature Model (ISO 19125-1), spatial objects may be single geometries or geometry collections. The latter can be used for aggregated geometries (such as a country feature, constituted by an aggregate of its mainland and its island polygons) [1]. The following operations are identified:

- **DeleteGeoPart**: Delete parts of geometric objects.
- **MoveGeoPart**: Move parts of geometric objects.
- **MoveGeoWhole**: Move entire geometric objects.
- **SimplifyGeo**: Simplify geometric objects : composite of the above two.
- **MergeGeo**: Merging geometric objects.
- **GeneraliseGeo**: Generalisation of geometric objects : composite of the above.

Deleting an entire geometric object is performed with the ActOnFeatureType operation (see Section C.2).

Examples can be found in Figure C.1.

[1] In the more advanced ISO Spatial Schema standard (ISO 19107) these geometries are modelled as GM_Primitive and GM_Aggregate classes respectively.

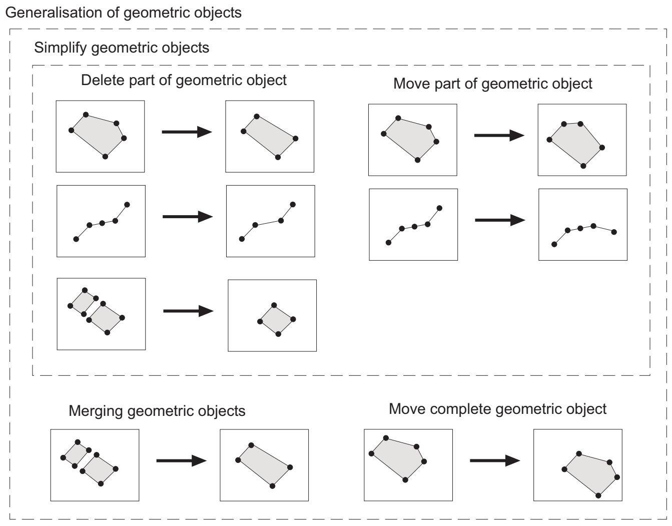

Generalisation of geometric objects
Simplify geometric objects
Delete part of geometric object
Move part of geometric object
Merging geometric objects
Move complete geometric object

Figure C.1: Operations on geometric objects.

C.4.7 Topology

These operations test the topological relationship between geometric objects. Topological relationships have been defined by Egenhofer by means of the possible intersections of geometric object boundary, interior and exterior in the so called 9 intersection model (9IM) [65]. His model was extended by Clementini et al. with dimension information, in the dimension extended method(DEM) [52] and expressed in object-calculus terms in the calculus based method (CBM). This model is as expressive as the 9IM, but groups the potential topological relationships in a basic set of five general operation types that can be applied on area, line and point objects (see Table C.1). The grouping makes it easier for end-users to apply in a GIS environment, yet its underlying operators can be used for implementation. The distinction between topological relationships becomes clear from the decision tree in Figure C.2. In the figure, λ_1 and λ_2 denote the feature type (represented by their geometric objects). λ^o, denotes the interior of λ. The function *dim* returns the dimension of a point-set.

The above operators have been adopted in the ISO standard on simple features (ISO 19125) through the specification of predicate tests. In software implementations these predicate test should be made available through methods supported by geometries. The standard uses slightly different terms, i.e., *Touches, Within, Crosses, Overlaps and Disjoint* for the CBM operations. They are specified as follows (example of *Touches*):

	A/A	L/L	P/P	L/A	P/A	P/L
Touch	x	x		x	x	x
In	x	x	x	x	x	x
Cross		x		x		
Overlap	x	x				
Disjoint	x	x	x	x	x	x

Table C.1: The five topological relationships of the calculus based method (rows in the table) compared to the geometric object combinations to which they can be applied (columns in the table). A = Area, L = Line, P = Point. The crosses indicate a possible application of the relationship. Table derived from [52].

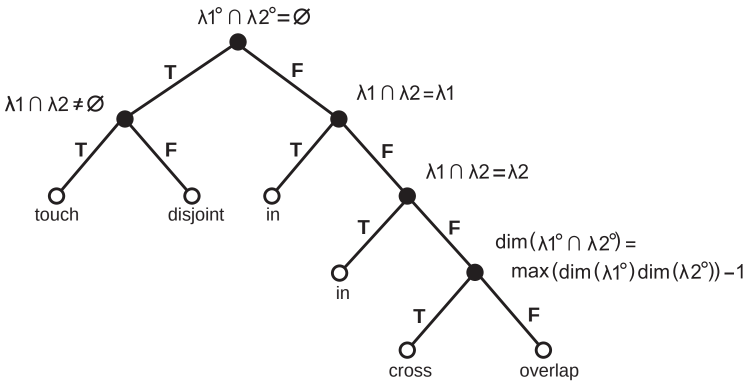


Figure C.2: Topological relationships decision tree. Figure taken from [52].

Touches(*anotherGeometry* : *Geometry*) : *Integer*
Returns 1 (TRUE) if this geometric object 'spatially touches' anotherGeometry.

It has to be noted that the application of topological relations as described in the ISO 19125 standard (SFeature) is not completely conform Table C.1. In contrast to the CBM, (1) it allows the 'overlap' relation between points and (2) it allows the 'crosses' relation between point and area and between point and line. This is due to the fact that ISO 19125 (SFeature) considers Points as well as MultiPoints to be included in 'P' the relation table. Also, MultiLineStrings and MultiPolygons are included in respectively 'L' and 'A', but this does not change the table.

ISO 19125 (SFeature) also adds the operations *Contains, Intersects, Equals and relate.* They are specified as follows:

- *Contains*, defined as $a.Contains(b) \Leftrightarrow b.Within(a)$
- *Intersects*, defined as $a.Intersects(b) \Leftrightarrow \neg(a.Disjoint(b))$
- *Equals* (Returns TRUE if geometries are spatially equal)
- *Relate* (Allows for a more elaborated testing on intersections between the interior, boundary and exterior of two geometric objects. It uses a pattern matrix, representing the intersection pattern of the 9IM.)

In the OPERA ontology the above topological relationships are used to identify operation types that enable the testing of topological relationships between geometric objects. In summary, the following operation types are identified, following the semantics of ISO 19125 (SFeature):

- **Touches** Tests whether two geometric objects spatially touch.
- **Within** Tests whether one geometric object is spatially within the other.
- **Contains** Tests whether one geometric object spatially contains the other.
- **Crosses** Tests whether one geometric object spatially crosses the other.
- **Overlaps** Tests whether one geometric object spatially overlaps the other.
- **Disjoint** Tests whether one geometric object is spatially disjoint to the other.
- **Equals** Tests whether one geometric object is spatially equal to the other.
- **Intersects** Tests whether one geometric object spatially intersects the other.
- **Relate** Tests whether one geometric object is spatially related the other. The relation is specified by the pattern matrix.

For simplicity, a general operation is added to OPERA that integrally performs the above tests. It inputs geometric object pairs and returns the topological relationship of each pair:

- **CreateTopo**: Operation that uses geometric object pairs to calculate topological relationships between these objects. It returns either
  - a spatial association type (GF_SpatialAssociationType) with a topological association role (GF_AssociationRole, being Touches, Within, Contains, Crosses, Overlaps, Disjoint, Equals, Intersects or Relate), or
  - topological objects (as defined in ISO 19107 (Spatial): TP_Object)

C.4.8 Metric measurement

Measurement operations return the following property values

- of geometric objects: Volume, Area, Length, and Perimeter
- between geometric objects: Distance, Bearing

These operations have been defined as operations on features (GF_Operation) in ISO 19107 (Spatial) and partly in the ISO 19125 (SFeature). The associated geometric objects are provided in Table C.2.

	Simple Features ISO19125-1	Feature model ISO19107
VolumeMeasure	Not defined	GM_Solid GM_MultiSolid
AreaMeasure	Surface MultiSurface	GM_GenericSurface GM_MultiSurface GM_Solid GM_MultiSolid
LengthMeasure	Curve MultiCurve	GM_GenericCurve GM_MultiCurve
DistanceMeasure	Between Geometries	Between GM_Object's
PerimeterMeasure	Not defined	GM_GenericSurface GM_MultiSurface
BearingMeasure	Not defined	Between GM_Point's

Table C.2: Measurements on geometric objects.

Note 1: The DistanceMeasure operation can also be classified under the category *Distance based.*
Note 2: Operations that provide statistics on measurements (e.g., providing the polygon with the greatest area) are considered to create a thematic attribute type based on a metric measurement (e.g., 'area') for each feature instance and then apply the *Calculate* operation on the attribute values.

Other operations compare the spatial characteristics of geometric objects:

- **MatchGeo**: Identifies whether geometric objects or parts of them are similar. This used to evaluate whether they could represent the same feature or feature part (e.g. edge matching).

C.4.9 Overlay

These operations use topological relationships between geometric objects to create new features and/or create new attribute values. Newly created features carry a combination of attributes of the input features. An overlay operation contains a spatial combination part and an attribute combination part. Overlay operations are defined for all three categories of object features, grid coverages and grid cells, but they are implemented differently for object and grid. Grid coverages and grid cells that are being overlayed, are considered to have the same spatial reference. Their overlay only contains the attribute combination part. This equates this operation, with the CrossCalculate operation (for grid coverages and grid cells). With the above distinction in mind, we identify two sub-operations:

- **ObjectOverlay**: Overlay of two or more object features.
- **GridOverlay**: Overlay of two or more grid coverages or grid cells.

Spatial aspects of overlay For the spatial part, four set operators are identified, following the methods identified for spatial analysis on geometric objects in ISO 19125 (SFeature):

- **Intersection**
- **Union**
- **Difference**
- **Symmetric difference**

ISO 19125 (SFeature) specifies these operations for geometric objects as follows (example of *Intersection*):

> *Intersection*(*anotherGeometry* : *Geometry*) : *Geometry*
> Returns a geometric object that represents the Point set intersection of this geometric object with anotherGeometry.

Examples of the operations are depicted for polygons in Figure C.3. As a geometric object may involve a GeometryCollection, the operations may execute on more than one geometry object pair.

Attribute combination of overlay After the spatial combination of features, their thematic attributes can be combined and attached to the newly formed features. A distinction between the methods of attribute combination is made by Chrisman [50], based on the combinator rule used:

- Enumeration rule: The individual thematic attributes of the input features are preserved as concatenations, similar to the CrossConcatenate operation in Section C.4.3
- Contributory rule: Each (or a subset) of the individual thematic attributes of the input features contribute to the output attribute, by means of a function.

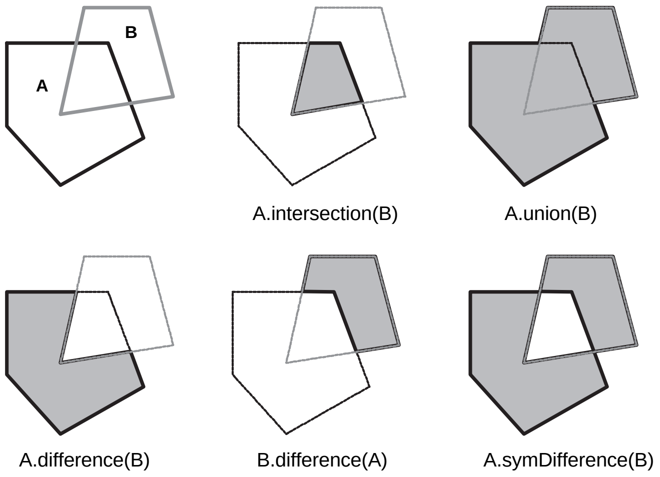


Figure C.3: Overlay operations between geometric objects. Figure taken from [279]. For an explanation of the operations, see Table C.3.

- Dominance rule: The thematic attribute assigned to the output feature is one value chosen from the individual input attributes, e.g., the maximum or an attribute from a predetermined feature type. The dominance rule is a special case of the contributory rule.
- Interaction rule: Similar to contributory rule, but the attributes re combined pairwise in steps. The combination criterion/function in each step may differ.

C.4.10 Distance-based

These operations use distances between geometric objects to select features or create new features.

- **EqualDistance**: The EqualDistance operation creates geometric objects that have an equal distance to input geometric objects (e.g., an operation that calculates Thiessen polygons).
- **FixedDistance**: The FixedDistance operation creates geometric objects that have a fixed distance to input geometric objects (e.g., a buffer operation).
- **Nearest**: The Nearest operation identifies geometric objects that are nearest to input geometric objects.

Operation	Meaning
A.intersection(B)	The intersection of two Geometries A and B is the set of all direct positions which lie in both A and B.
A.Union(B)	The union of two Geometries A and B is the set of all direct positions which lie in A or B.
A.difference(B)	The difference between two Geometries A and B is the set of all direct positions which lie in A but not in B.
B.difference(A)	The difference between two Geometries B and A is the set of all direct positions which lie in B but not in A.
A.symDifference(B)	The symmetric difference of two Geometries A and B is the set of all direct positions which lie in either A or B but not both.

Table C.3: Meaning of overlay operations between geometric objects as depicted in Figure C.3.

C.4.11 Neighbourhood

Neighbourhood operations use the attribute values of features surrounding a target feature to create new attributes and/or features. Such operations may also be performed for more than one target feature. The neighbourhood of a feature can be defined in terms of metrics (i.e., Euclidean distance) or in terms of topology (i.e., with the *intersects* relationship [50]. Topological near features may be far in terms of distance. Neighbourhoods are applicable to object features as well as to grid coverages. The attribute values of the input features are combined in the same way as in the overlay operation (see Section C.4.9. For example, a buffer operation can be seen as a neighbourhood operation, creating a feature with attribute values according to the contributory rule.

In this category we can distinguish:

- **CalculateSlope**: This operation calculates the rate of change of elevation [12].
- **CalculateAspect**: This operation calculates the direction that a surface faces [12].
- **GridFilter**: Filter operations that traverse all cells in a grid with a window and assign a value for each window position based on a function. This includes smoothing filters, edge detection filters, image enhancement filters, etc.

C.4.12 Connectivity

Connectivity operations accumulate values as they traverse over a feature or a set of connected features [2].

These operations are applied on object features as well as on grid coverages. In the latter case they are typically implemented by filters based on neighbourhoods. This combination of characteristics is the reason why they appear differently in various classifications. For example, some of their subclasses are placed under neighbourhood operations ([43]), or connectivity operations ([12]), or 'iterative operations' ([50]).

- **Spread** The spread operation calculates the accumulated value of a function from one or more source locations in all directions. It can be used to identify drainage points.
- **Seek** A seek operation performs a directed search outward in a step-by-step manner from a start location using a specified decision rule [12]. Typically it uses a 'friction' service to calculate the path of least resistance from a source location to a 'lowest' point.
- **Network** A network is a set of interconnected linear features that form a pattern or framework. Typically network operations are used for utility and traffic analysis. The following operations are identified [12]:
  - **LoadPrediction**: This operation calculates the quantity that 'flows' through specific parts of the network.
  - **RouteOptimisation**: This operation finds the optimal route between two points. 'Optimal' may be 'shortest', 'fastest', or constrained by any other criterion.
  - **ResourceAllocation**: This operation partitions a network into service zones that are assigned to specified target positions.
- **Viewshed**: A viewshed operation (also called intervisibility determines the locations that are within the unobstructed line-of-sight of a viewing position [12].

C.4.13 Interpolation

Interpolation is defined in mathematics as the calculation of the value of a function between the values already known. In a geo-information context this involves two cases (expressed in terms of ISO 19107 (Spatial)):

- **InterGeoObject**: The interpolation operation calculates a number of DirectPositions between known DirectPositions (so called control points) as representations of a geometric object such as a curve or surface.

[2]this characterisation is adapted from [12]

- **InterAttribute**: The interpolation operation calculates the value of a feature attribute at a DirectPosition between DirectPositions already known.

Interpolation operations are distinguished by their method (linear, cubic spline, etc.) A non-exhaustive list of methods is given in [129]. Extrapolation is considered to be similar to interpolation, with the difference that the calculated DirectPosition is lies not *between* but *outside* the known DirectPositions.

C.4.14 Geometric transformation

Geometric transformations change the coordinates of all DirectPositions contained by all geometric objects of a feature type. This may involve the change of the geometric objects' coordinate reference system. This classification does not follow the transformation classification of Chrisman, which does not clearly distinguish with many of the other operations described. The following transformations are identified in this thesis:

- **GeoObjectChange** These operations change the geometric objects that represent feature types into another geometric object type. In the context of ISO 19125 (SFeature) and ISO 19107 (Spatial) this may involve any geometric object type as defined in these standards. For example, in the context of ISO 19107 (Spatial), an operation of this type may convert a Digital Elevation Model (DEM) into a Triangular Irregular Network (TIN). In addition, this operation type involves transformations between object and grid models:
  - **ObjectToGrid**: These operations convert a data set containing object features to a grid coverage. They are commonly known as 'vector to raster' operations or 'rasterisation'
  - **GridToObject**: These operations convert a grid coverage to a data set containing object features, commonly known as 'raster to vector' operation or 'vectorisation'. Mechanisms for vectorisation and rasterisation are not covered by the ISO 19100 standards. They are described in various other literature resources, for example [226, 290].
- **Resampling**: A resampling operation transforms the attribute values of a source grid to those in a target grid. The change of grid may be due to a resolution change, rectification, coordinate reference system change, etc. A resampling operation includes an interpolation mechanism.
- **ChangeCRS**: A change in coordinate reference system my involve a coordinate conversion or transformation. A coordinate *conversion* is defined by ISO 19111 (RefCoord) [130] as 'a one-to-one mapping of coordinates based on one coordinate reference system to another coordinate reference system on the same datum'. A coordinate *transformation* changes coordinates to another datum [130]. Subclasses are:

  - **CoordConversion** (Datum is not changed)
  - **CoordTrans** (Change datum)

C.4.15 Temporal

These operations create, delete or change temporal attributes of features. Examples of temporal operations are change detection, temporal reference system transformation, etc. (see ISO 19119 (Services) [132] for a short list of services in which these operations are used. Note that temporal information can also be stored as thematic attribute. In that case the operations of type *Among thematic attributes* apply.

C.5 Feature presentation manipulation

- **CartoSymbolEdit**: Creates or edits a cartographic symbol.
- **CartoSymbolAssign**: Assigns cartographic symbols to features.

C.6 Service creation and management

- **ChainEdit**: Operations that support the creation or editing of a chain of existing services/operations.
- **ChainValidation**: Operations that validate the chaining of service/operation components (also referred to as 'matchmaking'). These operations may use the *QueryMetaInfo* operation.
- **ServiceDoc**: Operations that extract meta-information from a service, such as method names and parameter names.

C.7 Service execution

These operations are, among others, used by workflow enactment services:

- **BuildRequest**: Operation that builds a service request from a service description (e.g. a SOAP request from a WSDL document).
- **ServiceExecute**: Executes a service or service chain
- **ServiceSchedule**: Schedules a service chain across servers.
- **ServiceOrderHandling**: Operation that involves the handling ordering of a service or service chain (subscription, quoting, status querying, billing, etc.).

- **EventNotification**: Operation that notifies clients of a certain event, such as the moment a service or data set becomes available.

C.8 MetaInfo creation and storage management

- **MetaInfoManagement**: Operations that involve the management of meta-data stores (authentication, query optimisation, etc.).
- **AnnotateMetaInfo**: Operations that support the annotation of meta-information to features, data sets, or services, that cannot be extracted automatically.
- **PublishMetaInfo**: Operations that allow users to publish the meta-information of their data and services.

C.9 MetaInfo processing

- **QueryMetaInfo**: Operations that query meta-information. These operations are used for searching data or service catalogs/registries. The latter may be cascaded. Subclasses are:
  - **QueryFeatureMetaInfo**: Querying information about features in data sets and service
  - **QueryFeatureMetaMetaInfo**: Querying information about schema's, information models, feature ontologies, etc.

C.10 MetaInfo presentation manipulation

- **MetaInfoStyleEdit**: Creates or edits a presentation style for meta-information.
- **MetaInfoStyleAssign**: Assigns a presentation style to meta-information.

Appendix D

OPERA-R I/O parameters for feature processing operations

The table below lists the input and output parameter types of the OPERA operations, which model is described in Section 5.1.1 and which classification is described in Appendix C. These parameter types are coded as constraints in the OWL implementation of the OPERA ontology. The table columns read as follows.

- Operation: The name of the operation as appears in OPERA. An indentation of the name indicates a subclass.
- Application scope: An indication whether the operation's input-output pair can be of one of the following data structure types: O = Object feature, G = Grid, C = Grid cell coverage. Combinations are possible. For example, 'OG' means that the operation has an object feature input-output pair or a grid coverage input-output pair. A 'T' indicates a transformation between any of the three types, which is for example the case for the operations ObjectToGrid and GridToObject.
- Input/output parameter types: The is the name of either of the following:
  - An 'OP' class (e.g., *OP_Point*) as described in the operation part of Section 5.1.1 or a class at the meta-level of the feature symbol ontology (e.g., *GF_ThematicAttributeType*). The indication of such a metaclass means that the operation type may have any parameter type that instantiates that metaclass.
  - Another parameter type, such as Distance or SelectedFeature. Parameters come along with their data type, indicated after a full colon.

Section 5.5.3 provides more background on the meaning of input and output parameter types.

Operation	Application scope	Input parameter types	Output parameter types
FeatureInstantiation			
CreateFeatureInstance	OG	InstanceValueArray : String InstanceValueArray : String	GF_FeatureType AND AnyProperty
CreatePropertyInstance	OG	InstanceValueArray : String	GF_PropertyType
AcrossAttributeTypes			
SpatLoc	OC	GF_SpatialAttributeType	GF_LocationAttributeType
LocSpat	OC	GF_LocationAttributeType	GF_SpatialAttributeType
AmongThematicAttributes			
Group, Classify, Rank, Scale, Evaluate	OGC	GF_ThematicAttributeType AND DomainOfValues	GF_ThematicAttributeType AND DomainOfValues
Separate	OGC	GF_ThematicAttributeType AND DomainOfValues	GF_ThematicAttributeType AND DomainOfValues
Calculate	OGC	GF_ThematicAttributeType	GF_ThematicAttributeType
CrossConcatenate	OGC	GF_ThematicAttributeType AND DomainOfValues	GF_ThematicAttributeType AND DomainOfValues
CrossCalculate, Proportion	OGC	GF_ThematicAttributeType	GF_ThematicAttributeType
FeatureSelection			
SelectByThemQuery	OC	GF_ThematicAttributeType	SelectedFeature : List
SelectByTopo	O	GeometricObject OR GF_SpatialAssociationType with TopologicalAsociationRole OR TopologicalObject	SelectedFeature : List SelectedFeature : List SelectedFeature : List
GetFeaturePropertyValues			
GetAttributeValues	OGC	GF_AttributeType	GF_AttributeType
GetAssociations	OGC	GF_AssociationRole	GF_AssociationRole
GetBehaviour	OGC	GF_Operation	GF_Operation
GeoObjectChange			
DeleteGeoPart, MoveGeoPart	O	GeometricObject (not primitive)	GeometricObject (not primitive)
MoveGeoWhole, SimplifyGeo, GeneraliseGeo	O	GeometricObject	GeometricObject
MergeGeo	O	GeometricObject	GeometricObject
Topology			
Touches, Within, Contains, Crosses, Overlaps, Disjoint, Equals, Intersects, Relate	O O	GeometricObject OR (GeometricObject AND GF_SpatialAssociationType with TopologicalAssociationRole) OR	TopoTestResult : Boolean TopoTestResult : Boolean

	O	TopologicalObject	TopoTestResult : Boolean
CreateTopo	O O	GeometricObject GeometricObject	GF_SpatialAssociationType with TopologicalAsociationRole OR TopologicalObject
MetricMeasurement			
MatchGeo	O	GeometricObject	(MatchTestResult : Boolean) OR GeometricObject
DistanceMeasure	O	GeometricObject	Distance : Float
BearingMeasure	O	ISO19107: GM_Point	Bearing : Float
VolumeMeasure	O	ISO 19107: GM_ Solid OR GM_MultiSolid	Volume : Float
LengthMeasure	O	ISO19125: Curve OR MultiCurve OR ISO19107: GM_GenericCurve OR GM_MultiCurve	Length : Float
AreaMeasure	O	ISO19125: Surface OR MultiSurface OR ISO19107: GM_GenericSurface OR GM_MultiSurface OR GM_Solid OR GM_MultiSolid	Area : Float
PerimeterMeasure	O	ISO19107: GM_GenericSurface OR GM_MultiSurface	Perimeter : Float
Overlay			
ObjectOverlay (Intersection, Union, Difference, SymDifference)	O	GeometricObject AND GF_ThematicAttributeType	GeometricObject AND GF_ThematicAttributeType
GridOverlay	GC	GF_ThematicAttributeType	GF_ThematicAttributeType
DistanceBased			
EqualDistance	OC	GeometricObject	GeometricObject AND GF_SpatialAssociationType with DistanceAssociationRole
FixedDistance	OC	GeometricObject	GeometricObject AND GF_SpatialAssociationType with DistanceAssociationRole
Nearest	OC	GeometricObject	SelectedFeature : List
Neighbourhood (CalculateSlope, CalculateAspect, GridFilter)	OGC	GF_ThematicAttributeType AND (GeometricObject OR (GF_SpatialAssociationType with TopologicalAssociationRole or DistanceAssociationRole) OR TopologicalObject)	GeometricObject AND GF_ThematicAttributeType

Connectivity			
Spread, Seek	C	GF_ThematicAttributeType AND (GeometricObject OR (GF_SpatialAssociationType with TouchesAssociationRole) OR TopologicalObject)	GeometricObject AND GF_ThematicAttributeType
Network			
LoadPrediction, ResourceAlloocation	O	GF_ThematicAttributeType AND Line AND ((GF_SpatialAssociationType with TouchesAssociationRole) OR TopologicalObject)	Line AND GF_ThematicAttributeType
RouteOptimisation	O	MultiLineString AND ((GF_SpatialAssociationType with TouchesAssociationRole) OR TopologicalObject)	LineString
Viewshed	C	GeometricObject GF_ThematicAttributeType	GeometricObject GF_ThematicAttributeType
Interpolation			
InterGeoObject	O	DirectPosition GeometricObject	DirectPosition
InterAttribute	OGC	DirectPosition GF_ThematicAttributeType	GF_ThematicAttributeType
GeoTransformation			
GeoObjectChange			
ObjectToGrid	T	Object feature	Grid coverage
GridToObject	T	Grid coverage	Object feature
Resampling	G	GeometricObject GF_ThematicAttributeType	GeometricObject GF_ThematicAttributeType
ChangeCRS	OGC	CoordinateReferenceSystem (SC_CRS) CoordinateSet	CoordinateSet
Temporal			
ActOnTemporalAttributeInstance : Sub class of ActOnPropertyInstance	OG		

Table D.1: OPERA-R I/O parameters for feature processing operations.

Appendix E

ADL Gazetteer OWL service description

This appendix contains the OWL code of the Alexandria Library (ADL) Gazetteer service, which is used as example geo-service in several parts of the thesis, for example in Sections 1.1, 5.6.1 and 6.1.

```
<?xml version="1.0"?> <rdf:RDF
    xmlns:serv="http://geoserver.itc.nl/lemmens/owl/implem.owl#"
    xmlns:symbol="http://geoserver.itc.nl/lemmens/owl/symbol.owl#"
    xmlns:expr="http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#"
    xmlns:nen3610="http://geoserver.itc.nl/lemmens/owl/nen3610.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:refont="http://geoserver.itc.nl/lemmens/owl/refont.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:opera="http://geoserver.itc.nl/lemmens/owl/opera.owl#"
    xmlns:concept="http://geoserver.itc.nl/lemmens/owl/concept.owl#"
    xmlns:ontogeo="http://geoserver.itc.nl/lemmens/owl/ontogeo-v3.0.owl#"
    xmlns:top10nl="http://geoserver.itc.nl/lemmens/owl/top10nl.owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:riskmap="http://geoserver.itc.nl/lemmens/owl/riskmap.owl#"
    xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
    xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
    xmlns:list="http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:time="http://www.isi.edu/~pan/damltime/time-entry.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:travel="http://geoserver.itc.nl/lemmens/owl/travel.owl#"
    xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
    xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://geoserver.itc.nl/lemmens/owl/adlgazetteer-service.owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xml:base="http://geoserver.itc.nl/lemmens/owl/adlgazetteer-service.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://geoserver.itc.nl/lemmens/owl/ontogeo-v3.0.owl"/>
  </owl:Ontology>
  <opera:LocSpat rdf:ID="_ADLGazOperation">
    <opera:hasInputPar>
```

```
      <opera:InputPar rdf:ID="ADLGazInputPar_1">
        <opera:hasParType rdf:resource="http://geoserver.itc.nl/lemmens/owl/ontogeo-v3.0.owl#AddressType"/>
      </opera:InputPar>
    </opera:hasInputPar>
    <opera:hasOutputPar>
      <opera:OutputPar rdf:ID="ADLGazOutputPar_1">
        <opera:hasParType rdf:resource="http://geoserver.itc.nl/lemmens/owl/ontogeo-v3.0.owl#PointType"/>
      </opera:OutputPar>
    </opera:hasOutputPar>
  </opera:LocSpat>
  <serv:GeoService rdf:ID="_ADLGazService">
    <serv:makesAvailable rdf:resource="#_ADLGazOperation"/>
  </serv:GeoService>
</rdf:RDF>
```

Appendix F

ADL Gazetteer WSDL service description

This WSDL file was generated by Carlos Granell in a joint project with the author (see for more information Sections 7.1.1 and 8.3.4).

```
<?xml version="1.0" encoding="UTF-8" ?>
 <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/ADLGazClient"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:tns="http://localhost:8080/axis/services/ADLGazClient"
xmlns:xsd1="http://localhost:8080/axis/services/ADLGazClient.xsd"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">


<wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://localhost:8080/axis/services/ADLGazClient.xsd">
    <complexType name="tCoordinates">
    <sequence>
<element name="latitude" type="xsd:double"/> <element
name="longitude" type="xsd:double"/> </sequence> </complexType>
<element name="RequestType" type="xsd:string"/> <element
name="ResponseType" type="xsd1:tCoordinates"/> </schema>
</wsdl:types>


 <!--
WSDL created by Apache Axis version: 1.3 Built on Oct 05, 2005
(05:23:37 EDT) -->
 <wsdl:message name="getCoordinatesResponse">
<wsdl:part name="output" element="xsd1:ResponseType" />
</wsdl:message>
 <wsdl:message name="getCoordinatesRequest">
<wsdl:part name="name" element="xsd1:RequestType" /> </wsdl:message>



 <wsdl:portType name="ADLGazClient">
 <wsdl:operation name="getCoordinates">
<wsdl:input message="tns:getCoordinatesRequest"
name="getCoordinatesRequest" /> <wsdl:output
message="tns:getCoordinatesResponse" name="getCoordinatesResponse"
/> </wsdl:operation> </wsdl:portType>


 <wsdl:binding name="ADLGazClientSoapBinding" type="tns:ADLGazClient">
<wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
 <wsdl:operation name="getCoordinates">
<wsdlsoap:operation soapAction="" />
 <wsdl:input name="getCoordinatesRequest">
<wsdlsoap:body
namespace="http://localhost:8080/axis/services/ADLGazClient"
use="literal" /> </wsdl:input>

 <wsdl:output name="getCoordinatesResponse">
<wsdlsoap:body
namespace="http://localhost:8080/axis/services/ADLGazClient"
use="literal" /> </wsdl:output> </wsdl:operation>


</wsdl:binding>
 <wsdl:service name="ADLGazClientService">
 <wsdl:port binding="tns:ADLGazClientSoapBinding" name="ADLGazClient">
<wsdlsoap:address
location="http://localhost:8080/axis/services/ADLGazClient" />
</wsdl:port> </wsdl:service> </wsdl:definitions>
```

Appendix G

ISO 19119 mapping

This appendix provides all ontology mappings between ISO 19119 (Services) classes and OPERA classes. These mappings are explained in Section 6.3.1. In the list below the mappings are grouped according to the ISO 19119 classification of services. Some mappings make use of the shorthand notation for control flow patterns as introduced in Section 4.7 in a way that is explained in Section 6.3.1.

Geographic human interaction services

$\mathit{iso19119{:}CatalogueViewer} \sqsubseteq \mathit{opera{:}MetaDataInteract}$

$\mathit{iso19119{:}GeographicViewer} \sqsubseteq \mathit{opera{:}MapDisplay}$

$\mathit{iso19119{:}GeographicSpreadsheetViewer} \sqsubseteq \mathit{opera{:}FeaturePropertyDisplay}$

$\mathit{iso19119{:}ServiceEditor} \sqsubseteq$
  $\mathit{opera{:}InvokeOperation} \sqcap$
  $(\exists\forall\ \mathit{opera{:}invokesOperation.opera{:}ServiceOperation}) \sqcap$
  $(\exists\forall\ \mathit{opera{:}invokesOperation.}($
    $\exists\forall\ \mathit{opera{:}ActsOnServiceType.iso19119{:}GeographicProcessing}))$

$\mathit{iso19119{:}ChainDefinitionEditor} \sqsubseteq$
  $\mathit{opera{:}InvokeOperation} \sqcap$
  $(\exists\forall\ \mathit{opera{:}invokesOperation.opera{:}ServiceEdit})$

$\mathit{iso19119{:}WorkflowEnactmentManager} \sqsubseteq$
  $\mathit{opera{:}InvokeOperation} \sqcap$

$(\exists\forall\ \mathit{opera{:}invokesOperation.ServiceExecution})$

$\mathit{iso19119{:}GeographicFeatureEditor} \sqsubseteq$
$(\mathit{opera{:}MapDisplay} \sqcup \mathit{opera{:}AnnotateMetaInfo})$

$\mathit{iso19119{:}GeographicSymbolEditor} \sqsubseteq$
$\mathit{opera{:}InvokeOperation} \sqcap$
$(\exists\forall\ \mathit{opera{:}invokesOperation.opera{:}FeatureDataPresentation})$

$\mathit{iso19119{:}FeatureGeneralizationEditor} \sqsubseteq$
$\mathit{opera{:}InvokeOperation} \sqcap$
$(\exists\forall\ \mathit{opera{:}invokesOperation.opera{:}GeneraliseGeo})$

$\mathit{iso19119{:}GeographicDataStructureViewer} \sqsubseteq$
$\mathit{opera{:}InvokeOperation} \sqcap$
$(\exists\forall\ \mathit{opera{:}invokesOperation.}(\mathit{opera{:}Extractmetainfo} \sqcup \mathit{opera{:}Extractfeature}))$

Geographic model information management services

$\mathit{iso19119{:}FeatureAccessService} \sqsubseteq$
$\mathit{opera{:}ExtractFeature} \sqcap$
$(\exists\forall\ \mathit{opera{:}appliesToDataStruc.symbol{:}ObjectFeature})$

$\mathit{iso19119{:}MapAccessService} \sqsubseteq \mathit{opera{:}ExtractMap}$

$\mathit{iso19119{:}CoverageAccessService} \sqsubseteq$
$\mathit{opera{:}ExtractMap} \sqcap$
$(\exists\forall\ \mathit{opera{:}appliesToDataStruc.symbol{:}Coverage})$

$\mathit{iso19119{:}CoverageAccessServiceSensor} \sqsubseteq \mathit{iso19119{:}CoverageAccessService}$

$\mathit{iso19119{:}SensorDescriptionService} \sqsubseteq \mathit{opera{:}QueryMetaInfo}$

$\mathit{iso19119{:}ProductAccessService} \sqsubseteq \mathit{opera{:}DataSourceManagement}$

$\mathit{iso19119{:}FeatureTypeService} \sqsubseteq \mathit{opera{:}FeatureModelling}$

$\mathit{iso19119{:}CatalogueService} \sqsubseteq \mathit{opera{:}QueryFeatureMetaInfo}$

$\textit{iso19119:RegistryService} \sqsubseteq \textit{opera:QueryFeatureMetaMetaInfo}$

$\textit{iso19119:GazetteerService} \sqsubseteq \textit{opera:LocSpat}$

$\textit{iso19119:OrderHandlingService} \sqsubseteq \textit{opera:ServiceOrderHandling}$

$\textit{iso19119:StandingOrderService} \sqsubseteq \textit{opera:ServiceOrderHandling}$

Geographic workflow task management services

$\textit{iso19119:ChainDefinitionService} \sqsubseteq \textit{opera:ChainEdit}$

$\textit{iso19119:WorkflowEnactmentService} \sqsubseteq$
$(\textit{opera:ServiceExecute} \sqcup \textit{opera:ServiceSchedule})$

$\textit{iso19119:SubscriptionService} \sqsubseteq \textit{opera:EventNotification}$

Geographic processing services

Geographic processing services - Spatial

$\textit{iso19119:CoordinateConversionService} \sqsubseteq \textit{opera:CoordConversion}$

$\textit{iso19119:CoordinateTransformationService} \sqsubseteq \textit{opera:CoordTrans}$

$\neg\,((\textit{iso19119:CoverageVectorConversionService} \sqcap \textit{opera:GeoObjectChange}) \sqsubseteq \bot)$

$\textit{iso19119:ImageCoordinateConversionService} \sqsubseteq \textit{iso19119:CoordinateConversionService}$

$\textit{iso19119:Rectification} \sqsubseteq \textit{opera:Resampling}$

$\textit{iso19119:OrthoRectification} \sqsubseteq \textit{iso19119:Rectification}$

$\textit{iso19119:SensorGeometryModelAdjustmentService} \sqsubseteq \textit{opera:Resampling}$

$\textit{iso19119:ImageGeometryModelConversionService} \sqsubseteq \textit{opera:Resampling}$

---

iso19119:SubSettingServiceSpatial $\sqsubseteq C$
in which C is an ontological concept that describes the control flow pattern below:

```
<Iterate>
    opera:ExtractGeoInfoFromStream
    opera:SelectByTopo
</Iterate>
```

---

iso19119:SamplingServiceSpatial $\sqsubseteq C$
in which C is an ontological concept that describes the control flow pattern below:

```
<Iterate>
    opera:ExtractGeoInfoFromStream
    opera:SelectByTopo
</Iterate>
```

---

iso19119:TilingChangeService $\sqsubseteq$ *opera:DataSourceManagement*

---

iso19119:DimensionMeasurementService $\sqsubseteq$ *opera:SpatialMeasurement*

---

iso19119:FeatureManipulationService $\sqsubseteq$ *opera:GeometricObjectChange*

---

iso19119:FeatureMatchingService $\sqsubseteq$ *opera:MatchGeo*

---

iso19119:FeatureGeneralizationServiceSpatial $\sqsubseteq$ *opera:GeneraliseGeo*

---

iso19119:RouteDeterminationService $\sqsubseteq$ *opera:RouteOptimisation*

---

iso19119:PositioningService $\sqsubseteq$ *opera:ExtractGeoInfoFromStream*

---

iso19119:ProximityAnalysisService $\sqsubseteq C$
in which C is an ontological concept that describes the control flow pattern below:

```
<Sequence>
    opera:FixedDistance
    opera:SelectByWithin
</Sequence>
```

Geographic processing services - Thematic

---

iso19119:GeoparameterCalculationService $\sqsubseteq$ *opera:AmongThematicAttributes*

---

iso19119:ThematicClassificationService $\sqsubseteq$ *opera:AmongFeatureProperties*

---

iso19119:FeatureGeneralizationServiceThematic $\sqsubseteq$ *opera:AmongFeatureProperties*

---

iso19119:SubsettingServiceThematic $\sqsubseteq$ *opera:SelectByThemQuery*

---

iso19119:SpatialCountingService $\sqsubseteq$ C
in which C is an ontological concept that describes the control flow pattern below:

```
<Sequence>
    opera:SelectByWithin
    opera:Count
</Sequence>
```

---

iso19119:GeographicInformationExtractionServices $\sqsubseteq$ *opera:EctractObjectInfo*

---

iso19119:ImageProcessingService $\sqsubseteq$ *opera:GridFilter*

---

iso19119:ReducedResolutionGenerationService $\sqsubseteq$ *opera:Resampling*

---

iso19119:ImageManipulationServices $\sqsubseteq$
(*opera:GridFilter* $\sqcup$ *opera:AmongFeatureProperties*)

---

iso19119:ImageUnderstandingServices $\sqsubseteq$
(*opera:GridFilter* $\sqcup$ *opera:ExtractObjectInfo*)

---

iso19119:ImageSynthesisServices $\sqsubseteq$
(*iso19119:ImageProcessingService* $\sqcup$ *iso19119:ImageManipulationServices*)

---

iso19119:MultiBandImageManupilation $\sqsubseteq$ *iso19119:ImageManipulationServices*

---

iso19119:ObjectDetectionService $\sqsubseteq$ *opera:EctractObjectInfo*

---

iso19119:GeoParsingService $\sqsubseteq$ *opera:EctractObjectInfo*

---

iso19119:GeocodingService $\sqsubseteq$ *opera:LocSpat*

Geographic processing services - Temporal

---

iso19119:ChangeDetectionServices $\sqsubseteq$
(*iso19119:ImageUnderstandingServices* $\sqcap$ *opera:Temporal*)

---

iso19119:TemporalReferenceSystemTransformationService $\sqsubseteq$ *opera:Temporal*

---

iso19119:SubsettingServiceTemporal $\sqsubseteq$ C
in which C is an ontological concept that describes the control flow pattern below:

```
<Iterate>
    opera:ExtractGeoInfoFromStream
    opera:SelectByTemporalAttribute
</Iterate>
```

---

iso19119:SamplingServiceTemporal $\sqsubseteq$ C
in which C is an ontological concept that describes the control flow pattern below:

```
<Iterate>
    opera:ExtractGeoInfoFromStream
    opera:SelectByTemporalAttribute
</Iterate>
```

---

iso19119:TemporalProximityAnalysisService $\sqsubseteq$ *opera:Temporal*

Geographic processing services - Metadata

---

iso19119:StatisticalCalculationService $\sqsubseteq$ *opera:ExtractMetaInfo*

---

iso19119:GeographicAnnotationService $\sqsubseteq$ *opera:AnnotateMetaInfo*

Geographic communication services

---

iso19119:EncodingService $\sqsubseteq$ *opera:Coding*

---

iso19119:TransferService $\sqsubseteq$ *opera:DataSourceAccess*

---

iso19119:GeographicCompressionService $\sqsubseteq$ *opera:FormatConversion*

---

iso19119:GeographicFormatConversionService $\sqsubseteq$ *opera:FormatConversion*

---

iso19119:MessagingService $\sqsubseteq$ *opera:DataSourceManagement*

---

iso19119:RemoteFileAndExecutableManagement $\sqsubseteq$ *opera:DataSourceManagement*

Bibliography

[1] Aalst, W.M.P. van der, Hofstee, A.H.M. ter, Kiepuszewski, B., and Barros, A. Workflow Patterns. *Distributed and Parallel Databases* (2003), 5–51.

[2] Aduna. *AutoFocus User's Guide*, 2005. release 2005.1.

[3] Agarwal, P. Ontological considerations in GIScience. *International Journal of Geographical Information Science 19* (2005), 501–536.

[4] Akkiraju, R., Farrell, J., J.Miller, Nagarajan, M., Schmidt, M., Sheth, A., and Verma, K. Web Service Semantics - WSDL-S. A joint UGA-IBM Technical Note version 1.0, April 2005. Available at http://lsdis.cs.uga.edu/projects/METEOR-S/WSDL-S.

[5] Albertoni, R., Bertone, A., and Martino, M. D. Semantic web and information visualization.

[6] Albrecht, J. *Universelle GIS-Operationen.* PhD thesis, Fachbereich Sozial- und Kulturwissenschaften der Hochschule Vechta, 1995.

[7] Albrecht, J. Universal analytical GIS operations: a task-oriented systematization of data structure-independent GIS functionality. In *Geographic Information Research: transatlantic perspectives.* (1998), M. Craglia and H. Onsrud, Eds., Taylor & Francis, pp. 577–591.

[8] Ankolekar, A., Paolucci, M., and Sycara, K. P. Towards a Formal Verification of OWL-S Process Models. In *Proceedings International Semantic Web Conference* (2005), pp. 37–51.

[9] Antoniou, G., Franconi, E., and van Harmelen, F. Introduction to semantic web ontology languages. In *Reasoning Web, Proceedings of the Summer School, Malta, 2005* (Berlin, Heidelberg, New York, Tokyo, 2005), N. Eisinger and J. Małuszyński, Eds., no. 3564 in Lecture Notes in Computer Science, Springer-Verlag.

[10] Antoniou, G., and van Harmelen, F. Web Ontology Language: OWL. In *Handbook on Ontologies* (2004), S. Staab and R. Studer, Eds., Springer, pp. 67–92.

[11] Arlow, J., and Neustadt, I. *UML 2 and the Unified Process.* Addison-Wesley, 2005.

[12] Aronoff, S. *Geographic Information systems: A Management Perspective.* WDL Publications, Ottawa, Canada, 1993.

[13] Arpinar, I. B., Sheth, A., Ramakrishnan, C., Usery, E. L., Azami, M., and Kwan, M.-P. Geospatial Ontology Development and Semantic Analytics. In *Handbook of Geographic Information Science,*, J. P. Wilson and A. S. Fotheringham, Eds. Blackwell Publishing, 2004.

[14] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., Eds. *The Description Logic Handbook; Theory, Implementation and Applications.* Cambridge University Press, 2003.

[15] Baader, F., Horrocks, I., and Sattler, U. Description logics. In *Handbook on Ontologies*, S. Staab and R. Studer, Eds. Springer, 2004, pp. 3–28.

[16] Baader, F., and Nutt, W. Basic Description Logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge University Press, 2003, pp. 43–95.

[17] Bakker, N., Bruns, B., and Storm, M. Gegevensmodel TOP10NL. Tech. Rep. Version 2.3 (IM7361), Topografische Dienst Kadaster/ International Institute for Geo-Information Science and Earth Observation, February 2005. (In Dutch).

[18] Balzer, S., Liebig, T., and Wagner, M. Pitfalls of OWL-S A Practical Semantic Web Use Case. In *ICSOC04, November 1519, 2004, New York, New York, USA.* (2004).

[19] Barstow, A., Skall, J. H. M., Pollock, J., Martin, D., Marcatte, V., McGuinness, D. L., Yoshida, H., and Roure, D. D. OWL-S ontologies. World Wide Web Consortium web site, November 2004. Available at http://www.w3.org/Submission/2004/07/.

[20] Bass, L., Clements, P., and Kazman, R. *Software Architecture in Practice*, second ed. Addison-Wesley, 2003.

[21] Bechhofer, S. The DIG Description Logic Interface: DIG/1.1. In *Proceedings of DL2003 Workshop, Rome, Italy* (June 2003).

[22] Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. OWL Web Ontology Language Reference. W3c recommendation (editors: Mike dean and guus schreiber), World Wide Web Consortium, February 2004. Available at http://www.w3.org/TR/owl-ref/. Accessed December 2005.

[23] Beckett, D. RDF/XML Syntax Specification (Revised). W3C Recommendation, W3C, February 2004.

[24] Berardi, D., Gruninger, M., Hull, R., and McIlraith, S. Towards a first-order ontology for web services. In *W3C Workshop on Constraints and Capabilities for Web Services* (October 2004).

[25] Berneres-Lee, T. A parenthetical discussion to the Web Architecture at 50,000 feet and the Semantic Web roadmap. http://www.w3.org/DesignIssues/RDFnot.html, September 1998. Accessed January 2006.

[26] Berners-Lee, T., and Fischetti, M. *Weaving the Web*, first ed. Harper San Francisco, 1999.

[27] Berners-Lee, T., Hendler, J., and Lassila, O. The semantic web. *Scientific American May* (2001).

[28] Bernstein, A., and Klein, M. Discovering services: Towards High-Precision Service Retrieval. In *Proceedings International Workshop on Web Services, E-Business, and the Semantic Web* (May 2002), Lecture Notes In Computer Science.

[29] Bishr, Y. *Semantic aspects of Interoperable GIS.* PhD Thesis, ITC publication number 56, International Institute for Geo-information Science and Earth Observation, Enschede, the Netherlands, 1997.

[30] Bishr, Y. Overcoming the semantic and other barriers to GIS interoperability. *int. j. geographical information science 12*, 4 (1998), 299–314.

[31] Bittner, T., Donnelly, M., and Winter, S. Ontology and Semantic Interoperability. In *Large scale 3D data integration: problems and challenges*, D. Prosperi and S. Zlatanova, Eds. September 2005.

[32] Bock, C., and Gruninger, M. PSL: A Semantic Domain for Flow Models. *Software and Systems Modeling Journal, 4* (2005), 209–231.

[33] Bolognesi, T., and Brinksma, E. Introduction to the ISO specification language LOTOS. *Comput. Netw. ISDN Syst. 14*, 1 (1987), 25–59.

[34] Borgida, A., and Brachman, R. J. Conceptual Modeling with Description Logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge University Press, 2003, pp. 1–40.

[35] Borgida, A., Mylopoulos, J., and Wong, H. K. T. Generalization/Specialization as a Basis for Software Specification. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages* (1984), M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Eds., Springer, pp. 87–117.

[36] Bossard, M., Feranec, J., and Otahel, J. CORINE land cover technical guide Addendum. Tech. Rep. 40, European Environment Agency, May 2000.

[37] Bouquet, P., Serafini, L., and Zanobini, S. Semantic Coordination: A New Approach and an Application. In *Proceedings The SemanticWeb - ISWC 2003 Second International SemanticWeb Conference* (Sanibel Island, FL, USA, 2003), J. M. Dieter Fensel, Katia Sycara, Ed., vol. Lecture Notes in Computer Science Volume 2870 / 2003, Springer-Verlag Heidelberg.

[38] Bowers, S., and Ludäscher, B. An ontology-driven framework for data transformation in scientific workflows. In *Proceedings of the International Workshop on Data Integration in the Life Sciences (DILS'04), March 25-26, 2004 Leipzig, Germany.* (2004), Springer, Ed., vol. 2994 of *LNCS*.

[39] Brewer, D. *Radiant annotation tool for WSDL-S.* LSDIS and the University of Georgia. On-line documentation; available at http://lsdis.cs.uga.edu/projects/meteor-s/downloads/OnlineDoc/.

[40] Brodeur, J., and Bédard, Y. Extending geospatial repositories with geosemantic proximity functionalities to facilitate the interoperability of geospatial data. In *Symposium on Geaspatial Theory, Processing and Applications* (Ottawa, 2002).

[41] Brodie, M. The promise of distributed computing and the challenges of legacy information systems. In *Advanced Database Systems: Proceedings of the 10th British National Conference on Databases* (1992), P. Gray and R. Lucas, Eds., Springer Verlag (Heidelberg, FRG).

[42] Bucher, B. Integrating structured descriptions of processes in geographical metadata. In *Developments in Spatial Data Handling: 11th International Symposium on Spatial Data Handling* (2004), P. F. Fisher, Ed., Springer.

[43] By, R.A. de, Georgiadou, Y., Knippers, R. A., Kraak, M.-J., Sun, Y., Weir, M. J., and van Westen, C. J. *Principles of Geographic Information Systems - An introductory textbook*, 3rd ed. ITC Educational

Textbook Series; 1. International Institute for Geo-Information science and Earth Observation (ITC), 2004.

[44] Cabral, L., Domingue, J., Motta, E., Payne, T., and Hakimpour, F. Approaches to Semantic Web Services: An Overview and Comparisons. In *Proceedings First European Semantic Web Symposium, Heraklion, Greece* (May 2004).

[45] Calvanese, D., De Giacomo, G., and Lenzerini, M. Ontology of Integration and Integration of Ontologies. In *Proceedings of the 2001 Description Logic Workshop (DL 2001)* (2001), CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-49/, pp. 10–19.

[46] Caprotti, O., Dewar, M., and Turi, D. Mathematical Service Matching Using Description Logic and OWL. In *Proceedings Mathematical Knowledge Management, Third International Conference, MKM 2004, Bialowieza, Poland* (2004), A. Asperti, G. Bancerek, and A. Trybulec, Eds., vol. 3119 of *Lecture Notes in Computer Science*, Springer.

[47] CGDI. Canadian geospatial data infrastructure web portal. http://cgdi-dev.geoconnections.org/prototypes/owsview/. Accessed September 2005.

[48] CGDI Architecture Working Group. Canadian Geospatial Data Infrastructure architecture description, Revision: Version 1. Tech. rep., Canadian Geospatial Data Infrastructure, December 2001.

[49] Chrisman, N. A transformational approach to GIS operations. *int. j. geographical information science 13, no. 7* (1999), 671–637.

[50] Chrisman, N. *Exploring Geographical Information Systems*, second ed. Wiley and Sons, 2002.

[51] Cleaveland, R., and Smolka, S. A. Strategic directions in concurrency research. *ACM Comput. Surv. 28*, 4 (1996), 607–625.

[52] Clementini, E., Di Felice, P., and van Oosterom, P. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Proceedings 3rd International Symposium on Large Spatial Databases, Singapore* (June 1993), no. 692 in LNCS, pp. 277–295.

[53] Cohn, A., and Hazarika, S. Qualitative Spatial Representation and Reasoning : An Overview. *Fundamenta Informaticae 43* (2001), 2–32.

[54] Cornet, R. *RICE (RACER Interactive Client Environment) Manual.* Department of Medical Informatics, University of Amsterdam, April 2004. Technical Report 200301.

[55] Cranefield, S., and Purvis, M. UML as an ontology modelling language. In *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99)* (1999).

[56] Cruz, I. F., and Xiao, H. Using a Layered Approach for Interoperability on the Semantic Web. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE03)* (2003).

[57] Daconta, M. C., Obrst, L. J., and Smith, K. T. *The Semantic Web: A guide to the future of XML, Web Services and Knowledge Management.* Wiley, 2003.

[58] Dahchour, M., Pirotte, A., and Zimányi, E. Definition and Application of Metaclasses. In *Proc. of the 12th Int. Conf. on Database and Expert Systems Applications, DEXA01, Munich, Germany* (2001), H. Mayr, J. Lazansky, G. Quirchmayr, and P. Vogel, Eds., LNCS 2113, Springer-Verlag, pp. 32–41.

[59] DAML Services Coalition. DAML-S: Web Service Description for the Semantic Web. In *Proceedings of The First International Semantic Web Conference (ISWC),Sardinia, Italy* (2002), Springer-Verlag Heidelberg.

[60] Deitel, H., Deitel, P., and Nieto, T. *Internet & World Wide Web: How to Program*, second ed. Prentic Hall, 2002.

[61] Donini, F. M. Compexity of reasoning. In *The Description Logic Handbook: Theory, Implementation, and Applications*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge University Press, 2003, pp. 485–495.

[62] Dou, D., McDermott, D., and Qi, P. Ontology Translation on the Semantic Web. *Journal on Data Semantics II LNCS 3360* (2004), 3557.

[63] Duckham, M., and Worboys, M. An algebraic approach to automated geospatial information fusion. *International Journal of Geographical Information Science 19* (2005), 537–557.

[64] Egenhofer, M. Toward the semantic geospatial web. In *Proceedings ACM GIS* (2002).

[65] Egenhofer, M., and Herring, J. Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical report, Department of Surveying Engineering, University of Maine, 1990.

[66] Einspanier, U., Lutz, M., Senkler, K., Simonis, I., and Sliwinski, A. Toward a Process Model for GI Service Composition. In *Proceedings of Münster GI-days 2003* (2003).

[67] Elenius, D., Denker, G., Martin, D., Gilham, F., Khouri, J., Sadaati, S., and Senanayake, R. The OWL-S Editor A Development Tool forSemantic Web Services. In *Proceedings 2nd Annual European Semantic Web Conference (ESWC), Heraklion, Crete* (June 2005), vol. 3532, Springer.

[68] European Commission - Joint Research Centre. INSPIRE GeoPortal. http://eu-geoportal.jrc.it/. Accessed September 2005.

[69] European Commission - Joint Research Centre. INSPIRE Principles. http://inspire.jrc.it/principles_en.html. Accessed September 2005.

[70] European Commission - Joint Research Centre. INSPIRE Architecture and Standards Position Paper. Inspire ast position paper, JRC-Institute for Environment and Sustainability, Ispra, October 2002.

[71] Fellbaum, C., Ed. *WordNet, An Electronic Lexical Database.* MIT Press, 1998.

[72] Fensel, D., Sycara, K., and Mylopoulos, J., Eds. *The SemanticWeb - ISWC 2003 Second International SemanticWeb Conference Sanibel Island, FL, USA, October 20-23, 2003* (2003), vol. 2870 / 2003 of *Lecture Notes in Computer Science*, Springer-Verlag Heidelberg.

[73] Fikes, R., Hayes, P., and Horrocks, I. OWL-QL A Language for Deductive Query Answering on the Semantic Web. *Journal of Web Semantics 2*, 1 (2004), 19–29.

[74] Fluit, C., Sabou, M., and van Harmelen, F. Supporting user tasks through visualisation of light-weight ontologies. In *Handbook on Ontologies* (2004), S. Staab and R. Studer, Eds., Springer, pp. 415–432.

[75] Fonseca, F., Davis, C., and Câmara, G. Bridging ontologies and conceptual schemas in geographic applications development. *Geoinformatica 7*, 4 (2003), 355–378.

[76] Fonseca, F., Egenhofer, M., Davis, C., and Câmara, G. Semantic Granularity in Ontology-Driven Geographic Information Systems. *Annals of Mathematics and Artificial Intelligence - Special Issue on Spatial and Temporal Granularity 36* (2002), 121–151.

[77] Fonseca, F., Egenhofer, M. J., Agouris, P., and Câmara, G. Using ontologies for integrated geographic information systems. *Transactions in GIS 6*, 3 (2002), 231–257.

[78] Fonseca, F. T., and Egenhofer, M. J. Ontology-driven geographic information systems. *ACM GIS 11* (1999), 14–19.

[79] Foster, I., Alpert, E., Chervenak, A., Drach, B., Kesselman, C., Nefedova1, V., Middleton, D., Shoshani, A., Sim, A., and Williams, D. The Earth System Grid - Turning Climate Datasets into Community Resources. AMS2002. http://dataportal.ucar.edu/esg/docs/ESGAMS_final.pdf, 2002. Accessed September 2005.

[80] Foster, I., Kesselman, C., and Tuecke, S. The Anatomy of the Grid Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications 15*, 3 (August 2001), 200–222.

[81] Foster, I., Middleton, D., and Williams, D. The Earth System Grid (ESG). Presentation at DOE SciDAC PI Meeting, Napa, California, USA. http://dataportal.ucar.edu/esg/docs/DOESciDACNapa_ESG_dist.ppt, March 2003. Accessed September 2005.

[82] Franconi, E. Description Logics, Propositional Description Logics. http://www.inf.unibz.it/~franconi/dl/course/slides/prop-DL/-propositional-dl.pdf, 2002. Description Logics course, module 3, Department of Computer Science, University of Manchester.

[83] Frank, A. U. Ontology for Spatio-temporal Databases. In *Spatiotemporal Databases: The Chorochronos Approach*, M. Koubarakis, Ed., Lecture Notes in Computer Science. Springer-Verlag, 2003, pp. 9–78.

[84] Franke, D., Hayes, P., Kendall, E., and McGuinness, D. The Model Driven Semantic Web. In *1st International Workshop on the Model-Driven Semantic Web (MDSW2004), Monterey, California, USA.* (September 2004).

[85] Garshol, L. M. Living with topic maps and RDF Topic maps, RDF, DAML, OIL, OWL, TMCL. Ontopia White paper. http://www.ontopia.net/topicmaps/materials/tmrdf.html, 2003. Accessed June 2005.

[86] GDAL. GDAL - Geospatial Data Abstraction Library. http://www.gdal.org. Accessed September 2005.

[87] GDMC. Results of the 4th GML Relay at Kadaster Topographical Service, January 26 2006, Emmen, The Netherlands. Available at http://www.gdmc.nl/events/relay4/results.html, 2006. Accessed March 2006.

[88] Genesereth, M. R., and Nilsson, N. J. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, 1987.

[89] Geotools. Geotools. http://www.geotools.org/. Accessed September 2005.

[90] Goble, C., and Roure, D. D. The Grid: An Application of the Semantic Web. *SIGMOD Record 31*, 4 (December 2002).

[91] Gonzalez-Castillo, J., Trastour, D., and Bartolini, C. Description logics for matchmaking of services. In *Workshop on Application of Description Logics* (2001).

[92] Goodchild, M. F. Spatial Analysis and GIS. In *Proceedings ESRI User Conference Pre-Conference Seminar* (2001).

[93] Goodchild, M. F., Egenhofer, M. J., and Fegeas, R. Interoperating GISs. In *Report of specialist meeting Varenius project, Panel on computational implementations of geographic concepts,Santa Barbara, California* (December 1997).

[94] Goodwin, J. Experiences of Using OWL at the Ordnance Survey. In *Proceedings OWL workshop, Galway, Ireland* (November 2005). Avialable at http://www.mindswap.org/2005/OWLWorkshop/.

[95] Goodwin, J. What Have Ontologies Ever Done For Us? Potential Applications at a National Mapping Agency. In *Proceedings OWL workshop, Galway, Ireland* (November 2005). Avialable at http://www.mindswap.org/2005/OWLWorkshop/.

[96] Gottschalk, K., Graham, S., Kreger, H., and Snell, J. Introduction to Web services architecture. *IBM Systems Journal 41*, 2 (2002), 170–177.

[97] Granell, C. *An Integrated Component-based Approach for Improving Service Reuse.* Phd thesis, Universitat Jaume I, Castellón, Spain, 2006. to be published.

[98] Granell, C., Gould, M., Grønmo, R., and Skogan, D. Improving Reuse of Web Service Compositions. In *Proceedings EC-Web 2005 , Copenhagen, Denmark* (August 2005), Lecture Notes in Computer Science 3590, pp. 358–367.

[99] GRASS. Geographic Resources Analysis Support System. http://grass.itc.it/. Accessed September 2005.

[100] Greenwood, J., and Hart, G. Sharing Feature Based Geographic Information - A Data Model Perspective. In *Proceedings of the 7th International Conference on GeoComputation, University of Southampton, United Kingdom* (September 2003).

[101] Grenon, P., and Smith, B. SNAP and SPAN: Towards Dynamic Spatial Ontology. *Spatial Cognition & Computation 4*, 1 (2004), 69–104.

[102] Grothe, M., Landa, H., and Steenbruggen, J. The Value of Gi4DM for Transport and Water Management. In *Geo-information for Disaster Management* (2005), P. van Oosterom, S. Zlatanova, and E. Fendel, Eds., Springer, pp. 129–153.

[103] Gruber, T. Toward principles for the design of ontologies used for knowledge sharing. In *Proceedings International Workshop on Formal Ontology, Padova, Italy* (1993), N. Guarino and R. Poli, Eds.

[104] Guarino, N. Understanding, Building, And Using Ontologies. *International Journal of Human-Computer Studies 46*, 2-3 (February/March 1997), 293 – 310.

[105] Guarino, N. Formal ontology and information systems. In *Formal Ontology in Information Systems. Proceedings of FOIS98, Trento, Italy, 6-8 June 1998.* (1998), N. Guarino, Ed., pp. 3–15.

[106] Haarslev, V., Lutz, C., and Möller, R. Foundations of spatioterminological reasoning with description logics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)* (June 1998), A. G. Cohn, L. Schubert, and S. Shapiro, Eds., Morgan-Kaufmann Publishers, pp. 112–124.

[107] Haarslev, V., Moeller, R., and Wessel, M. *RACER Users Guide and Reference Manual Version 1.7.19*, 2004.

[108] Haarslev, V., and Moller, R. Racer: An OWL Reasoning Agent for the SemanticWeb. In *Proceedings of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with the 2003 IEEE/WIC International Conference on Web Intelligence, Halifax, Canada.* (October 2003), p. 9195.

[109] Haas, H., and Brown, A. Web Services Glossary. Working group note, W3C, February 2004.

[110] Handschuh, S., Staab, S., and Volz, R. On deep annotation. In *Proceedings of The Twelfth International World Wide Web Conference, Budapest, Hungary* (May 2003).

[111] Harlow, M., Prince, G., Jones, C., Tucker, C., and Reinhart, J. *Geoprocessing Commands Quick Reference Guide.* ESRI, 2004.

[112] Hartmann, J., Sure, Y., Haase, P., del Carmen Suárez-Figueroa, M., Studer, R., Gómez-Pérez, A., and Palma, R. Ontology Metadata Vocabulary and Applications. In *Proceedings International Conference on Ontologies, Databases and Applications of Semantics, Workshop on Web Semantics (SWWS)* (October 2005), R. Meersman, Ed.

[113] Hess, C., and de Vries, M. From Models to Data: A Prototype Query Translator for the Cadastral Domain. In *Workshop on Standardization in the Cadastral Domain, held by COST Action G9 and FIG Commission 7, Bamberg, Germany* (December 2004).

[114] Hess, C., and Schlieder, C. Ontology-based Verification of Core Model Conformity in Conceptual Modeling. In *Workshop on Standardization in the Cadastral Domain, held by COST Action G9 and FIG Commission 7, Bamberg, Germany* (December 2004).

[115] Hill, L. L. Core elements of digital gazetteers: placenames, categories, and footprints. In *Research and Advanced Technology for Digital Libraries : Proceedings of the 4th European Conference, ECDL 2000 Lisbon, Portugal, September 18-20, 2000* (2000), J. Borbinha and T. Baker, Eds., Springer, pp. 280–290.

[116] Hiramatsu, K., and Reitsma, F. GeoReferencing the Semantic Web: ontology based markup of geographically referenced information. In *Proceedings Joint EuroSDR/EuroGeographics workshop on Ontologies and Schema Translation Services, Paris, France* (April 2004). Available at: http://www.mindswap.org/2004/geo/geoStuff_files/Hiramatsu-Reitsma04GeoRef.PDF.

[117] Hollink, L., Schreiber, G., Wielemaker, J., and Wielinga, B. Semantic Annotation of Image Collections. In *Proceedings Knowledge Capture 2003, Knowledge Markup and Semantic Annotation Workshop* (October 2003), S. Handschuh, M. Koivunen, R. Dieng, and S. Staab, Eds.

[118] Horridge, M., Knublauch, H., Rector, A., Stevens, R., and Wroe, C. *A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools Edition 1.0*, 2004.

[119] Horrocks, I. DAML+OIL:a Reason-able Web Ontology Language. In *Proceedings of the International Conference on Extending Database Technology (EDBT)* (2002).

[120] Horrocks, I., McGuinness, D. L., and Welty, C. A. Digital Libraries and Web-Based Information Systems. In *The Description Logic Handbook: Theory, Implementation, and Applications*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., 427-449. Cambridge University Press, 2003.

[121] Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics 1*, 1 (2003).

[122] Horrocks, I., Sattler, U., and Tobies, S. Reasoning with Individuals for the Description Logic SHIQ . In *Proceedings of the 17th International Conference on Automated Deduction* (2000), vol. 1831 of *Lecture Notes In Computer Science*, Springer-Verlag London, UK, pp. 482 – 496.

[123] IBM Corporation and The Eclipse Foundation. Eclipse Platform Technical Overview. http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-overview-2005-12.pdf, December 2005. Accessed December 2005.

[124] Islam, A. S., Bermudez, L., Fellah, S., Beran, B., and Piasecki, M. Implementation of the Geographic Information - Metadata (ISO 19115:2003) Norm using the Web Ontology Language (OWL). http://www.pages.drexel.edu/∼leb27/research.html, 2004.

[125] ISO/IEC. International Standard Information technology, Open Distributed Processing, Reference model (RM-ODP): Foundations, First edition. Tech. Rep. 10746-2, International Organization for Standardizatiopn, International Electrotechnical Commission, September 1996.

[126] ISO/IEC. International Standard Information technology, Open Distributed Processing, Reference model (RM-ODP): Overview, First edition. Tech. Rep. 10746-1, International Organization for Standardizatiopn, International Electrotechnical Commission, 1998.

[127] ISO/TC211. Final Draft International Standard 19125-2: Geographic information - Simple feature access Part 2: SQL option. Tech. Rep. ISO/TC 211 N 970, International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, August 2000.

[128] ISO/TC211. Final Draft International Standard 19101 Geographic information - Reference model. Tech. Rep. N 1197, International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, October 2001.

[129] ISO/TC211. Final Draft International Standard 19107 Geographic information - Spatial schema. Text for registration as FDIS ISO/FDIS 19107:2002(E) N1324, International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, 2002.

[130] ISO/TC211. Final Draft International Standard 19111 Geographic information - Spatial referencing by coordinates, as sent to the ISO Central Secretariat for registration as FDIS. Tech. Rep. N 1292, International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, June 2002.

[131] ISO/TC211. Final Draft International Standard 19109 Geographic information - Rules for application schema. Tech. Rep. ISO/FDIS 19109:2003(E), International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, 2003.

[132] ISO/TC211. Final Draft International Standard 19119 Geographic information - Services. Tech. Rep. ISO/TC 211 N 1540, International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, November 2003.

[133] ISO/TC211. Final Draft International Standard 19110, Geographic information Methodology for feature cataloguing. Tech. Rep. N 1567, International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, January 2004.

[134] ISO/TC211. Final Draft International Standard 19123 Geographic information - Schema for coverage geometry and functions. Tech. Rep. ISO/FDIS 19123:2004(E) N 1740, International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, 2004.

[135] ISO/TC211. ISO/TC 211 N 1563 Text for IS 19125-1 Geographic information Simple feature access Part 1: Common architecture, as sent to the ISO Central Secretariat for issuing as International Standard. Tech. rep., ISO/TC 211 Secretariat, 2004.

[136] ISO/TC211. New work item proposal and DTS 19103, Geographic information Conceptual schema language.

[137] ISO/TC211. Draft International Standard 19131 Geographic information - Data product specifications. Draft International Standard ISO/TC 211 N 1786, International Organization for Standardization, Technical Committee ISO/TC 211, Geographic information/Geomatics, April 2005.

[138] ISO/TC211. ISO/TC 211 N 1762 Text for ISO 19133 Geographic information - Location based services - Tracking and navigation as sent to ISO for publication. Tech. rep., ISO/TC 211 Secretariat, 2005.

[139] KAGIS. ISA-Map, INTERREG III B CADSES Project. Project web site: http://www.isamap.info/index.html, 2006. Accessed January 2006.

[140] Kalfoglou, Y., and Schorlemmer, M. Ontology mapping:the state of the art. *The Knowledge Engineering Review 18* (2003), 1–13.

[141] Kalyanpur, A., Hendler, J., Parsia, B., and Golbeck, J. SMORE - Semantic Markup, Ontology, and RDF Editor. http://www.mindswap.org/papers/SMORE.pdf.

[142] Kavouras, M. A Unified Ontological Framework For Semantic Integration. In *Proceedings Workshop on Next Generation Geospatial Information, Cambridge, Massachusetts, USA* (October 2003).

[143] Kim, S., and Rosu, M. A Survey of Public Web Services. In *Proceedings EC-Web 2004* (August 2004), vol. LNCS 3182.

[144] Kirwan, B., and Ainsworth, L. K. *A guide to task analysis.* Taylor and Francis, 1992.

[145] Klien, E., and Lutz, M. The Role of Spatial Relations for Automating the Semantic Annotation of Geodata. In *Proceedings Conference on Spatial Information Theory (COSIT 2005)* (2005).

[146] Klien, E., Lutz, M., and Kuhn, W. Ontology-Based Discovery of Geographic Information Services - An Application in Disaster Management. *Computers, Environment and Urban Systems (CEUS) In Press, Available online* (July 2005).

[147] Klien, E., and Probst, F. Requirements for Geospatial Ontology Engineering. In *Proceedings of the 8th Conference on Geographic Information Science (AGILE 2005), Estoril, Portugal* (2005), F. Toppen and M. Painho, Eds., pp. 251–260.

[148] Knippers, R., and Kraak, M.-J. TOP10 vector Object-oriented. Tech. rep., ITC, 2002.

[149] Knublauch, H., Fergerson, R. W., Noy, N. F., and Musen, M. A. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In *Proceedings Third International Semantic Web Conference, Hiroshima, Japan* (2004).

[150] Koutsomitropoulos, D. A., Fragakis, M. F., and S.Papatheodorou, T. A Methodology for Conducting Knowledge Discovery on the Semantic Web. In *Proceedings Sixteenth ACM Conference on Hypertext and Hypermedia (Hypertext 2005), Salzburg, Austria* (September 2005).

[151] Kresse, W., and Fadaie, K. *ISO Standards for Geographic Information.* Springer, 2004.

[152] Kuhn, W. Ontologies in Support of Activities in Geographic Space. *International Journal of Geographical Information Science 15(7)* (2001), 613–631.

[153] Kuhn, W. Semantic reference systems. *International Journal of Geographical Information Science 17*, 5 (July-August 2003), 405–409.

[154] Kuhn, W., and Raubal, M. Implementing Semantic Reference Systems. In *Proceedings 6th AGILE Conference on Geographic Information Science, Lyon, France* (2003), M. Gould, R. Laurini, and S. Coulondre, Eds., Presses Polytechniques et Universitaires Romandes, pp. 63–72.

[155] Lara, R., Roman, D., Polleres, A., and Fensel, D. A Conceptual Comparison of WSMO and OWL-S. In *Proceedings European Conference on Web Services (ECOWS 2004), Erfurt, Germany* (September 2004), pp. 254–269.

[156] Larman, C. *Applying UML and patterns*, second ed. Prentice Hall, 2002.

[157] Lemmens, R. Ontology based chaining of distributed geographic information systems. In *Poster proceedings of the 2nd international semantic web conference ISWC, Sanibel, Florida, USA* (October 2003).

[158] Lemmens, R. Modelling Semantic Web Services with OWL-S and Geo Domain Ontologies. Poster, PhD day ITC, October 2004.

[159] Lemmens, R. Exploitation of ontology mappings for the discovery of geo-web services. In *Proceedings GIScience 2006* (September 2006), IfGI Prints. Forthcoming.

[160] Lemmens, R., and Arenas, H. Semantic Matchmaking in Geo Service Chains: Reasoning with a Location Ontology. In *Proceedings 15th International Workshop on Database and Expert Systems Applications (DEXA 2004), Zaragoza, Spain* (2004), IEEE Computer Society, pp. 797–802.

[161] Lemmens, R., and de By, R. Distributed GIS and metadata : methods for the description of interoperable GIS components. In *Proceedings ISPRS Commission II Symposium, Working group 5, International workshop on mobile and internet GIS, Wuhan, China* (August 2002). Keynote.

[162] Lemmens, R., and de Vries, M. Semantic Description of Location Based Web Services using an Extensible Location Ontology. In *Proceedings of Münster GI-days 2004 : Geoinformation and mobility* (July 2004), IfGI prints 22, pp. 261–276.

[163] Lemmens, R., de Vries, M., and Aditya, T. Semantic extension of geo web service descriptions with ontology languages. In *Proceedings of the AGILE 2003 conference, The science behind infrastructure, Lyon, France* (April 2003), pp. 595–600.

[164] Lemmens, R., Granell, C., Wytzisk, A., de By, R., Gould, M., and van Oosterom, P. Integrating Semantic and Syntactic Descriptions for Chaining Geographic Services. *IEEE Internet Computing* (September/October 2006), 18–28. Forthcoming.

[165] Lemmens, R., Granell, C., Wytzisk, A., de By, R., Gould, M., and van Oosterom, P. Semantic and syntactic service descriptions at work in geo-service chaining. In *Proceedings 9th AGILE Conference on Geographic Information Science, Visegrád, Hungary* (2006), J. Suarez and B. Markus, Eds., College of Geoinformatics, University of West Hungary, pp. 51–61.

[166] Lenat, D. B. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM 38*, 11 (1995), 33–38.

[167] Lesperance, Y., Kelley, T. G., Mylopoulos, J., and Yu, E. S. Modeling Dynamic Domains with ConGolog. In *CAiSE'99* (1999), M. Jarke and A. Oberweis, Eds., Springer-Verlag Berlin Heidelberg.

[168] Li, L., and Horrocks, I. A Software Framework For Matchmaking Based on Semantic Web Technology. In *WWW2003, Budapest Hungary* (2003).

[169] Lieberman, J., Pehle, T., and Dean, M. Semantic Evolution of Geospatial Web Services. In *Proceedings W3C Workshop on Frameworks for Semantics in Web Services* (2005). Available at http://www.w3.org/2005/04/FSWS/accepted-papers.html.

[170] Limbourg, Q., Pribeanu, C., and Vanderdonckt, J. Towards uniformed of task models in a model-based approach. In *Proceedings of the 8th International Workshop on Design, Specification and Verification of Interactive Systems, Glasgow* (2001), C. Johnson, Ed., vol. 2220 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 164–182.

[171] Liskov, B., and Guttag, J. *Abstraction and Specification in Program Development.* MIT Press/McGraw-Hill, 1986.

[172] Lutz, M. Ontology-based service discovery in spatial data infrastructures. In *Proceedings Workshop on Geographic Information Retrieval (GIR 2005), Bremen, Germany* (2005).

[173] Lutz, M., and Klien, E. Ontology-Based Retrieval of Geographic Information. *International Journal of Geographical Information Science 20*, 3 (2006), 233–260.

[174] Lutz, M., Riedemann, C., and Probst, F. A classification framework for approaches to achieving semantic interoperability between gi web services. In *Proceedings Conference on Spatial Information Theory COSIT 2003, Ittingen, Switzerland* (2003), W. Kuhn, M. Worboys, and S. Timpf, Eds., pp. 200–217.

[175] Maathuis, B., and van Westen, C. Flood hazard analysis using multitemporal SPOT-XS imagery. In *ILWIS application guide.* International Institute for Aerospace Survey and Earth Sciences (ITC), 2005, ch. 2, pp. 19–28.

[176] Maedche, A., Motik, B.and Silva, N., and Volz, R. Mafra - a mapping framework for distributed ontologies. In *Proceedings of the EKAW (Knowledge Engineering and Knowledge Management)* (2002), vol. 2473 of *Lecture Notes in Computer Science*, Springer.

[177] Maedche, A., and Staab, S. Ontology Learning. In *Handbook on Ontologies* (2004), S. Staab and R. Studer, Eds., Springer, pp. 173–190.

[178] Maedche, A., and Zacharias, V. Clustering Ontology-based Metadata in the Semantic Web. In *Proceedings Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002* (2002), T. Elomaa, H. Mannila, and H. Toivonen, Eds.

[179] Manola, F., and Miller, E. RDF Primer, W3C Recommendation. http://www.w3.org/TR/rdf-primer/, February 2004. Accessed January 2006.

[180] Mark, D. M., Smith, B., and Tversky, B. Ontology and Geographic Objects: An Empirical Study of Cognitive Categorization. In *Proceedings COSIT, Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science* (1999), C. Freksa and D. M. Mark, Eds., no. 1661 in Lecture Notes in Computer Science, Springer, p. 283298.

[181] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. OWL-S: Semantic Markup for Web Services. W3C Member Submission, expository document, World Wide Web Consortium, November 2004. Available at http://www.w3.org/Submission/2004/07/.

[182] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., and Schneider, L. The WonderWeb Library of Foundational Ontologies, Preliminary Report. Wonderweb deliverable d17, National Research Council, Institute of Cognitive Sciences and Technology, Italy, May 2003.

[183] McGuinness, D. L. Ontologies come of age. In *The Semantic Web: Why, What and How*, D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, Eds. MIT Press, 2001.

[184] McGuinness, D. L., and van Harmelen, F. OWL Web Ontology Language Overview. W3C Recommendation, World Wide Web Consortium, February 2004. Available at http://www.w3.org/TR/owl-features/. Accessed December 2005.

[185] Melnik, S., and Decker, S. A Layered Approach to Information Modeling and Interoperability on the Web. In *Proceedings ECDL'00 Workshop on the Semantic Web* (Lisbon, Portugal, 2000).

[186] Microsoft Corporation. DCOM Technical Overview. http://msdn.microsoft.com/library/, November 1996. Accessed September 2005.

[187] Miller, G. A. Nouns in Wordnet. In *WordNet: An Electronic Lexical Database* (1998), C. Fellbaum, Ed., MIT Press, pp. 23–46.

[188] Molenaar, M. *An introduction to the theory of spatial object modelling.* Taylor and Francis, 1998.

[189] Moormann Zaremski, A., and Wing, J. M. Specification matching of software components. *ACM Transactions on Software Engineering and Methodology 6*, 4 (1997), 333–369.

[190] Morales, J. *Model-driven design of geo-information services.* PhD Thesis, ITC dissertation number 110, International Institute for Geo-information Science and Earth Observation, Enschede, the Netherlands, 2004.

[191] Motta, E., Domingue, J., Cabral, L., and Gaspari, M. IRS-II: A Framework and Infrastructure for Semantic Web Services. In *Proceedings The SemanticWeb - ISWC 2003 Second International SemanticWeb Conference* (Sanibel Island, FL, USA, 2003), J. M. Dieter Fensel, Katia Sycara, Ed., vol. Lecture Notes in Computer Science Volume 2870 / 2003, Springer-Verlag Heidelberg.

[192] Narayanan, S., and McIlraith, S. Analysis and Simulation of Web Services. *Computer Networks 42* (2003), 675–693.

[193] Nardi, D., and Brachman, R. J. An Introduction to Description Logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge University Press, 2003, pp. 1–40.

[194] Niles, I., and Pease, A. Towards a standard upper ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001) Ogunquit, Maine, October 17-19* (2001), C. Welty and B. Smith, Eds.

[195] Normcommissie 351 240 Geo-informatie. NEN 3610:2005 Basismodel Geo-informatie. Normative document, Nederlands Normalisatie-instituut, September 2005. In Dutch.

[196] Noy, N., McGuinness, D., and Hayes, P. J. Semantic Integration & Interoperability Using RDF and OWL. W3c editor's draft (editors: Mike ushold and christopher menzel), World Wide Web Consortium, November 2005. Available at http://www.w3.org/2001/sw/BestPractices/OEP/SemInt/. Accessed December 2005.

[197] Noy, N. F. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record 33*, 4 (December 2004), 65–70.

[198] Noy, N. F., and McGuinness, D. L. Ontology development 101: A guide to creating your first ontology. Tech. Rep. KSL-01-05, Knowledge Systems Laboratory, March 2001.

[199] Noy, N. F., and Musen, M. A. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *In Proceedings of AAAI-2000, Austin, Texas.* (2000), MIT Press/AAAI Press.

[200] OASIS. OASIS web portal for Universal Description, Discovery and Integration (UDDI). http://www.uddi.org/. Organization for the Advancement of Structured Information Standards. Accessed September 2005.

[201] OASIS. Web Services Business Process Execution Language (WSBPEL), Version 2.0. Committee Draft. http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsbpel, September 2005. Accessed December 2005.

[202] Object Management Group. Unified Modeling Language 2.0 Specification. Available at http://www.uml.org/#UML2.0, July 2004. Accessed January 2006.

[203] OGC. The OpenGIS Abstract Specification, Topic 5: Features, Version 4. OpenGIS Project Document 99-105r2, Open Geospatial Consortium, 1999. Editor: Cliff Kottman.

[204] OGC. OGC Web Services Initiative Thread Set 2 and Demonstration, Annex B, Appendix D: OGC Interoperability Program Service Model (IPSM). Request for quotation and call for participation, Open Geospatial Consortium, March 2002.

[205] OGC. The OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture Version 4.3. Tech. rep., Open Geospatial Consortium, 2002. Editor: Percivall, George.

[206] OGC. Web Map Service. Part 2: XML for Requests using HTTP POST. OpenGIS Implementation Specification OGC 02-017, OpenGIS Consortium, March 2002.

[207] OGC. OpenGIS Reference Model. Tech. rep., Open Geospatial Consortium, 2003.

[208] OGC. Web Coverage Service (WCS), Version 1.0.0. OpenGIS Implementation Specification OGC 03-065r6, Open GIS Consortium, August 2003. Editors: John D. Evans.

[209] OGC. OpenGIS Catalogue Services Specification Version 2.0. OpenGIS Implementation Specification OGC 04-021r2, Open GIS Consortium, May 2004. Editors: Douglas Nebert and Arliss Whiteside.

[210] OGC. OpenGIS Geography Markup Language (GML) Implementation Specification Version 3.1.0. OpenGIS Recommendation Paper OGC 03-105r1, Open GIS Consortium, February 2004. Editors: Simon Cox, Paul Daisey, Ron Lake, Clemens Portele and Arliss Whiteside.

[211] OGC. Web Map Service, Version: 1.3. OGC Implementation OGC 04-024, Open Geospatial Consortium, August 2004.

[212] OGC. Web Spatial Analysis Service (WSAS) Implementation Specification version 0.0.1. OpenGIS Discussion Paper OGC 05 048, Open Geospatial Consortium, August 2004. Editors: Florian Straub and Andreas Donaubauer.

[213] OGC. Activity Plan for a Proposed Interoperability Experiment Version 1.0. OGC Project Document 05-034, Open Geospatial Consortium, March 2005.

[214] OGC. Filter Encoding Implementation Specification Version 1.1.0. OpenGIS Implementation Specification OGC 04-095, Open Geospatial Consortium, May 2005. Editor: Panagiotis A. Vretanos.

[215] OGC. OGC Web Services Common Specification Version 1.0. OGC Implementation Specification OGC 05-008, Open Geospatial Consortium, May 2005. Editor: Arliss Whiteside.

[216] OGC. OpenGIS Location Services (OpenLS): Core Services. OpenGIS Implementation Specification OGC 05-016, Open Geospatial Consortium, May 2005. Editor: Marwa Mabrouk.

[217] OGC. Sensor Observation Service. OpenGIS Draft Implementation Specification OGC 05-088r1, Open GIS Consortium, October 2005. Editors: Arthur Na and Mark Priest.

[218] OGC. Web Feature Service Implementation Specification, Version: 1.1.0. OpenGIS Implementation Specification OGC 04-094, Open Geospatial Consortium, May 2005.

[219] OGC. Web Processing Service (WPS) Interoperability Experiment: Final report. OGC Interoperability Experiment Report OGC 05-051, Open Geospatial Consortium, May 2005. Editors: Peter Schut and Steven Keens.

[220] Ort, E. Ease of Development in Enterprise JavaBeans Technology. http://java.sun.com/developer/technicalArticles/ebeans/ejbease/index.html, October 2004. Accessed September 2004.

[221] Patel, C. *iAnnotate Documentation/User Manual.* Department of Biomedical Informatics, Columbia University, New York, October 2005. Avialable at http://www.dbmi.columbia.edu/ cop7001/iAnnotateTab/iannotate_doc.htm.

[222] Patel-Schneider, P. F., Hayes, P., and Horrocks, I. OWL Web Ontology Language Semantics and Abstract Syntax . W3c recommendation, World Wide Web Consortium, February 2004. Available at http://www.w3.org/TR/owl-semantics/. Accessed December 2005.

[223] Patil, A., Oundhakar, S., Sheth, A., and Verma, K. METEOR-S Web service Annotation Framework. *WWW 2004, May 1722, 2004, New York, New York, USA.* (2004).

[224] Pirotte, A., and Massart, D. Integrating Two Descriptions of Taxonomies with Materialization. *Journal of Object Technology 3*, 5 (May-June 2004), 143149.

[225] Pirotte, A., Zimányi, E., Massart, D., and Yakusheva, T. Materialization: a powerful and ubiquitous abstraction pattern. In *Proceedings of the 20th VLPB Conference Santiago, Chile* (1994).

[226] Piwowar, J., Ledrew, E., and Dudycha, D. Integration of Spatial Data in Vector and Raster Formats in a Geographic Information System Environment. *International Journal of Geographic Information Systems 4*, 4 (1990), 429–444.

[227] Probst, F. ACE-GIS project, download site for ontologies and prototypical tool for semantics-based web service discovery. http://musil.uni-muenster.de/onto/ACE/, 2004. Accessed December 2005.

[228] Probst, F., Gibotti, F. R., Morantes, A. M. P., Esbri, M. A., de Barros Filho, M. B. B., Gutierrez, M., and Kuhn, W. Connecting iso and ogc standards to the semantic web. In *Third International Conference on Geographic Information Science, Adelphi, MD, USA.* (2004).

[229] Probst, F., Maué, P., Soon, K. H., Schade, S., Sen, S., and Kuhn, W. ACE-GIS, Adaptable and Composable E-commerce and Geographic Information Services, D6.3.0 - Report and Refereed Conference and Journal Publications on Algebraic Model and Pilot Prototype. Ist-2001-37724 project report, September 2004.

[230] Quartel, D., Pires, L. F., and van Sinderen, M. On architectural support for behavour refinement in distributed systems design. *Transactions of the SDPS, Society for Design and Process Science 6*, 1 (2002), 1–30.

[231] Quek, A., and Shah, H. A Comparative Survey of Activity-based Methods for Information Systems Development. In *ICEIS - 6th International Conference on Enterprise Information Systems, Porto, Portugal* (2004).

[232] Racer. *RacerPro Reference Manual Version 1.8.* Racer Systems GmbH & Co. KG, April 2005.

[233] Racer. *RacerPro User's Guide Version 1.8.* Racer Systems GmbH & Co. KG, April 2005.

[234] Rao, J. *Semantic Web Service Composition via Logic-based Program Synthesis.* PhD thesis, Norwegian University of Science and Technology, December 2004.

[235] Rao, J., and Su, X. A Survey of Automated Web Service Composition Methods. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004, San Diego, California, USA* (July 2004).

[236] Rector, A. Foundations of the Semantic Web:Ontology Engineering Building Ontologies 1b, Classes and Instances, Concepts and Individuals. http://www.cs.man.ac.uk/ rector/Modules/CS646-2004/Ontology-building-2004-lect4b-individuals-and-classes.pdf, 2004. Lecture slides.

[237] Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., and Wroe, C. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In *Proceedings 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW), Whittlebury Hall, Northamptonshire, UK* (2004).

[238] Reitsma, F., and Bittner, T. Scale in object and process ontologies. In *COSIT 2003* (2003), W. Kuhn and M. W. ans S. Timpf, Eds., pp. 13–27.

[239] Richards, D. Reverse Engineering Ontologies from Performance Systems. In *Proceedings 22nd Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGES), (ES2002), Cambridge, UK* (December 2002).

[240] Risicokaart. Risk map glossary. http://www.risicokaart.nl/. Accessed September 2005.

[241] Rodríguez, A., and Egenhofer, M. Determining Semantic Similarity Among Entity Classes from Different Ontologies. *IEEE Transactions on Knowledge and Data Engineering 15*, 2 (2003), 442–456.

[242] Rodríguez, M. A., Cruz, I. F., Levashkin, S., and Egenhofer, M. J., Eds. *GeoSpatial Semantics: First International Conference, GeoS 2005, Mexico City, Mexico* (November 2005), LNCS 3799.

[243] Sabou, M., Wroe, C., Goble, C., and Mishne, G. Learning Domain Ontologies for Web Service Descriptions: an Experiment in Bioinformatics. In *Proceeedings of the 14th International World Wide Web Conference (WWW2005), Chiba, Japan* (May 2005).

[244] Sattler, U. Description logic reasoners. http://www.cs.man.ac.uk/-~sattler/reasoners.html. Accessed January 2006.

[245] Sattler, U. Description Logics for Ontologies. http://www.cs.man.ac.uk/-~sattler/publications/hschrift.ps, April 2003. Habilitationsschrift, TU Dresden.

[246] SAVE. Leidraad Risico-inventarisatie Model-risicokaart - RRGS. Tech. Rep. Version 1.1 PRK-87, Ingenieurs/adviesbureau SAVE, November 2003.

[247] Schade, S. Sensors on the Way to Semantic Interoperability. In *Proceedings GI-Days 2005: Geo-Sensor Networks - From Science to Practical Applications.. Münster.* (2005), ifgi-Prints No. 23.

[248] Schlieder, C., Vögele, T., and Visser, U. Qualitative Spatial Reasoning for Information Retrieval by Gazetteers. In *Proceedings of the Conference on Spatial Information Theory (COSIT'01)* (September 2001), D. Montello, Ed., Springer-Verlag, pp. 336–351.

[249] Schmelzer, R., Vandersypen, T., Bloomberg, J., Siddalingaiah, M., Hunting, S., Qualls, M., Darby, C., Houlding, D., and Kennedy, D. *XML and Web Services Unleashed*, 1st edition ed. Sams, 2002.

[250] Schmidt-Belz, B., Nick, A., Robertshaw, S., and Zipf, A. Smart tourism taxonomy. Crumpet project report ist-1999-20147/gmd/wp1/d1.5, CRUMPET, Creation of USer Friendly Mobile Services Personalised for Tourism, 2001.

[251] Schwering, A., and Raubal, M. Spatial Relations for Semantic Similarity Measurement. In *Proceedings 2nd International Workshop on Conceptual Modeling for Geographic Information Systems, CoMoGIS2005, 24th International Conference on Conceptual Modeling, ER 2005, Klagenfurt, Austria* (October 2005), Lecture Notes in Computer Science, Springer.

[252] Sintek, M. Ontoviz Tab documentation. http://protege.stanford.edu/-plugins/ontoviz/readme.txt, May 2004. Accessed September 2005.

[253] Skylab. Skylab Mobile Systems OGC WMS Server List. http://www.skylab-mobilesystems.com/en/wms_serverlist.html. Accessed September 2005.

[254] Stevens, S. On the theory of scales of measurement. *Science 103* (1946), 677–680.

[255] Störrle, H., and Hausmann, J. H. Towards a Formal Semantics of UML 2.0 Activities. *Software Engineering* (2005), 117–128.

[256] Stoter, J., Lemmens, R., Köbben, B., and Bakker, N. J. Semantic Data Integration of a Topographic Database. In *Proceedings Workshop on Multiple Representation and Interoperability of Spatial Data, Hannover, Germany* (February 2006).

[257] Stuckenschmidt, H., and Harmelen, F. v. *Information sharing on the Semantic Web*. Springer, 2005.

[258] Stumme, G., and Maedche, A. Ontology merging for federated ontologies on the semantic web. In *Proceedings of the International Workshop for Foundations of Models for Information Integration (FMII-2001), Viterbo, Italy* (September 2001).

[259] SWSI. Semantic Web Services Ontology (SWSO) Version 1.0. Part of the initial technical report of the Semantic Web Services Language (SWSL) Committee of the Semantic Web Services Initiative (SWSI). Tech. rep., SWSI, May 2005.

[260] Szyperski, C., Gruntz, D., and Murer, S. *Component Software: Beyond Object Oriented Programming*, second ed. Addison-Wesley Professional, 2002.

[261] Tanenbaum, A. *Computer networks*, third edition ed. Prentice-Hall, 1996.

[262] Tang, S. Matching of web service specifications using daml-s descriptions. Master's thesis, Technische Universitat Berlin, 2004.

[263] Timpf, S. Geographic Activity Models. In *Foundations of Geographic Information Science*, M. Goodchild, M. Duckham, and M. F. Worboys, Eds. Taylor& Francis., 2003, pp. 241–254.

[264] Trastour, D., Bartolini, C., and Preist, C. Semantic web supporting for the business-to-business e-commerce lyfecycle. In *WWW2002, Honolulu, HAwaii, USA* (2002).

[265] Tu, S. W., Eriksson, H., Gennari, J., Shahar, Y., and Musen, M. A. Ontology-Based Configuration of Problem-Solving Methods and Generation of Knowledge-Acquisition Tools: Application of PROTÉGÉ-II to Protocol-Based Decision Support. *Artificial Intelligence in Medicine 7*, 3 (1995), 257–289.

[266] Uitermark, H. *Ontology-based geographic data set integration.* PhD thesis, Universiteit Twente, 2001.

[267] Uitermark, H. T., van Oosterom, P. J. M., Mars, N. J. I., and Molenaar, M. Ontology-Based Geographic Data Set Integration. In *Proceedings International Workshop on Spatio-Temporal Database Management STDBM'99, Edinburgh, Scotland, UK* (September 1999), M. H. Böhlen, C. S. Jensen, and M. O. Scholl, Eds., vol. 1678 of *Lecture Notes in ComputerScience*, Springer, Berlin, pp. 60–78.

[268] Uschold, M., and Gruninger, M. Ontologies: Principles, methods and applications.

[269] Vallecillo, A. RM-ODP: The ISO Reference Model for Open Distributed Processing. *DINTEL Edition on Software Engineering. No. 3. ISBN: 84-931933-2-1, pp.. March 2001*, 3 (March 2001), 69–99.

[270] van Heijs, G., Schreiber, A. T., and Wielinga, B. J. Using explicit ontologies for KBS development. *International Journal of Human-Computer Studies 42*, 2/3 (1997), 183–292.

[271] Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., and Ciravegna, F. Mnm: Ontology driven semi-automatic and automatic support for semantic markup. In *Proceedings of EKAW 2002* (2002), LNCS 2473, Springer, p. 379391.

[272] Vckovski, A. *Interoperable and Distributed Processing in GIS.* Research Monographs in Geographic Information Systems. 1998.

[273] Veltman, K. H. Towards a Semantic Web for Culture. *Journal of Digital Information 4*, 4 (2004).

[274] Verbraeck, A., Saanen, Y., Stojanovic, Z., Valentin, E., Meer, K. v. d., Meijer, A. B., and Shishkov, B. What are building blocks? In *Building blocks for Effective Telematics Application Development and Evaluation*, A. Verbraeck and A. Dahanayake, Eds. Delft University of Technology, Delft, 2002.

[275] Verbyla, D. L. *Practical GIS Analysis.* Taylor and Francis, 2001.

[276] Vinoski, S. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine 14*, 2 (February 1997).

[277] Visser, U., Vögele, T., and Schlieder, C. Spatio-Terminological Information Retrieval using the BUSTER System. In *Proceedings of the EnviroInfo, Wien (AT)* (2002), pp. 93–100.

[278] Vissers, C. A., Pires, L. F., Quartel, D. A., and van Sinderen, M. J. The architectural design of distributed systems; Reader for The Design of Telematic Systems. University of Twente, March 1998.

[279] Vivid Solutions. Jts topology suite technical specifications. http://www.vividsolutions.com/jts/jtshome.htm. Accessed December 2005.

[280] Vögele, T., Hübner, S., and Schuster, G. BUSTER - an information broker for the semantic web. *Künstliche Intelligenz 3* (July 2003), 31–34.

[281] Vries, M.E. de. *Distributed geo-information.* PhD thesis, Delft University of Technology, 2006 (Forthcoming).

[282] W3C. Extensible Markup Language (XML). Main page for World Wide Web Consortium (W3C) XML activity and information. World Wide Web Consortium. http://www.w3.org/XML/. Accessed September 2005.

[283] W3C. Semantic Web Services Interest Group web site. http://www.w3.org/2002/ws/swsig/. Accessed January 2006.

[284] W3C. Web Services Choreography Description Language (WS-CDL) Version 1.0. W3c working draft, World Wide Web Consortium, December 2004.

[285] W3C. Semantic Web Services Framework (SWSF) Overview. W3c member submission, World Wide Web Consortium, September 2005.

[286] Wagner, R. M., Panzer, N., and Menge, F. SDI Initiative North Rhine-Westphalia (GDI NRW) / Germany -SDI NRW Joint Project 2004: Identification of Enhanced SDI Elements. In *Proceedings GSDI-8 Cairo, Egypt, April 16-21, 2005* (2005).

[287] Wessel, M., and Möller, R. A High Performance Semantic Web Query Answering Engine. In *Proceedings International Workshop on Description Logics* (2005), I. Horrocks, U. Sattler, and F. Wolter, Eds.

[288] Winter, S., and Tomko, M. Translating the Web Semantics of Georeferences. In *Web Semantics and Ontology*, D. Taniar and W. Rahayu, Eds. Idea Publishing, Hershey, PA, 2006. In press.

[289] Wohed, P., van der Aalst, W. M., Dumas, M., and ter Hofstede, A. H. Pattern based analysis of bpel4ws. Tech. rep., Queensland University of Technology., 2002.

[290] Worboys, M., and Duckham, M. *GIS A Computing Perspective*, second ed. CRC Press, 2004.

[291] Yao, K.-T., Ko, I.-Y., Neches, R., and MacGregor, R. Semantic Interoperability Scripting and Measurements. In *Proceedings Working Conference on Complex and Dynamic Systems Architecture, Brisbane, Australia* (2001).

[292] Yeh, A. G.-O., and Qiao, J. J. Modelobjects: a model management compnent for the development of planning support systems. *Computers, Environment and Urban Systems 29* (2005), 133–157.

[293] Zhang, L., and Jeckle, M. Proceedings preface. In *International Conference on Web Services ICWS - Europe 2003, Erfurt, Germany* (September 2003), L.-J. Zhang and M. Jeckle, Eds., Lecture Notes in Computer Science, 2853.

[294] Zloof, M. M. Query-by-Example: A data base language. *IBM Systems Journal 16-4* (1977).

Curriculum Vitae

Rob Lemmens was born on May 2^{nd} 1965 in Den Haag, The Netherlands. He graduated in 1989 at the Delft University of Technology, Faculty of Geodesy, with a specialisation in Mathematical Geodesy and Point positioning. After research periods at the car navigation laboratory of Philips (Carin) and the National Aerospace Laboratory, he worked at the Faculty of Geodesy in a research project on the integrated use of GPS and echo soundings at sea.

From 1994 until 1998, he was seconded to the University of Zambia as a project consultant and lecturer in GIS and land surveying, where he was responsible for the development of the computer infrastructure of the Department of Surveying.

Since 1998 he works as assistant professor at the International Institute for Geo-information Science and Earth Observation (ITC), department of Geo-information Processing, where he develops and offers several courses in GIS, application development and internet GIS. He supervises graduate students on topics in Geo-information Infrastructures, geo-services and interoperability, and recently, geo-semantics. At ITC, he has been carrying out his PhD research on the topic of semantic interoperability of distributed geo-services.

Rob is actively participating in national and international projects in the field of Geo-information Infrastructures in Africa and Asia. He is ITC's representative for the Open Geospatial Consortium (OGC), where he participates in the working group on Information Communities and Semantics. He is also ITC's representative for the Association of Geographic Information Laboratories in Europe (AGILE), where he is active in the working group on Interoperability. In September 2004, he has initiated the Geosemantics Interest Group, which hosts a web portal with information about geo-semantics researchers.