
Declaration and Enforcement of Access Restrictions for Distributed Geospatial Information Objects

Andreas Matheus

Fakultät für Informatik der Technischen Universität München
Lehrstuhl für Angewandte Informatik

Declaration and Enforcement of Access Restrictions for Distributed Geospatial Information Objects

Andreas Matheus

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. P. P. Spies
Prüfer der Dissertation: 1. Univ.-Prof. Dr. J. Schlichter
2. Univ.-Prof. Dr. G. Teege,
Universität der Bundeswehr München

Die Dissertation wurde am 4.11.2004 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 1.02.2005 angenommen.

Zusammenfassung

Geoinformationen sind heutzutage in vielen Bereichen eine wesentliche Grundlage für die Bearbeitung komplexer Arbeitsabläufe. Anbieter von Geoinformationen sind verschiedene staatliche und amtliche Stellen sowie private Institutionen und Gesellschaften, die Geoinformationen in verschiedenen Formaten erheben und anschließend fortführen. Die sich daraus ergebende Verteiltheit von heterogenen Geoinformationen, kann über eine Infrastruktur von interoperablen Geo Web Services verfügbar gemacht werden. Ein Geo Web Service ist ein spezieller Web Service, der Funktionalitäten bereit stellt mit der eine Erzeugung, Verarbeitung und Fortführung von Geoinformationen möglich ist. Dies geschieht über standardisierte Schnittstellen, die unabhängig vom jeweiligen Format der angebotenen Geoinformationen ist. Eine kombinierte Nutzung, wie sie in komplexen Arbeitsabläufen erforderlich ist, basiert somit auf der Kombination von Abbildungen von Geoinformationen, die von den Geo Web Services auf Anfrage erstellt werden.

Weil die Nutzung mancher Geoinformationen lizenzrechtlichen Einschränkungen unterliegt, oder weil eine Bezahlung gefordert ist, muss der Anbieter von Geo Web Services entsprechende Zugriffsbeschränkungen deklarieren und durchsetzen. Dieser Schutz vor unerlaubtem Zugriff kann durch eine Zugriffskontrolle erfolgen. Geht man von einer objektartigen Struktur der Geoinformationen aus, so können Zugriffsbeschränkungen direkt für diese Informationsobjekte oder deren Raumbezug deklariert und durchgesetzt werden. Die wesentlichen Anforderungen sind:

- Eine Zugriffsbeschränkung gilt für bestimmte Typen von Informationsobjekten.
- Eine Zugriffsbeschränkung gilt nur für bestimmte Informationsobjekte.
- Eine Zugriffsbeschränkung gilt für ein geographisches Gebiet und somit für bestimmte Typen oder Instanzen von Informationsobjekten, die eine bestimmte raumbezogene Relation zu diesem geographischen Gebiet besitzen.

Das Ziel dieser Arbeit ist die Deklaration und Durchsetzung dieser Zugriffsbeschränkungen in einer Infrastruktur von verteilten raumbezogenen Informationsobjekten, die über interoperable Geo Web Services verfügbar sind. Ausgehend von einer validierbaren, XML kodierten Darstellung von Informationsobjekten, deren Raumbezug und Geometrie durch die Verwendung der Geography Markup Language (GML), eine XML Auszeichnungssprache des Open GIS Consortiums Inc. (OGC) beschrieben ist, soll eine Lösung aufgezeigt werden, die die obigen Anforderungen umsetzt. Bei der Deklaration der Zugriffsbeschränkungen kann es wegen der Feingranularität und des Raumbezugs zu verschiedenen Arten von Inkonsistenzen

der deklarierten Zugriffsbeschränkungen kommen. Hier werden Mechanismen vorgestellt, die eine Detektion und Klassifizierung bestehender Inkonsistenzen erlauben, um eine erforderliche Behebung vornehmen zu können.

Eine Durchsetzung der deklarierten Zugriffsbeschränkungen erfolgt unter Erweiterung des OASIS Standards eXtensible Access Control Markup Language (XACML). Diese Erweiterung (GeoXACML) beinhaltet zum einen die erforderlichen Sprachelemente, um die raumbezogenen Zugriffsbeschränkungen deklarieren zu können und zum anderen die entsprechenden Algorithmen, die eine entsprechende Durchsetzung erlauben. Die vorgestellte Lösung ist prototypisch als Web Service implementiert und stellt somit eine umfassende Lösung zur Deklaration und Durchsetzung von Zugriffsbeschränkungen bei der Nutzung von verteilten Geoinformationen mittels Geo Web Services dar.

Abstract

Today, geographic information is the key data for enabling decision making in many business sectors. Government agencies, private institutions and commercial companies provide geographic data of different kinds. Independent from each other, they collect and maintain geographic data in different formats. This results in an infrastructure of heterogenous and distributed geographic data. A service-oriented infrastructure can make the geographic data accessible in an interoperable way, independent from the original data format. A specific implementation of this infrastructure is based on Geo Web Services. A Geo Web Service is a specialization of a Web Service that allows the creation, continuation and geo-processing of geodata. The required functionality is available through standardized service interfaces. The combined use of different kinds of geographic data relies on the capabilities of a service, as it transforms the original data format to an interoperable format, based on the service invocation.

The access to geographic data can be restricted due to different issues such as licensing or commercial use. The data providers can achieve the desired protection by the declaration and enforcement of restrictions as they regulate the use of a service. The functional capabilities for the declaration and enforcement can be provided by an access control system. Assuming an object-oriented data model of the geographic data, access restrictions can be enforced for individual objects according to the following requirements:

- A restriction must be enforceable for all objects of a particular class.
- A restriction must be enforceable for individual objects.
- A restriction must be enforceable for individual objects based on their geometry if a particular topological relation between the object geometry and a given geometry is satisfied.

The aim of this work is the declaration and enforcement of these requirements for the infrastructure of heterogenous and distributed geospatial information objects, as they are accessible via the service-oriented infrastructure. Assuming a valid XML markup of the objects and their geometry using the Geographic Markup Language (GML), which is an international standard of the Open GIS Consortium, Inc. (OGC), a solution is introduced that allows the declaration and enforcement. Due to the introduced restrictions, the declaration of access restrictions can result in different kinds of inconsistencies. This work describes mechanism for the detection and classification of the different kinds of inconsistencies as they help to correct the problems.

The declaration and enforcement of the access restrictions is based on an extension of the eXtensible Access Control Markup Language (XACML), which is an international standard of the Organization for the Advancement of Structured Information Standards (OASIS). This extension (GeoXACML) contains the language constructs in order to declare the spatial restrictions and provides the algorithms to allow enforcement. The prototype implementation of the introduced solution for the service oriented infrastructure provides a comprehensive contribution for the use of protected heterogenous and distributed geographic information.

Acknowledgement

This work is the result of research done during my affiliation at the Technische Universität München. The topic of this work was embossed by the research project GeoPortal, sponsored by the Bavarian Government.

First of all, I would like to say thank you to Univ.-Prof. Dr. Johann Schlichter for supervising this work and give the freedom to work on it. Next, I like to say thank you to Univ.-Prof. Dr. Gunnar Teege for been a mentor during research and the preparation of this work. Then, I like to say thank you to my colleagues for many fruitful discussions; in particular Dr. Michael Koch and Dr. Wolfgang Wörndl.

I would also like to say thank you to Mrs. Suzette LaGray and Mr. Andre Farci for their American English proofreading and Mr. Matthias Presch for his valuable feedback.

Finally, I like to say thank you to my wife Nicole for her support, motivating words and giving me the freedom during leisure time to continue working on this dissertation.

Contents

Zusammenfassung	v
Abstract	vii
Acknowledgement	ix
1 Introduction and Motivation	1
1.1 Interoperable Use of Distributed and Heterogenous Geodata	2
1.2 Geodata and Geospatial Information Object	4
1.3 Protected and Distributed Geospatial Information Objects	5
1.4 A Motivating Example	7
1.5 The Scientific Method of this Dissertation	10
2 Basic Concepts	13
2.1 XML, XML Schema and Xpath	13
2.1.1 Extensible Markup Language (XML)	13
2.1.2 XML Schema	14
2.1.3 Xpath	15
2.2 Geodata	16
2.2.1 Geospatial Information Object	16
2.2.2 Geometry	19
2.2.3 Testing Topological Relations between Geometries	21
2.3 The Geography Markup Language (GML)	23
2.3.1 Encoding of Geospatial Information Objects and Geometry	23
2.3.2 Encoding of an Application Specific Data Model	27
2.3.3 GML and Interoperability	30
2.4 Interoperable Use of Distributed and Heterogenous Geodata	31

2.4.1	The Open GIS Consortium and their Interoperability Specifications	32
2.4.2	Definition of Interoperability	33
2.4.3	The Web Map Service (WMS) Implementation Specification	35
2.4.4	The Web Feature Service (WFS) Implementation Specification	37
2.4.5	Conclusion on Interoperability and Implications to Access Control	39
2.5	Introduction to Access Control	39
2.5.1	Used Terminology	39
2.5.2	Different Strategies for Managing Access Rights	40
2.5.3	The Basic Access Control System	41
2.5.4	Introduction to Role Based Access Control	42
2.5.5	Introduction to Rule Based Access Control	43
2.5.6	Enforcement of Declared Permissions	44
2.5.7	Finding Applicable Policies and Rules	45
2.5.8	Deriving an Authorization Decision	47
2.5.9	Illustrating the Decision Process	47
2.5.10	Different Strategies for the Declaration of Access Restrictions	50
2.6	Introduction to Distributed Access Control	52
2.6.1	Standardized Language for Assertions	53
2.6.2	Standardized Permission Language	54
2.6.3	Standardized Communication Between Components	54
3	Access Control Requirements and Related Standards and Systems	55
3.1	Access Control Requirements	55
3.1.1	Enforce Restrictions on the Resources, Independent from the Service	55
3.1.2	Class- and Object-Based Requirements	55
3.1.3	Spatial Requirements	56
3.1.4	Representational Requirements	57
3.1.5	Temporal Requirements	57
3.1.6	Communication Security Requirements	57
3.2	Framing the Problem Space and Identifying Infrastructure Constraints	57
3.2.1	Constraints from the Distributed Aspect	59
3.3	Standards for Distributed Access Control	60
3.3.1	Security Access Control Markup Language (SAML)	60
3.3.2	XML Access Control Language (XACL)	61
3.3.3	eXtensible Access Control Markup Language (XACML)	62

3.3.4	Digital Rights Management (DRM)	68
3.3.5	EXtensible Rights Markup Language (XrML)	70
3.4	Systems, Implementing Distributed Access Control	70
3.4.1	Shibboleth	71
3.4.2	Cardea	71
3.5	Conclusion of Useability and Implications for this Work	71
4	Declaration and Enforcement of Access Restrictions	73
4.1	Declaration of Restrictions	73
4.1.1	The XACML Request Tuple	74
4.1.2	Class-Based Restrictions	74
4.1.3	Object-Based Restrictions	76
4.1.4	Spatial Restrictions	77
4.1.5	Complex Spatial Restrictions	78
4.1.6	Declaration of General/Exceptional Restrictions	80
4.2	Enforcement of Restrictions	80
4.2.1	Enforcement of Class-Based Restrictions	81
4.2.2	Enforcement of Object-Based Restrictions	83
4.2.3	Enforcement of Spatial Restrictions	83
4.3	Visualization of Access Restrictions	94
4.3.1	Visualization of the Permission Hierarchy	94
4.3.2	Visualization of Spatial Restrictions	96
4.3.3	Visualization of Combined Spatial Restrictions	98
4.3.4	Remarks to the Visualization	100
4.4	Approximate Detection of Inconsistent Permissions	101
4.4.1	The Essential Test Conditions	103
4.4.2	Approximate Detection of Unreachable Class-Based Permissions	105
4.4.3	An Illustrating Example of Unreachable Permissions	108
4.4.4	Approximate Detection of Complete Class-Based Permissions	110
4.4.5	Approximate Detection of Contrary Class-Based Permissions	111
4.4.6	Approximate Detection of Unreachable Object-Based Permissions	112
4.4.7	Approximate Detection of Complete Object-Based Permissions	112
4.4.8	Approximate Detection of Contrary Object-Based Permissions	113
4.4.9	Approximate Detection of Unreachable Spatial Permissions	113
4.4.10	Approximate Detection of Incomplete Spatial Permissions	114

4.4.11	Approximate Detection of Contrary Spatial Permissions	114
4.5	Exact Detection of Inconsistent Permissions	130
4.5.1	Exact Detection of Unreachable Permissions	132
4.5.2	Exact Detection of Contrary Permissions	132
4.5.3	Exact Detection for Complete Permissions	134
4.6	Recommending a Structured Declaration of Permissions	142
4.6.1	Implications, using One Authorization Service	143
4.6.2	The Coordinate Reference System	144
4.6.3	Recommended Matching of PolicySet, Policy and Rule	145
4.6.4	An Illustrating Example, obeying the Recommend Structure	147
5	Evaluation and System Design	151
5.1	Architecture	151
5.2	GeoXACML, the Geospatial Extension to XACML	153
5.2.1	Extending the XACML Data Types	153
5.2.2	Extending the XACML Functions	154
5.3	Declaration of Permissions, using GeoXACML Encoding	156
5.3.1	The Target Element	156
5.3.2	The Declaration of Class-Based Restrictions	157
5.3.3	The Declaration of Object-Based Restrictions	158
5.3.4	The Declaration of Spatial Restrictions	159
5.4	Enforcement of Declared Permissions	159
5.4.1	The Authorization Decision Request	160
5.4.2	An Enforcement Service for the Web Feature Service	162
5.4.3	An Enforcement Service for the Web Map Service	165
5.5	Implementation and Evaluation of a Geospatial Authorization Service	171
5.5.1	Implementation of the SpatialPDP Main Class	171
5.5.2	Implementation of the GeoXACML Attribute Values	172
5.5.3	Implementation of the SpatialSelectorModule	174
5.5.4	Implementation of the GeoXACML Rule Combining Algorithms	175
5.5.5	Implementation of the GeoXACML Spatial Functions	176
5.6	Evaluation of the Implemented System	177
5.6.1	Evaluation of Class-Based Restrictions	177
5.6.2	Evaluation of Object-Based Restrictions	178
5.6.3	Evaluation of Spatial Restrictions	180

5.6.4	Evaluation of the Spatial Method Within	180
5.6.5	Evaluation of the Spatial Method Touches	180
5.6.6	Evaluation of the Complex Spatial Restriction	182
6	Conclusion and Outlook	187
6.1	Conclusion	187
6.2	Recommendation to the World of (Geospatial) Science	188
6.3	Outlook	189
6.3.1	The ‘Not Authorized’ Response and the Adversary Issue	189
6.3.2	The Handling of Requests with Insufficient Permissions	191
6.3.3	Permission Management and the Development of Service Orchestrations	193
6.3.4	Context-based Permissions	193
6.3.5	Dynamic Negotiation of Authentication Information under Consideration of Privacy	196
A	Notation	197
A.1	Nomenclature	197
A.2	Font Types	202
A.3	XML Spy Diagram Notation	203
B	The City Model	205
B.1	Map of the City Model	205
B.2	City Model Application Schema	206
B.3	City Model GML Document	207
	Bibliography	209

List of Figures

1.1	Combined use of different kinds of geospatial data	2
1.2	Consolidated data integration	3
1.3	Federated data integration	3
1.4	Initial situation before land parcel splitting	7
1.5	The resulting situation after land parcel splitting	7
1.6	Workflow for splitting a land parcel and updating the ownership information	8
1.7	Alternative example process	10
1.8	The structure of this dissertation	12
2.1	Object-oriented data model example	14
2.2	The WGS84 coordinate grid for Germany	20
2.3	The Gauss-Krüger coordinate grid for Germany	20
2.4	Simple geometry examples	21
2.5	Simplified geometry model for GML 2.1.2	24
2.6	GML 2.1.2 feature schema	25
2.7	The City Model example data model	29
2.8	GML application schema for the city model example	30
2.9	The basic functionality of an access control system	41
2.10	Representation of an example policy structure	49
2.11	Infrastructure of distributed access control	52
3.1	The service infrastructure for online access to protected resources	58
3.2	XML Schema definition of the XACML authorization decision request	63
3.3	XML Schema definition of the XACML PolicySet element	64
3.4	XML Schema definition of the Subject matching	65
3.5	The different Digital Rights Management processes	69
3.6	How to play an encrypted Windows Media with Microsoft DRM	70

4.1	Activity diagram of the specific enforcement process	81
4.2	G_R is outside of G_P	86
4.3	G_R is inside the hole of G_P	86
4.4	G_R is on the outer boundary of G_P	86
4.5	G_R is on the inner boundary of G_P	86
4.6	G_R is inside the ring of G_P	87
4.7	G_R is outside of G_P	87
4.8	G_R is inside the hole of G_P	87
4.9	G_R is outside but touching G_P	88
4.10	G_R is inside the hole but touching G_P	88
4.11	G_R is on the outer boundary of G_P	88
4.12	G_R is on the inner boundary of G_P	88
4.13	G_R is inside the ring but touching G_P on the outer boundary	89
4.14	G_R is inside the ring but touching G_P on the outer boundary	89
4.15	G_R is inside the ring of G_P	89
4.16	G_R crosses G_P 's outer boundary	89
4.17	G_R crosses G_P 's inner boundary	89
4.18	G_R crosses G_P 's inner and outer boundary	90
4.19	G_R crosses G_P 's inner and outer boundary	90
4.20	G_R is outside of G_P	91
4.21	G_R is inside the hole of G_P	91
4.22	G_R is outside but touching G_P	91
4.23	G_R is inside the hole but touching G_P	91
4.24	G_R is outside but touching G_P	91
4.25	G_R is inside the hole but touching G_P	91
4.26	G_R is inside the ring but touching G_P on the outer boundary	92
4.27	G_R is inside the ring but touching G_P on the inner boundary	92
4.28	G_R is inside the ring but touching G_P on the outer boundary	92
4.29	G_R is inside the ring but touching G_P on the inner boundary	92
4.30	G_R is inside the ring of G_P	93
4.31	G_R is overlapping G_P on the outer boundary	93
4.32	G_R is overlapping G_P on the inner boundary	93
4.33	G_R is overlapping G_P	94
4.34	G_R is overlapping G_P	94

4.35	Visualization of an example permission tree	95
4.36	Line styles for rendering spatial restricted areas	96
4.37	Line styles for rendering negative spatial restricted areas	97
4.38	Visual representation of the spatial example permission	97
4.39	Visual representation of the spatial example permission	97
4.40	Illustrating a permission-tree using an AND-graph	99
4.41	Visualization of the spatial conditions of an AND-graph	99
4.42	Illustrating a permission-tree using an OR-graph	99
4.43	Visualization of the spatial conditions of an OR-graph	99
4.44	Visualization of spatial permissions, as a multi-layered map	101
4.45	A permission tree with unreachable and potentially unreachable rules	108
4.46	An annotated permission tree with unreachable rules	110
4.47	A simple complete permission tree	112
4.48	Classification legend for illustration of satisfying and not satisfying geometries	117
4.49	Illustrating the test constraint $\text{Disjoint}(G_{P1}, G_{P2})$	118
4.50	Illustrating the test constraint $\text{Overlaps}(G_{P1}, G_{P2})$	119
4.51	Illustrating the test constraint $\text{Within}(G_{P2}, G_{P1})$	120
4.52	Illustrating the test constraint $\text{Equals}(G_{P1}, G_{P2})$	120
4.53	Illustrating the test constraint $\text{Disjoint}(G_{P1}, G_{P2})$	122
4.54	Illustrating the test constraint $\text{Overlaps}(G_{P1}, G_{P2})$	123
4.55	Illustrating the test constraint $\text{Equals}(G_{P1}, G_{P2})$	123
4.56	Illustrating the test constraint $\text{Within}(G_{P2}, G_{P1})$	124
4.57	Illustrating the test constraint $\text{Within}(G_{P2}, G_{P1})$	125
4.58	Illustrating the test constraint $\text{Within}(G_{P1}, G_{P2})$	125
4.59	Illustrating the test constraint $\text{Disjoint}(G_{P1}, G_{P2})$	127
4.60	Illustrating the test constraint $\text{Disjoint}(G_{P1}, G_{P2})$	127
4.61	Illustrating the test constraint $\text{Disjoint}(G_{P1}, G_{P2})$	128
4.62	Illustrating the test constraint $\text{Touches}(G_{P1}, G_{P2})$	129
4.63	Illustrating the test constraint $\text{Within}(G_{P1}, G_{P2})$	130
4.64	A permission tree with contrary rules	133
4.65	An annotated permission policy tree with contrary rules	134
4.66	Policy tree for the detection of incompleteness	136
4.67	Policy tree, annotated with incompleteness information	140
4.68	A permission tree with double matching	141

4.69	Recommended permission tree	150
5.1	4-Tier architecture for a service oriented architecture	151
5.2	Extended 4-Tier architecture, including access control	153
5.3	Calculation of the real-world location for the <code>GetFeatureInfo</code> request	170
5.4	Class diagram of the GeoXACML attributes	174
5.5	Class diagram of the GeoXACML combining algorithms <code>and</code> and <code>or</code>	176
6.1	Unveiling of restricted information without permission	190
6.2	Software limitations for requesting a map for a restricted areas	192
6.3	Applying an obligation image to the WMS map	192
6.4	Transparent chaining of protected services	194
6.5	Translucent chaining of protected services	195
6.6	Opaque chaining of protected services	195
A.1	Conventions for diagrams, taken with Altova XML Spy	203
B.1	The City Model map	205

List of Tables

2.1	XML Schema constructus for object-oriented markup	14
2.2	Xpath predefined location path and node set functions	17
2.3	Xpath 1.0 example expressions	18
2.4	Geospatial information objects of the City Model	29
2.5	Semantics of the combining algorithms	48
3.1	Requirements for access control standards	59
4.1	Xpath expressions for class-based restrictions	75
4.2	Xpath example expressions for object-based restrictions, based on the City Model example, 2.3.2, page 28	76
4.3	Truth table for the logical AND/OR combination of rule outcomes	79
4.4	Semantics of the GeoXACML combining algorithms <code>or</code> and <code>and</code>	79
4.5	Which spatial methods can be used to determine a particular topological method	85
4.6	Evaluation of matching criteria for permissions	104
4.7	Worst case classification for the topological relations <code>Within</code> and <code>Within</code>	121
4.8	Worst case classification for the topological relations <code>Within</code> and <code>Touches</code>	126
4.9	Worst case classification for spatial relations <code>Touches</code> and <code>Touches</code>	130
4.10	Metadata for the Services A and B (figure 3.1)	146
5.1	GeoXACML structured data types for some simple geometries	154
5.2	GeoXACML spatial functions	155
5.3	URI, class name and geometry data type for the spatial attributes	173
5.4	Test cases for the evaluation of class-based restrictions	178
5.5	Test cases for evaluation of object and class-based restrictions	179
5.6	Test cases for evaluation of spatial restrictions using the spatial relation <code>Within</code>	181
5.7	Test cases for evaluation of spatial restrictions using the spatial relation <code>Touches</code>	182

5.8	Test cases for evaluation of complex spatial restrictions using the combining algorithm and	185
5.9	Test cases for evaluation of complex spatial restrictions using the combining algorithm or	186

List of Listings

2.1	XML Schema representation of the object-oriented example data model	15
2.2	XML representation of a data collection of the object-oriented data model example	16
2.3	GML encoding of the simple geometry <code>Point</code>	26
2.4	GML encoding of the simple geometry <code>LineString</code>	26
2.5	GML encoding of the simple geometry <code>LinearRing</code>	27
2.6	GML encoding of the simple geometry <code>Polygon</code>	27
2.7	Simple GML application schema	28
2.8	GML feature collection snippet of the city model	31
2.9	WMS GetMap requests for retrieving the map from figure 1.1	36
3.1	XML Schema definition of the <code><Condition></code> element	65
3.2	XACML <code><Condition></code> element example	66
3.3	BNF description of the tree representation of the <i>Authorization decision</i>	67
3.4	Associating a permission to a role	67
3.5	Declaring permissions for a role hierarchy in XACML	68
4.1	Resource content example, comprising 'real' information objects	82
4.2	Resource content template example, comprising 'template' information objects	83
4.3	Resource content template for the City Model	105
4.4	Encoding of an environment attribute value that defines the used CRS	144
4.5	Service operation selector for associating policy sets to Service A's operation <code>DescribeFeatureType</code>	146
4.6	Rule matching to ensure CRS correlation	147
5.1	Encoding of a 2-D <code>Point</code> geometry as a XACML <code><AttributeValue></code>	154
5.2	GeoXACML encoding of a spatial condition	155
5.3	XACML encoding of the subject, identified as Bob	156
5.4	XACML encoding of the operation <code>read</code>	157
5.5	Example XACML encoding of a class-based restriction using <code>string-equal</code>	157

5.6	Example XACML encoding of a class-based restriction using a tag count . . .	158
5.7	Example XACML condition for selecting one specific information object . . .	158
5.8	Example XACML condition that matches all but one specific information object	159
5.9	Example XACML condition, encoding of an object-based restriction	160
5.10	Example GeoXACML Rule expressing a spatial restriction	161
5.11	Example for a WFS Insert operation	165
5.12	XACML authorization decision request for the WFS Insert operation	166
5.13	XML Schema definition for a WMS resource content	167
5.14	WMS example request for the operation <code>GetMap</code>	168
5.15	XACML encoded authorization decision request for the operation <code>GetMap</code> . .	169
5.16	WMS example request for the operation <code>GetFeatureInfo</code>	171
5.17	GeoXACML encoding of an authorization decision for a WMS example request for the operation <code>GetFeatureInfo</code>	172
5.18	Code segment for the main class	173
5.19	Code segment of the <i>PointAttribute</i> constructor	174
5.20	Selection of the root node of the resource content	175
5.21	Code segment for the processing of the nodes, matching the Xpath expression	176

Chapter 1

Introduction and Motivation

This chapter frames the problems of interoperability for online access to distributed geodata and their combined use as it outlines the infrastructure for this work. It also highlights the requirements for a distributed access control system that allows the declaration and enforcement of access restrictions to protected resources; geospatial information objects. This chapter gives a motivating example that will be reused throughout this work to illustrate theoretical concepts. It closes with the explanation of the scientific methods for this dissertation, which are similar to the procedures for software development: requirements collection, evaluation of capabilities of existing standards and systems, developing of a solution and finally evaluation and system design for that solution.

Geodata is an important information base for decision making in simple and complex work processes in different fields of use. The term geodata is an abbreviation for geospatial information¹. It is a synonym for information with a relative location to the surface of the Earth.

Different data providers, like government agencies, organizations and private companies, collect and maintain geodata of different kinds, such as satellite images of the weather and surface conditions, land parcel information as well as street and topographic maps. Organizations and private companies may collect geodata related to their field of work such as their infrastructure of oil pipelines, radio link antennas or electricity cables.

In particular, the combination of specific kinds of geodata is the key for enabling and supporting complex decision making in the first place. One example for such decision making is whether to issue a gale warning for certain areas of the Gulf of Mexico. This decision making is based on the combined use of this information: Populated areas, political boundaries and coastlines for the area of the Gulf of Mexico and a weather map that shows the gale intensity and movement. The decision making requires the combination of geodata from multiple data providers; one for populated areas and the coastline and another for the weather maps. This combination in form of a map is shown in figure 1.1 [OGC 2000-028, p. 6].

Each data provider can restrict the access to their own geodata for different reasons such as licensing issues or commercial aspects. Data users may obtain access rights, which allow designated use of the protected geodata.

¹Some publications also refer to it as geospatial, spatial data

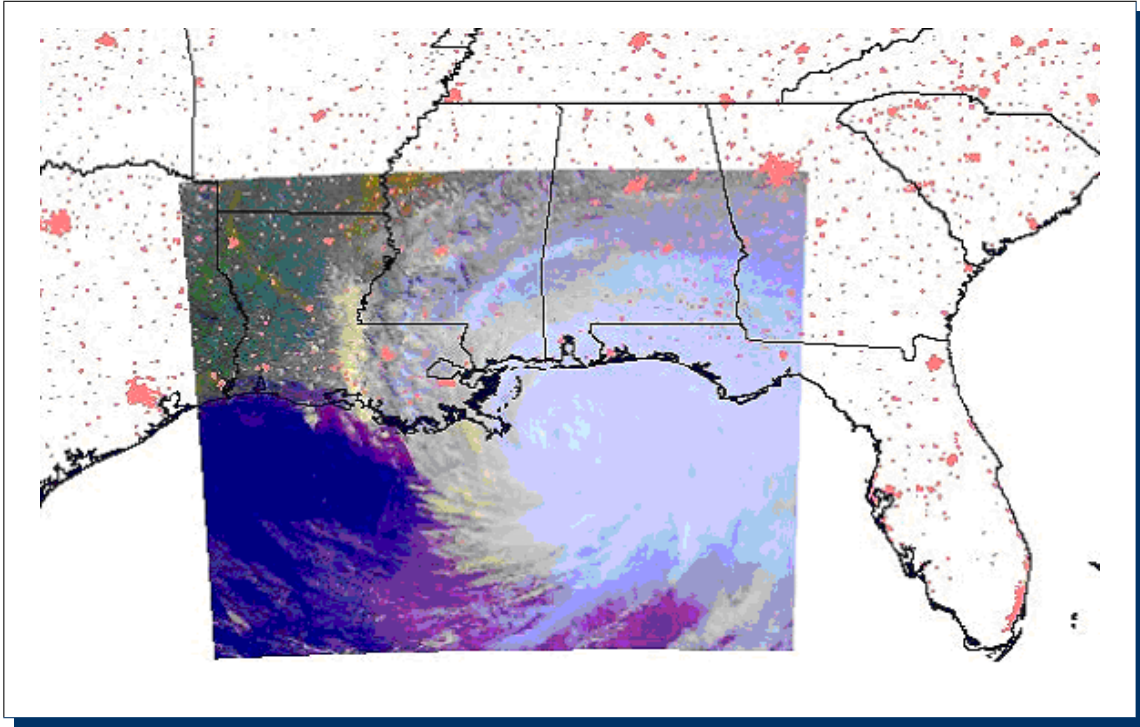


Figure 1.1: Political boundaries, coastlines, and populated areas of the Southeastern United States and a NOAA hurricane image of the Gulf of Mexico [[OGC 2000-028](#)]

1.1 Interoperable Use of Distributed and Heterogenous Geodata

Different kinds of geodata are collected for different purposes by independent data providers. The major requirement for the combined use of geospatial data is its interoperability. But, the data format and model depend on the characteristics of the geodata and the intended use. Both can be specified by the data provider, independent from each other. This results in an initial situation of distributed and heterogenous geodata, as illustrated in [[Donaubauer 2004](#)].

For the combined use, this initial situation can be overcome by data integration. Geodata integration is the procedure that enables the combined use by bringing different kinds of geodata together for further processing. In [[IDC 2003](#)] data integration is described that it *‘takes many forms, from simple file transfers to virtual database platforms. According to IDC, data integration software attempts to provide nonnative, programmatic access to persistent structured data, whether in heterogeneous, homogeneous, distributed, or centralized data sources.’* IDC give three reasons for data integration: (i) Provide an integrated view of data, (ii) Allow multiple applications to behave cooperatively and harmoniously and (iii) Improve operational efficiency of the IT department.

They further highlight different integration strategies, from which two are important to mention in the context of geodata integration.

The consolidated data integration brings all different kinds of geodata together into one single database; the geodata repository. Any processing and maintenance of the data requires

access to this central geodata repository. This strategy is typically supported by traditional Geo Information Systems (GIS).

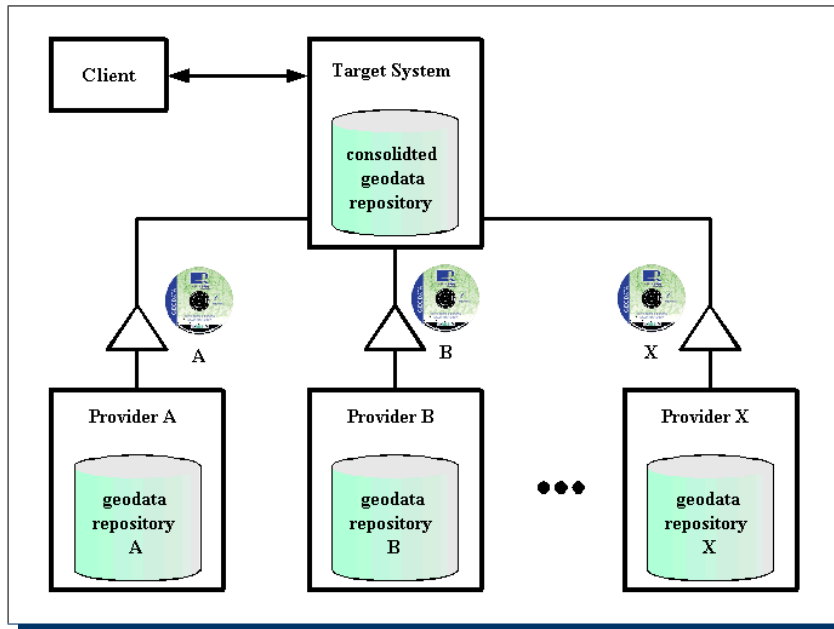


Figure 1.2: Consolidated data integration brings together different geodata kinds into one Target System

The federated data integration approach leaves the geodata in their original storage, where it is maintained and updated. The processing is possible through a virtual database schema that represents a mediated schema. This schema allows a common use of the data, as it is transformed to the appropriate structure of the federated data storages, before the actual access takes place. In such respect, the mediated schema hides the original data structure and model from the end user.

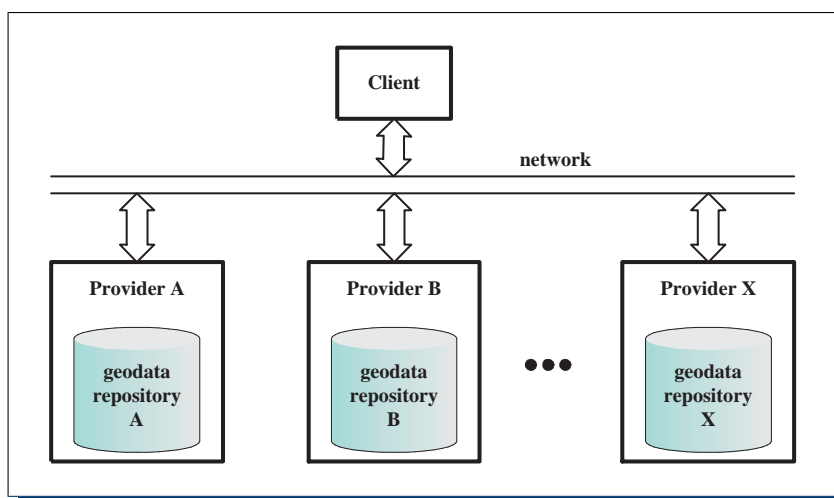


Figure 1.3: Federated data integration provides online access to distributed and heterogeneous geodata through mediator schema

Both approaches have pros and cons. The consolidated data integration can use geodata that has been shipped on CD from the different data providers to the end user's site. Here, an administrator has the duty to perform the data integration by importing the shipped data into the data store of the so-called Target System, which is typically a GIS. One disadvantage - among others- is that the geodata of the Target System is never up-to-date. This disadvantage is the major advantage of the federated integration, where the processing takes place on the up-to-date data.

One implementation of the federated data integration is based on Internet Technology. The data providers set up an Internet access point, called a Web Service that provides well defined access to the federated geodata. If the service supports geodata specific processing functions like mapping, intersection of geometries or coordinate reference system transformation, it is called a Geo Web Service (GWS). A simple black box view of a Geo Web Service can characterize it as a processing unit that combines processing functions and geodata for a particular field of use.

The interoperable use of geodata, obtained from a Geo Web Service relies on the basic concept that the output of the service can be controlled by its input parameters. This allows the decoupling between the original (internal) format and model as it is used by a data provider and an interoperable mediator data format, which is offered by the Geo Web Services. In short, this attempt is based on online and on time data transformation and not on exhaustive and permanent data storage.

One limitation of the federated data integration results from the required network. Geo-processing can require large amount of data that must be sent to the service or received from the service. Thus, the processing speed relies on the network capacity.

In order to use an online infrastructure of distributed services, which provide access to federated geodata, distributed access control is required. This ensures that a data provider can enforce the desired geodata protection.

1.2 Geodata and Geospatial Information Object

As already stated, geodata is collected and maintained in different data formats and data models, according to the purpose of processing. Some kinds of geodata are inappropriate to have a structure. For example, high volume image data as collected from weather satellites is raster-based geodata. It is not predestinated to exist in any data model. But other phenomena that can be observed and measured through specific (geodetic) equipment, can be modeled in an object-oriented model. For example, a street can be represented in an object-oriented data model.

According to ISO 19107, an information object -in this context- is an abstraction of a real world phenomenon. The information object is called a geospatial or spatial information object if associated with a location relative to the surface of the Earth.

According to object-oriented modeling, an object is an instance of a class that defines the properties of the object. This representation of information has the advantage in reference to the raster-based information that an object can be uniquely identified and automatically be processed. The processing of raster data mainly requires manual processing.

The object-oriented approach has another big advantage that allows to combine information that belongs together, define operations for processing and model relationships between the objects. This brings the tremendous advantage that in contrast to the previous case, a machine can distinguish between objects of either class and a human is not required looking at the graphical representation. Also, the object-oriented approach allows to give each instance of a class (an object) a unique identity where further processing can be based on, even if the values of the properties change. Based on this unique identification, it is possible to express relations between objects. These relations can be of spatial or non-spatial kind. An example for a non-spatial relation is the ownership relation; an example for a spatial relation is the adjacent relation, grounded on the geometries of a spatial information objects.

The object-oriented modeling of geospatial information objects requires the presentation of the geospatial characteristics. The location and shape of a geospatial information object is described by geometry.

1.3 Protected and Distributed Geospatial Information Objects

In general, the use of geodata might be restricted due to licensing issues or electronic commerce. Each data provider can declare permissions, which express the restrictions.

Requirements for declaration and enforcement of access restrictions in the context of geodata have been collected at an informal poll at the InterGeo 2002². The poll assumed that distributed and heterogenous geodata is available for online access via the introduced infrastructure of Geo Web Services. The collected requirements can be separated into four categories:

- (i) Requirements for **security** based access restrictions result from the a priori insecure Internet communication. The data provider argue that depending on the kind of the geodata a user likes to access, different security conditions like confidentiality, integrity and authentication must be met. If the declared conditions are not met, the request is rejected. As an example, the request of personal information such as ownership information of buildings requires security endurance, as stated in the law. If such a request is made via the Internet, security must be applied to the communication from the provider to the consumer. Another argument is that the data/service provider generates a certified result. It must be ensured that this result has not been tampered unnoticed during transmission from the data/service provider to the consumer. An example exists in this field of road work, where a digging request is made for a particular area to check for existing gas pipelines or electricity cables. It must be guaranteed that the 'no dig' result information has not been changed unnoticed to 'dig'.

Even though this is a very interesting topic, it will not be considered in further detail.

- (ii) Requirements for **service** based access restrictions focus on restricting the functionality of a service. One addressed requirement is to restrict a particular interface only to clients

²The InterGeo is a German trade fair that is held annually. Geodata provider and user, hardware and software providers of the Geoinformatic sector meet, in order to show their most recent and mature products.

from the local network and not to clients, connecting from the Internet. Another aspect is the time and day, when a service function can be invoked; resp. not to be invoked. As an example, a particular service interface can be invoked from all clients, connecting from a local network between the business hours, Mon-Fri 8 a.m. to 5 p.m.

Also, invocation parameter dependent restrictions have been named for mapping requests: The request of a binary (raster) map is free, if the scale is not finer than 1:5000; otherwise the request is restricted. If the map is requested in a vector format, which usually allows further processing, the map request is also restricted. But, different access rights are required to access a restricted binary map and a restricted vector format map. Also restrictions are desired, which depend on the used rendering style of the map. For example, if the map is rendered in black/white style, it is free but if rendered in color style, it's restricted.

These types of requirements are not the prime focus of this work. However, they can also be declared and enforced with the provided solution. This is based on the access control capability to express conditions.

- (iii) Requirements for **content** based access restrictions result from the object-oriented data model. The interviewed people said that it is important to express existing restrictions on the geodata for the online access; hence on the service level. In such a respect, the content is the information that either represents the input or the output of a service. For particular service interfaces, which allow the modification of the underlying geodata, the authorization must be checked before the actual access takes place. But for other interfaces, the authorization can only be checked after the result is generated. This is different from the traditional approach, where the authorization is always checked prior to execution.

Assuming an object-oriented data model on the service interface level, the 'content' comprises of information objects which are instances of a particular class. This results in the class-based and object-based restrictions.

A declared **class-based** restriction is related to a particular class and must be enforced for all objects, which are instances of that class. For example, an access restriction declared for the class **Building** must be enforced for the objects "The White House" and "The Dome of Cologne", assuming both are instances of that class.

An **object-based** restriction must be enforced for all objects, for which the specified characteristics match. Different possibilities exist for referencing: One attempt can use the identify of the object or the values of characterizing properties. Assuming the uniqueness of the identity, the restriction is enforced for the lifetime of the object. If using modifiable object properties for the declaration of an access restriction, the same restriction can apply to different objects for subsequent requests. For better readability and consistency, it was desired that a declared restriction can refer to one object or a set of objects. As an example, a restriction that identifies instances of the **Building** class by the string "The White House" does not match "The Dome of Cologne"; but the reg.-expr. string "*" matches both instances.

This topic is the first prime topic, covered by in this work.

- (iv) The request for **spatial** restrictions result from the geospatial characteristics of the information objects. Most interviewees highlighted this requirement as very important

and stated that it must be possible to declare spatial access restrictions to geospatial information objects, depending on their location and geometry. This shall be achieved by declaring a specific spatial condition that expresses the spatial relation between the geometry of requested geospatial information objects and a 2-D geometry for which the restriction is to be enforced.

This topic is the second prime topic, covered by this work.

The developed access control model, introduced in chapter 4, supports the declaration and enforcement of class-based and object-based access restrictions as well as 2-D spatial restrictions. The object-oriented model is based on the Simple Features specification ([OGC 1999-049]) and the Feature Geometry model, introduced in [OGC 2001-101].

1.4 A Motivating Example

The purpose of the following example is to show the need for a distributed access control that allows to declare and enforce access restrictions as highlighted earlier. For this purpose, a real world process (“Building Permit Process”) is simplified as necessary to focus on the main aspects: Restricted access to distributed heterogeneous and protected geodata resources. The example process (“parcel splitting and change of ownership”) involves different types of access to different kinds of geodata. The example data model is simplified from the real world data model in order to be useful for illustrating theoretical concepts throughout this work.

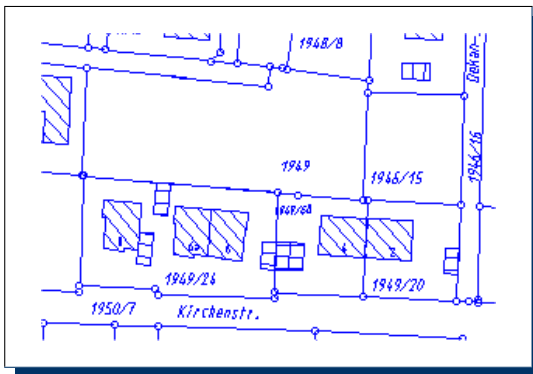


Figure 1.4: Initial situation before land parcel splitting

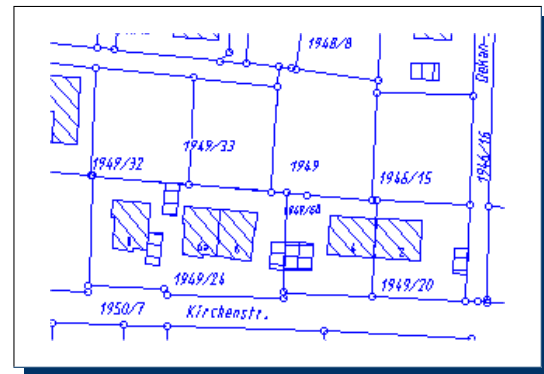


Figure 1.5: The resulting situation after land parcel splitting

The process is initiated if one or more land parcel owners like to open a construction site, let's say for building private houses. Their large land parcels must be split, sold and the new owner must be registered in the cadastre. On the newly created land parcels, the new owners can then build their own houses. Figure 1.4 shows an initial situation, where a large land parcel (no. 1949) is about to be split. The resulting situation is shown in figure 1.5, where inside the land parcel no. 1949, new land parcels with the no. 1949/32 and no. 1949/33 have been created³.

Looking at the process in more detail, it comprises of multiple sequential steps, as illustrated in figure 1.6. The first step is to lock the land parcel, which is to be split so that it

³For simplicity, the ownership information of the land parcels is not displayed.

cannot be modified meanwhile. In the example, the land parcel, identified by no. 1949 is locked. This is achieved by using the lock operation of the GWS Land Parcel. The second step involves the creation of the new land parcels, identified by the no. 1949/32 and 1949/33. The create operation of the GWS Land Parcel is used to do so. After creating the land parcels, the ownership information is updated by using the write operation of the GWS Land Parcel and Cadastre. The last step in the work flow is to update the geometry of land parcel 1949 and unlock it. This can be achieved, using the write and unlock operation of the GWS Land Parcel.

The detailed view on the work flow unveiled the requirements for distributed geodata. The data model that exists for the land parcel information is linked with the data model that exist for the ownership information. In order to set the ownership as a double sided relation, a mechanism must exist to uniquely refer to a person, which owns one or multiple land parcels. But, it must also be possible to refer to the land parcel(s) uniquely from the cadastre in order to associate ownership information in that data model. In the introduced infrastructure of online access, interfaces allow unique identification of resources, made available by the service, independent from the internal data structure and data model. One existing approach, how to assign a unique identity to resources is introduced in [Bishr 1999]. One approach for this service based infrastructure is to prefix the local identities with a unique Uniform Resource Name (URN). The service simply removes the URN, in order to get the original local identity.

In [Schlichter 2000], the unique naming of software agents is defined using a global name and a network address. The name has the email format, where the local name of the email represents the name of the agent and the global part represents the domain name of the agent. The network address represents the URL, which can be used to execute the agent. Adopting this to the given problem of unique resource identifiers, the URN can be created from the unique name of the service. For example, resources which are accessed via the service `http://foo.xyz` can use this identifier as a unique URN.

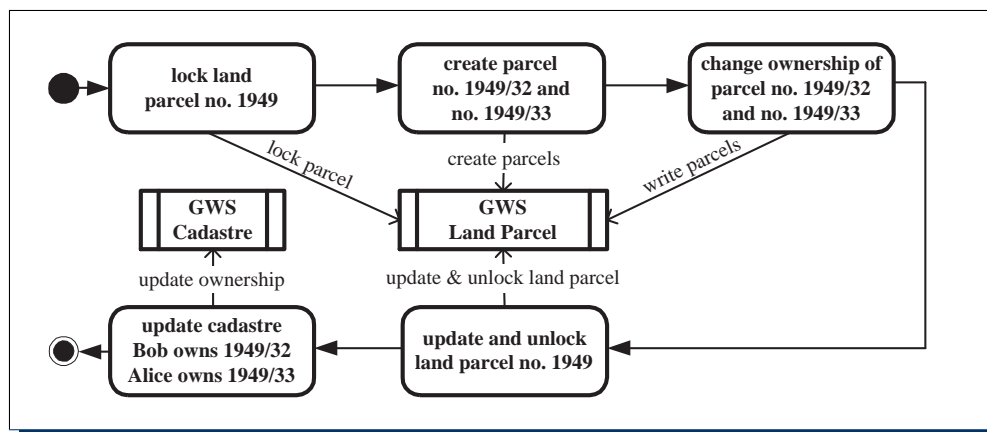


Figure 1.6: Workflow for splitting a land parcel and updating the ownership information

For further reference, it is assumed that the required geodata is modeled in an object-oriented way and exist in one data model⁴. However, separating the cadastre and the land

⁴The use of one data model simplifies the example for further use

parcel into different data models is not too difficult: The two-sided ownership relation must be represented as a reference to an information object of another data model and not as a reference to an information object, belonging to the same data model.

This process is based on the land parcel information (Ger. ‘‘Digitale Flurkarte’’) and the cadastre information about ownership (Ger. ‘‘Grundbuch’’). Assuming that Geo Web Services provide an interoperable access to the geodata, both kinds of geodata can be used together. The underlying data model must ensure that unique relations exist between a land parcel and its owner.

An abstraction of the real world data model is to represent the land parcel information in an object-oriented model. Using an object-oriented data model, it is possible to combine the required geodata in an application specific way. A land parcel can be modeled as a geospatial information object by the class `Landparcel` that has properties, which hold the border line, the identification number and a reference to the owner. The owner can be modeled as a non-spatial information object⁵ by the class `Person`.

The object-oriented model is also important for the declaration of access restrictions, because class- and object-based restrictions are based on that. In Germany, the access to the cadastral information is primarily restricted by existing law regulations⁶.

The following access restrictions can be envisioned for the example process:

- (i) read access on the class `Landparcel` is associated to `Bob` for an area that contains the requested land parcel (e.g. area of figure 1.4). This can be declared using the class-based and spatial restrictions.
- (ii) write and create access on the class `Landparcel` is associated to `Alice` for the area of the original land parcel (no. 1949). This area is determined by the `boundary` property of the `Landparcel` object. With the continuous splitting, the area of the original land parcel gets smaller and smaller. But because the access restriction refers to that boundary, it is always correct. Declaring an independent area based on the original land parcel shape would be incorrect, after the first split. This can also be declared using the class-based and spatial restrictions.
- (iii) lock and unlock access on the land parcel object (no. 1949) is associated to `Alice`. In contrast to `Bob`, she has the appropriate job assignment. This restriction can be declared using the object-based restriction.
- (iv) write access to the cadastre (class `Person`) is associated to `Joe`. This can also be declared using the class-based restriction.

An alternative to the previously described process is to obtain the required geodata (the content) from the designated services by online access. Further work uses the offline content, until all modifications are done. As the final step, the original content is updated via online access. This alternative process has the advantage that the online access and therefore the network load is minimized. The frequent access to the content -creating new land parcels and associate new ownership- is done offline. This process is to be favored if either many small

⁵No geometry is associated to a person in this model

⁶This might be different in other countries but Germany.

land parcels are to be created, in order to save network capacity or if the process requires a longer period of time as it can be expected with real private construction sites. In such a case, an engineering bureau can obtain the content and work on it on behalf of a government agency. This work flow is illustrated in figure 1.7.

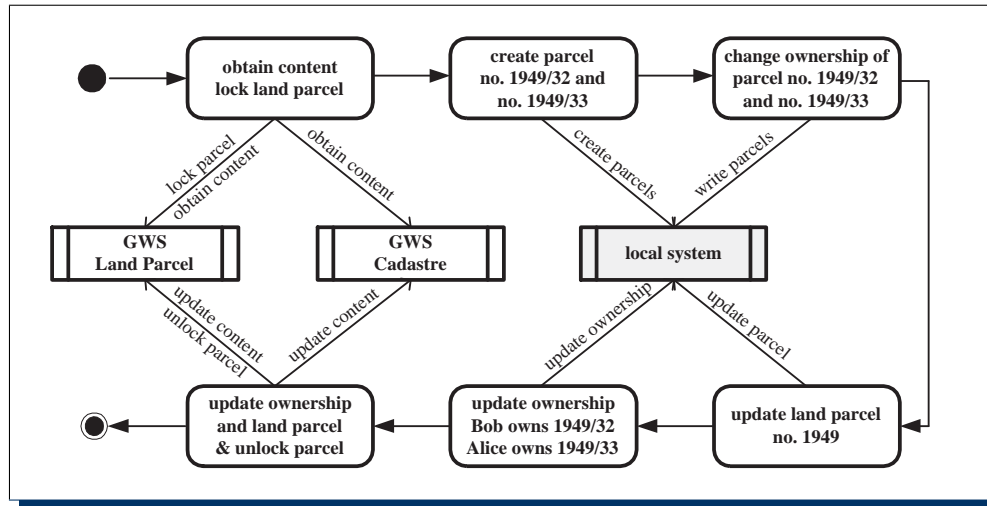


Figure 1.7: Alternative process for land parcel splitting and modification of ownership

The process starts, when the engineering bureau -in person of an employee- obtains the necessary geodata content. The content comprises of land parcel and ownership information. All required actions to split the land parcel and associate new owner is achieved using a local system. The last step in this alternative process is to update the original data stores and unlock existing locks. The same set of declared access restrictions apply to this alternative process. However, the enforcement is different. The enforcement in the previous example is limited to the monitoring of the online access. With this approach, it requires the enforcement of operations to an offline content. One strategy to enforce declared access restrictions on an offline content can be achieved by using the principles of the Digital Rights Management (DRM). In a nutshell, DRM relies on the encryption of the content, before transmitted from the provider to the user. Access to the (offline) content is possible by requesting a license, which includes a key for decryption. Assuming an appropriate encryption, the access restrictions can be enforced by controlling the license delivery. The user must specify the intended use and the access control derives a decision if the user has the rights to receive the requested key. For example, a user of the engineering bureau requests a license to perform create operation inside the area of the land parcel no. 1949 to create new land parcels (create access to the class `Landparcel`). If the license is issued, the local system must enforce them.

1.5 The Scientific Method of this Dissertation

This dissertation aims at the development and implementation of an access control system that can be used to declare and enforce the introduced restrictions: class-based, object-based and spatial restrictions. The scientific method, applied to this work, is similar to the software development and engineering process. First, requirements for an access control have been

collected in the field of distributed geoprocessing. For the collected requirement and the constraint of a distributed environment, existing standards and systems have been evaluated. The outcome of this evaluation was that no standard or system exists, which can be used to implement the requirements. This was the point to start developing a model that has the capabilities to implement the requirements. The model was developed according to/based on existing standards, which fit best into the initial situation. The final stage was to evaluate the developed model by implementation and testing.

This work is structured according to this modus operandi of software engineering. Chapter 1 (this chapter) gives a brief introduction to the initial situation and highlights the need for an access control system that supports the declaration and enforcement of class- and object-based restrictions as well as spatial restrictions. Chapter 2 introduces basic concepts, important to understand this work. Chapter 3 evaluates relevant standards and system in the field of distributed access. The chapter concludes with drawing a comparison and argument, why existing standards and systems are not capable to be used for implementing the requirements. Chapter 4 introduces the model for declaration and enforcement of the outlined access restrictions for the distributed environment. It introduces different structures for organizing existing access permissions in the context of deriving an authorization decision. It focuses on the problems of incomplete or incorrect policy sets and introduces approaches to avoid or to deal with these problem. Chapter 5 frames the evaluation of the developed access control and describes the implementation of the authorization process as an extension to the XACML standard. This work finishes with chapter 6, which covers the conclusion and outlook.

Figure 1.8 illustrates the possible reading paths of this work. Chapter 4 consists of two parallel paths: One covers the declaration and the other the enforcement of access restrictions.

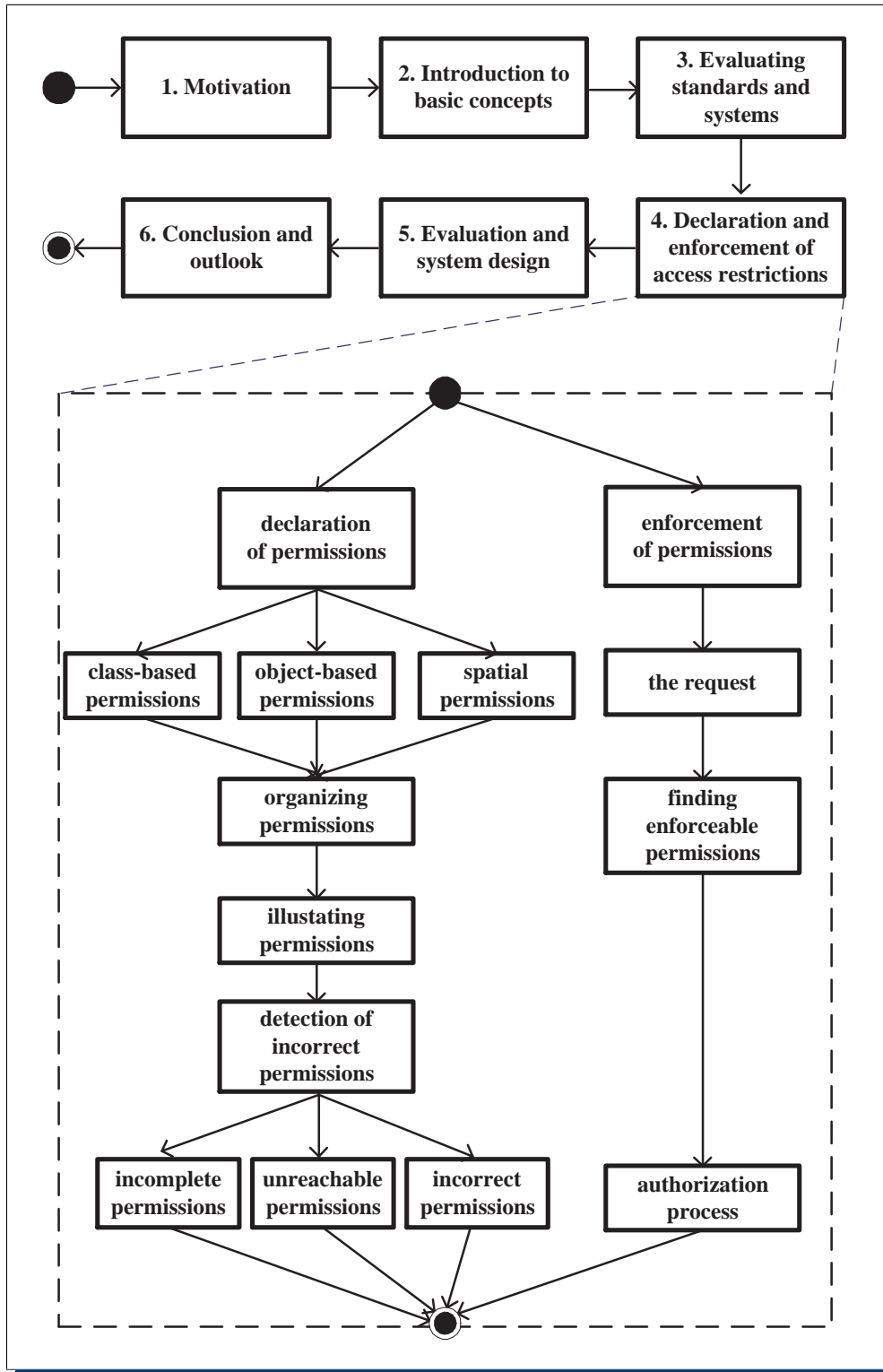


Figure 1.8: The structure of this dissertation.

Chapter 2

Basic Concepts

This chapter introduces the relevant concepts and standards for this work in a brief manner as it is relevant to understand this work. For advanced reading, the interested reader is referred to the original documents and standards, as cited in chapter Bibliography.

This second chapter also introduces own formalizations in order to describe existing concepts as they are being used throughout this work.

2.1 XML, XML Schema and Xpath

The eXtensible Markup Language (XML) ([W3C 2001b]), XML Schema ([W3C 2001d] and [W3C 2001e]) and Xpath ([W3C 1999]) are important W3C standards, used in this work. This section gives a brief introduction to understand the basic concepts and capabilities.

2.1.1 Extensible Markup Language (XML)

The “Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.” [W3C 2001b]).

In short, XML is a markup much like HTML but it was designed to describe data (focus on what the data is) and not how the data is presented. This is the distinguishing difference to HTML, which was designed to focus on the representation of data (how data looks like). XML uses tags (elements) for the markup, which are not predefined. The tags and the structure of the tags are defined by a Document Type Definition (DTD) or an XML Schema.

An XML document that does not refer to an XML Schema is well formed, if the structure obeys to the grammar of XML. An XML document is valid if the structure obeys to a referenced XML Schema.

2.1.2 XML Schema

XML Schema is an XML based successor of the Document Type Definition (DTD). XML Schema describes the markup of an XML document: Its structure, the elements (tags) that can occur in the structure and representation of the elements itself. An element can have attributes and properties, which are represented by child elements. XML Schema defines common capabilities to declare constraints on the XML document structure: The order and number of child elements, the type of the elements and attributes. In addition, XML Schema supports the object-oriented approach of inheritance through extensibility. It also supports the concept of namespaces, which make the unique identification of elements possible. In short, XML Schema defines an XML markup for an object-oriented data model.

Object-oriented construct	XML Schema equivalent
class / class-name	<complexType name="class-name" >
inherit class-name	<extension base="class-name" > as a sub-element of <complexType>
property / property-name	<element name="property-name" > or <attribute name="property-name" > as a subelement of <complexType>
object / object-name	<element name="class-name" > defined as global element

Table 2.1: XML Schema constructus for an XML markup of an object-oriented data model

Table 2.1 contains a lineup of XML Schema and object-oriented constructs. XML Schema provides the element and attribute constructs to represent class properties. As a rule of thumb, an element represents object characteristics, which are useful to a user, whereas the attribute holds metadata about the object, which is required for machine processing. One example of the attribute use in such respect is the unique identification of an object.

The following example provides an XML Schema representation of a simple object-oriented data model, which is shown in figure 2.1. The data model contains of the two classes `InformationObject` and `Person`. The `Person` class inherits from the `InformationObject` class. The `InformationObject` class has one property named `id` and the `Person` class has the characterizing property `name`.

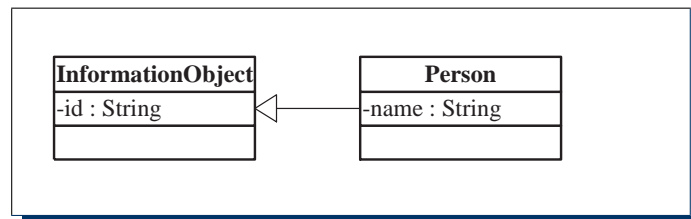


Figure 2.1: Object-oriented data model example

One possible XML Schema representation is shown in listing 2.1. The classes `InformationObject` and `Person` are represented as global elements and the `complexType` construct is being used to enable inheritance. The XML Schema requires to define a – so-called – root element, which becomes the container of the XML document that holds all instances of

marked-up objects. In this example, the root element has the name `OOExample` and it holds a sequence of objects of class `Person`. The example uses the namespace `am` that refers to `http://www.in.tum.de/am`. This is encoded using the root element's `xmlns` attribute.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema targetNamespace="http://www.in.tum.de/am"
3   xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:am="http://www.in.tum.de/am"
4   xmlns="http://www.in.tum.de/am" elementFormDefault="qualified"
5   attributeFormDefault="unqualified">
6   <xs:element name="OOExample">
7     <xs:annotation>
8       <xs:documentation>
9         An example representation of an object-oriented data model
10      </xs:documentation>
11    </xs:annotation>
12    <xs:complexType>
13      <xs:sequence maxOccurs="unbounded">
14        <xs:element ref="am:Person"/>
15      </xs:sequence>
16    </xs:complexType>
17  </xs:element>
18  <xs:element name="InformationObject" type="am:InformationObjectType"/>
19  <xs:element name="Person" type="am:PersonType"/>
20  <xs:complexType name="InformationObjectType">
21    <xs:attribute name="id" type="xs:string" use="optional"/>
22  </xs:complexType>
23  <xs:complexType name="PersonType">
24    <xs:complexContent>
25      <xs:extension base="am:InformationObjectType">
26        <xs:sequence>
27          <xs:element name="name" type="xs:string"/>
28        </xs:sequence>
29      </xs:extension>
30    </xs:complexContent>
31  </xs:complexType>
32 </xs:schema>

```

Listing 2.1: XML Schema representation of the object-oriented example data model (`OOExample.plr.xsd`)

One example of a data collection is shown in listing 2.2. The data collection is comprised of two objects, being instances of the class `Person`. The first instance may be identified by the lifetime identity (`id=P1`) or through the characteristic name (`name=Bob`).

2.1.3 Xpath

Xpath is a syntax to fetch parts of an XML document. Xpath is based on the UNIX path like expressions to refer to elements, which are separated by a slash (`'/'`). A path expression with a leading slash represents an absolute path, starting at the root node. A path expression with no leading slash refers to a relative path, starting at the currently selected node. It uses a tree representation of the XML document. The top node of the tree represents the document root, referenced by the path `'/'`. The nodes of the tree can be traversed by using a Xpath expression. The path `'/A/B'` refers to the element named `B`, which is a child element the root element `A`. The path expression can fetch a particular set of elements, using a double-slash `'//'`. The expression `'//A/B'` selects all sub-elements of the root element `A`, named `B`.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <OOExample xmlns="http://www.in.tum.de/am"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.in.tum.de/am OOExample.xsd">
5   <Person id="P1">
6     <name>Bob</name>
7   </Person>
8   <Person id="P2">
9     <name>Alice</name>
10  </Person>
11 </OOExample>

```

Listing 2.2: XML representation of a data collection of the object-oriented data model example

Xpath supports the declaration of Boolean expressions to test particular circumstances like the existence of particular elements, the unary of elements or if an element has particular characteristics, represented by sub-elements and attributes. Xpath also defines a library of functions that can be used with the path expression. The most important functions are the node set functions, which allow particular traversal of the tree and fetching of (string) values from elements and attributes. Table 2.2 presents a sub-set of Xpath 1.0 functions, which are important for the declaration of access restrictions in XACML.

For the previous example content of the object-oriented data model from figure 2.1, table 2.3 represents different Xpath expressions for fetching elements under different constraints.

2.2 Geodata

Geodata is geographic information about objects and phenomena with a location relative to the surface of the Earth. It is collected and maintained by different government agencies, organizations or commercial companies (data providers) for numerous purposes. The purpose pre-defines the format and the data model of the geodata. For advanced geoprocessing, geodata must have an object-oriented structure.

2.2.1 Geospatial Information Object

Many definitions of an information object exist. In this context, an information object is a representation of a real world entity. It comprises of a set of properties, holding the characterizing information and relations to other objects. This data model allows to query for the owner of a land parcel by evaluating the value of the according property.

A geographic, geospatial or just spatial information object is a specialization of an information object. It represents a real world entity that has a location, relative to the surface of the Earth. Its geographic characteristics such as the location and shape are encoded by geometry. The geospatial information object can hold explicit and implicit geospatial relations through their geometry. Making geospatial relations explicit is covered by topology, which

²The function `position` and `count` are added to the cited table.

Function	Argument	Description
ancestor	element-name	returns all element-name ancestors of the context node
ancestor-or-self	element-name	returns all element-name ancestors of the context node and if the context node is an element-name element, the context node as well
descendant	element-name	returns all element-name descendants of the context node
descendant-or-self	element-name	returns all element-name descendants of the context node and if the context node is an element-name element, the context node as well
following	element-name	returns all the element-name nodes that are after the context node in the document order, excluding any descendants, attribute nodes, and namespace nodes
following-sibling	element-name	returns all the following element-name siblings of the context node
preceding	element-name	returns all the element-name nodes that are before the context node in the document order, excluding any ancestors, attribute, nodes, and namespace nodes
preceding-sibling	element-name	returns all the preceding element-name siblings of the context node
parent	element-name	returns the parent of the context node if there is one and it is an element-name element, and otherwise returns nothing
child	element-name	returns all element-name elements children of the context node
self	element-name	returns the context node if it is an element-name element, and otherwise returns nothing
attribute	attribute-name	returns attribute-name of the context node
namespace	namespace	returns the namespace nodes of the context node
position	N/A	returns the position in the node list of the node that is currently being processed
count	element-name	returns the number of nodes in the argument node-set
name	element-name	returns the name of the tag

Table 2.2: Xpath predefined location path and node set functions [Damiani et. al. 2002, p. 182]²

is not covered by this work. However, this work uses spatial methods, which can test two geometries for a given topological relation.

In order to model the adjacency relation between land parcels from the example explicitly, the data model can be extended. A property like `adjacentTo` can be added to the class `Landparcel`, which is of type sequence and holds the identifiers of adjacent land parcels. In order to express that land parcel 1949/33 is adjacent to 1949/32, 1949/24 and 1949, the ‘`adjacentTo`’ property of 1949/33 holds the values ‘1949/32’, ‘1949/24’ and ‘1949’. However, the adjacency relation can also be determined dynamically by processing the geometries of

Xpath expression	Return type	Explanation	Xpath result
/OOExample/*	node-list	returns a node-list of all ancestor nodes	<Person id="P1">, <Person id="P1">
/OOExample/Person	node-list	returns a node-list of all ancestor nodes	<Person id="P1">, <Person id="P1">
/OOExample/Person/*	string-list	returns a node-list of all ancestor nodes	<name>="Bob", <name>="Alice"
/OOExample//*	node/string-list	returns a node-list of all decedent nodes	<Person id="P1">, <Person id="P1">, <name>="Bob", <name>="Alice"
/OOExample/Person[1]	node	returns the first node named <Person>	<Person id="P1">
/OOExample/Person[1]/name	string	returns the value of the node <name> of first node named <Person>	Bob
/OOExample/Person[@id='P2']/name	string	returns the value of the node name of node Person, identified by the attribute id, which value is "P2"	Alice
(/OOExample/Person[@id='P2']/name)="Alice"	boolean	returns the boolean result (true) of expression	true
count(//Person)	integer	returns the count of elements representing instances of class Person	2
count(//Person[@id='P1'])	integer	returns the count of instances of class <i>Person</i> with the id equals "P1"	1
name(//Person[1])	string	returns the tag name of the first element matching the Xpath	<i>Person</i>

Table 2.3: Xpath 1.0 example expressions for fetching elements based on different constraints and testing particular situations

each land parcel. Defining spatial adjacency as the condition of two geometries g_1 and g_2 if g_1 and g_2 share at least one point, then a comparison of all points of the geometry of g_1 and g_2 can result to an implicit relation. The purpose of this implicit relation example is to make the reader aware that geometry keeps implicit spatial relations that are relevant for the enforcement of spatial access restrictions.

2.2.2 Geometry

Geometry provides a quantitative description of geographic characteristics like location, shape in different dimensions. The encoding of geometry depends on the used coordinate reference system (CRS) or Spatial Reference System (SRS). A coordinate reference system models the shape of the Earth, using a particular mathematical model. The geometry changes, when the geospatial information is transformed from one coordinate reference system to another. Different coordinate reference systems are available, which represent the Earth surface with different accuracy for different areas.

Standardizing Coordinate Reference System Identifiers

“The European Petroleum Survey Group (EPSG) was formed in 1986. It comprises specialist surveyors, geodesists and cartographers from Oil Companies based in Europe and having international operations. Meetings are held twice yearly to discuss survey and positioning topics within those areas of oil industry business where cooperation is generally agreed to be mutually beneficial.” [EPSG 2004]. The geodesy working group of the European Petroleum Survey Group maintains and publishes a list of unambiguous CRSs. One important Coordinate Reference System is the World Geodetic Standard 1984 (WGS84), which is widely used with the Global Positioning System (GPS). The EPSG identification string for WGS84 is *EPSG:4326*.

In a 2D geometry model, a location is encoded as a two-valued coordinate tuple XY . The semantics of X and Y depend on the used CRS. For *EPSG:4326*, the coordinate is encoded as latitude and longitude. Latitude (LAT) is measured northward and southward of the Equator. Longitude (LON) is measured eastward and westward the prime meridian, which goes through the Royal Observatory, in Greenwich.

In Germany, locations are encoded using the Gauss-Krüger projection, which divides the globe into 120 zones. Each zone is 3° wide (1.5° eastward and 1.5° westward) and has its own (virtual) prime meridian. The prime meridian of Zone 1 is 3° east of Greenwich. The EPSG codes for the zones are *EPSG:31491* for Zone 1, *EPSG:31492* for Zone 2, etc. Germany is divided into five zones, 2 to 5. The coordinate tuple elements have the following meaning: X has the semantics of distance from the equator (Ger. ‘Hochwert’) and Y from the prime meridian of each zone (Ger. ‘Rechtswert’). Both distances are typically measured in meter. In order to avoid negative Y -distances, 500,000 meters is added. For encoding the zone, its number is prefixed to the actual Y value. As an example, $Y=4530000$ is a point 30km west of the 12° meridian.

Figures 2.2 and 2.3 show the different grids for Germany: Figure 2.2 displays the latitude and longitude grid for WGS84 and the figure 2.3 displays the grid for the Gauss-Krüger projection.

An example for the different encodings is based on the location of the main entrance of the Technische Universität München, Arcisstr. 21, Munich, Germany. The location is $LAT = N40^\circ, 08'56.4''$, $LON = E11^\circ 34' 11.6''$. The equivalent WGS84 encoding is $X = 40.1490^\circ$, $Y = 11.5699^\circ$. Because Munich resides in zone 4 (prime meridian 12° east of Greenwich), the Gauss-Krüger encoding is $X = 5318468.41$, $Y = 4425482.06$.

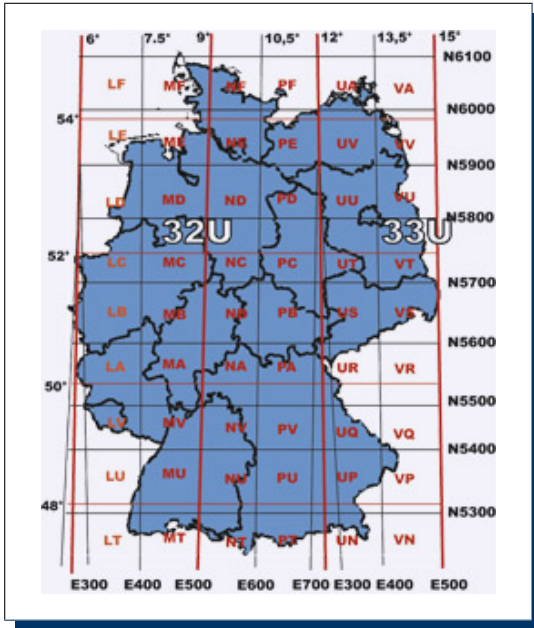


Figure 2.2: The WGS84 coordinate grid for Germany

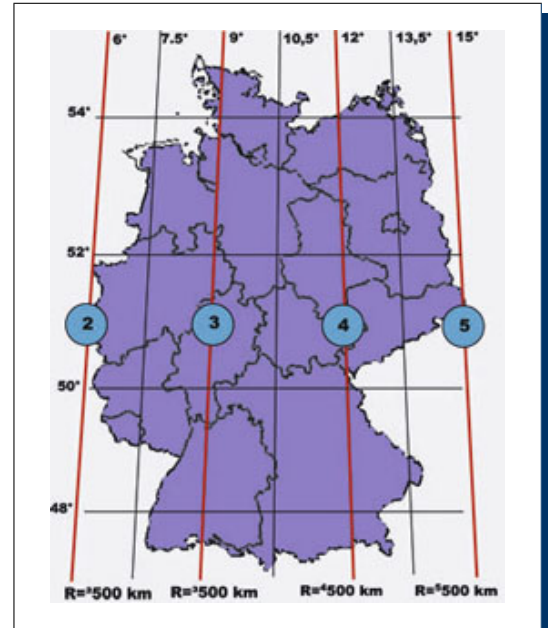


Figure 2.3: The Gauss-Krüger coordinate grid for Germany

Simple Geometries

This work uses a 2D geometry model according to the OGC specification Feature Geometry ([OGC 2001-101]). The following subset of simple geometries is relevant for this work:

Point: A 2D Point has the dimension zero (0). It can be encoded as a tuple of Coordinate Reference System (CRS) and the Coordinate, which represents the location. The numerical values of X and Y have the meaning as previously stated.

$$Point = \{CRS, Coord\} \quad (2.1)$$

$$Coord = X \ Y \quad (2.2)$$

The previous example of the location of the Technische Universität München, can be encoded using the **Point** construct:

$$TUMEntrance = \{EPSG:4326, 40.1490 \ 11.5699\}$$

Line: A 2D Line has the dimension one (1). It is encoded as a three-valued tuple: CRS and two coordinates using the CRS. The first coordinate represents the start location and the second coordinate represents the end location of the line. If $Coord_1$ and $Coord_2$ are identical, the dimension reduces to zero.

$$Line = \{CRS, Coord_1, Coord_2\} \quad (2.3)$$

LineString: A 2D LineString has the dimension one (1). It is encoded as a multi-valued tuple of at least two or more coordinates, encoded in the same CRS. A LineString can

be self crossing but never be closed. This is reflected in the constraint that Coord_1 must be different from Coord_N . A closed `LineString` becomes the semantics of a `LinearRing`.

$$\text{LineString} = \{CRS, \text{Coord}_1, \text{Coord}_i^*, \text{Coord}_N\} \mid \text{Coord}_1 \neq \text{Coord}_N \quad (2.4)$$

LinearRing: A 2D `LinearRing` has the dimension one (1). It describes the boundary of a 2D surface. It is encoded as a closed `LineString`: The first and last `Coord`-tuples are identical. A `LinearRing` must have at least three different coordinates.

$$\text{LinearRing} = \{CRS, \text{Coord}_1, \text{Coord}_2, \text{Coord}_i^+, \text{Coord}_1\} \quad (2.5)$$

Surface: A 2D `Surface` has the dimension two (2). It describes a 2D area, which outer boundary is encoded by a `LinearRing`. The surface can be simple or complex. If the `Surface` does not contain holes, it is called a simple surface. The holes of a complex surface have boundaries, which are called inner boundaries. Each inner boundary is encoded as a `LinearRing`. In order to describe a surface, each linear ring must have at least three different coordinates. Because the start and end coordinate are explicitly encoded, the `LinearRing` must have at least four coordinates. The first linear ring encodes the mandatory outer boundary and the other linear rings represent the inner boundaries of the holes.

$$\text{Surface} = \{\text{LinearRing}_1, \text{LinearRing}_i^*\} \quad (2.6)$$

The following examples show the encodings of a simple and a complex surface from figure 2.4. The encoding uses the pseudo CRS *foo*, which represents the Cartesian Coordinate System.

$$\begin{aligned} \text{SimpleSurface} &= \{foo, 0\ 0, 3\ 0, 3\ 4, 0\ 4, 0\ 0\} \\ \text{ComplexSurface} &= \{\{foo, 0\ 0, 3\ 0, 3\ 4, 0\ 4, 0\ 0\}, \{foo, 1\ 1, 2\ 1, 2\ 3, 1\ 3, 1\ 1\}\} \end{aligned}$$

Figure 2.4 shows examples of simple geometries.

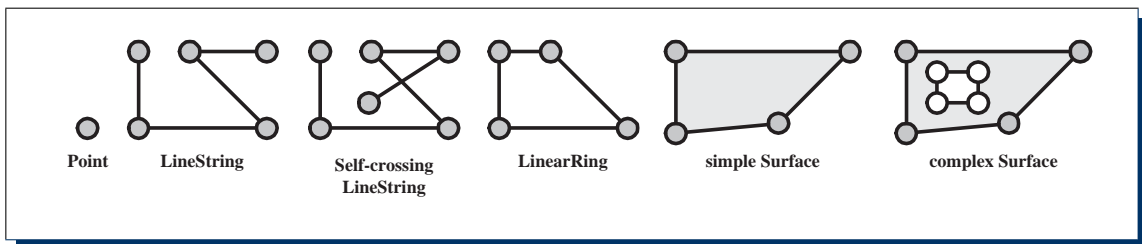


Figure 2.4: Examples of the simple geometries: Point, LineString, LinearRing and Surface

2.2.3 Testing Topological Relations between Geometries

Topology describes the characteristics of a geometry and the connectivity between geometries that stay invariant if the coordinate reference system changes. A common representation of

a topological structure is a graph, where the nodes represent the geometries and the edges represent the topological relation.

A *Spatial Relation Operation* is a Boolean function that tests two geometries for the existence of a particular topological spatial relation. The concepts of *Interior*, *Boundary* and *Exterior* ([Clementini 1993]) can be applied to 2-dimensional objects in 2-dimensional space (\mathcal{R}^2). The boundary of a given geometry g , expressed as $B(g)$ is a set of geometries of the next lower dimension. The boundary of a *Point* is the empty set (\emptyset). The boundary of a non-closed *Line* or *LineString* consists of two *Points*; the start and end location. The boundary of a closed *LineString* or a *LinearRing* is empty. The boundary of a *Surface* consists of its set of *LinearRings*. The interior of a given geometry g , expressed by $I(g)$ are the left over points when the boundary points are removed. The exterior of a given geometry g , expressed by $E(g)$ consists of those points, which are left over when the interior and boundary points are removed.

Different spatial relations, based on *The Dimensionally Extended Nine-Intersection Model* [OGC 1999-049] are defined. Seven of these methods are considered for testing spatial relations between two geometries in this work.

Let g be a geometry, then $\dim(g)$ returns the dimension of the geometry.

$$\dim(g) \in \{0, 1, 2\}, g \text{ is a geometry} \quad (2.7)$$

Disjoint: The two geometries have no *Point* in common. Given two (topologically closed) geometries $g1$ and $g2$,

$$\text{Disjoint}(g1, g2) \Leftrightarrow g1 \cap g2 = \emptyset \quad (2.8)$$

Touches: The two geometries have at least one point in common, but no interior points. Given two geometries $g1$ and $g2$, the *Touches* relation is defined for the following combinations of geometry dimensions: $\{\dim(g1)=2, \dim(g2)=2\}$, $\{\dim(g1)=1, \dim(g2)=1\}$, $\{\dim(g1)=1, \dim(g2)=2\}$, $\{\dim(g1)=0, \dim(g2)=2\}$, $\{\dim(g1)=0, \dim(g2)=1\}$. The *Touches* relation is not defined for the geometry combination $\{\dim(g1)=0, \dim(g2)=0\}$.

$$\text{Touches}(g1, g2) \Leftrightarrow (I(g1) \cap (g2) = \emptyset) \wedge (g1 \cap g2) \neq \emptyset \quad (2.9)$$

Crosses: The geometries have some but not all interior points in common. The method can be applied if the dimension of the intersection of the two geometries is less than that of at least one of the geometries. Given two geometries $g1$ and $g2$, the *Crosses* relation is defined for the following combinations of geometry dimensions: $\{\dim(g1)=0, \dim(g2)=1\}$, $\{\dim(g1)=0, \dim(g2)=2\}$, $\{\dim(g1)=1, \dim(g2)=1\}$, $\{\dim(g1)=1, \dim(g2)=2\}$.

$$\begin{aligned} \text{Crosses}(g1, g2) \Leftrightarrow & (\dim(I(g1) \cap I(g2)) < \max(\dim(I(g1)), \dim(I(g2)))) \\ & \wedge (g1 \cap g2 \neq g1) \wedge (g1 \cap g2 \neq g2) \end{aligned} \quad (2.10)$$

Within: Given two geometries $g1$ and $g2$, the *Within* relation is true if $g1$ lies in the interior of $g2$.

$$\text{Within}(g1, g2) \Leftrightarrow (g1 \cap g2 = g1) \wedge (I(g1) \cap I(g2) \neq \emptyset) \quad (2.11)$$

Overlaps: The geometries have some but not all points in common. The intersection of the two geometries have the same dimension as each geometry. Given two geometries $g1$ and $g2$, the **Overlaps** relation is defined for the following combinations of geometry dimensions: $\{\dim(g1)=2, \dim(g2)=2\}$, $\{\dim(g1)=1, \dim(g2)=1\}$, $\{\dim(g1)=0, \dim(g2)=0\}$.

$$\begin{aligned} \text{Overlaps}(g1, g2) \Leftrightarrow & (\dim(I(g1)) = \dim(I(g2)) = \dim(I(g1) \cap I(g2))) \wedge \\ & (g1 \cap g2 \neq g1) \wedge (g1 \cap g2 \neq g2) \end{aligned} \quad (2.12)$$

Intersects: The inverse of **Disjoint**³. Given two geometries $g1$ and $g2$,

$$\text{Intersects}(g1, g2) \Leftrightarrow \neg g1.\text{Disjoint}(g2) \quad (2.13)$$

Equals: The geometries are topological identical. Given two geometries $g1$ and $g2$,

$$\text{Equals}(g1, g2) \Leftrightarrow g1 \equiv g2 \quad (2.14)$$

Different geospatial relations between geometries are illustrated in section 4.2.3.

2.3 The Geography Markup Language (GML)

“The Open GIS Consortium, Inc. (OGC) is a member-driven, non-profit international trade association that is leading the development of geoprocessing inter operability computing standards. OGC works with government, private industry, and academia to create open and extensible software application programming interfaces for geographic information systems (GIS) and other mainstream technologies.” [OGC 2004]. “OGC envisions the full integration of geospatial data and geoprocessing resources into mainstream computing and the widespread use of interoperable geoprocessing software and geodata products throughout the information infrastructure.” [OGC 1999-100r1, p. 1].

The Geography Markup Language (GML) is an international standard of the OGC. “The Geography Markup Language (GML) is an XML encoding for the transport and storage of geographic information, including both the spatial and non-spatial properties of geographic features.” [OGC 2002-069, p. 2]. This work uses the GML version 2.1.2⁴, which assumes a linear interpolation between the coordinates of simple geometries. Even though the GML version 3.0 has been released, it is not considered relevant here. This is because the support of geometry types like spline and arc, which support a non linear interpolation between coordinates do not influence the developed access control model. It only influences the implementation of the system.

2.3.1 Encoding of Geospatial Information Objects and Geometry

GML provides an XML encoding for the object-oriented simple feature model ([OGC 1999-049]), shown in figure 2.5. In this respect, GML provides XML markup for the previous defined geographic information object. A feature is a GML encoding of an information object; a real

³In contrast to **Overlaps**, it is more general because no dimension restrictions exist.

⁴Please note that the GML version 2.1.2 is implicitly meant when using the term GML.

world phenomenon with no location to the surface of the Earth. A spatial feature is the GML encoding of a geographic information object that allows the presentation of spatial properties. In order to ensure interoperability, GML provides a definition for the XML markup of features, spatial features and their geometry. For the interoperable exchange of geodata, GML defines the construct of a feature collection.

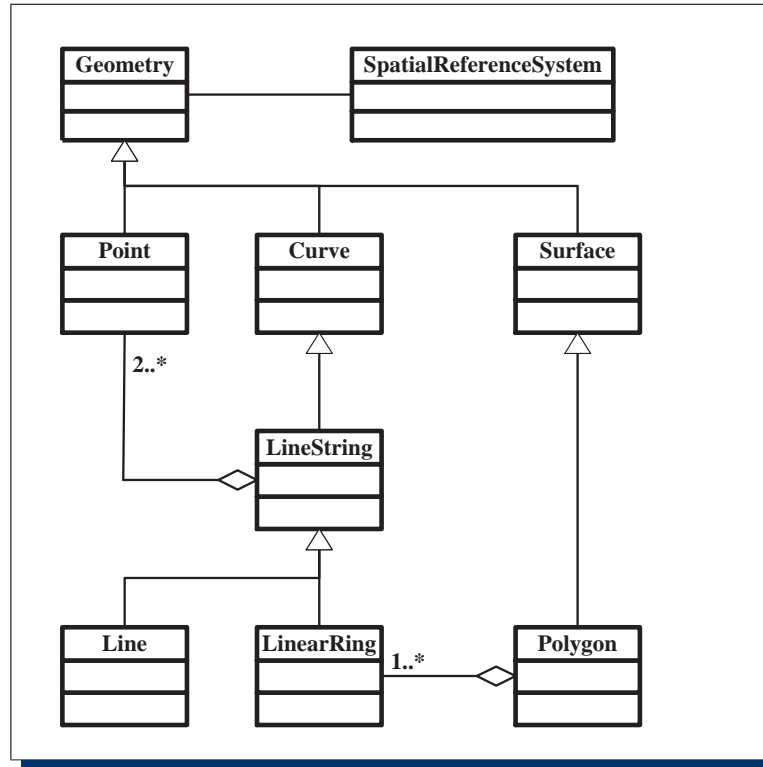


Figure 2.5: Simplified geometry model for GML 2.1.2 from [OGC 2002-069, p. 6], not showing the classes for geometry collection and multi geometry

GML 2.1.2 uses the W3C standard XML Schema ([W3C 2001d] and [W3C 2001e]) for the XML encoding. The XML namespace for the encoding is `gml`, which refers to the URL <http://www.opengis.net/gml>.

GML Feature and Feature Collection

The GML feature schema⁵ defines an XML markup for the GML object model. A geographic feature is essentially a named list of properties. Some or all of these properties may be geospatial, describing the position and shape of the feature. Each feature has a type, which is equivalent to a class in object modeling terminology, such that the class-definition prescribes the named properties that a particular feature of that type is required to have.

The characteristics of a feature are encoded using the $\{name, type, value\}$ triplet. For a GML encoding, these triplets are marked-up in XML. In the GML feature schema, the

⁵The feature schema is included in the file “feature.xsd”, which is available for download from the OGC Web site: <http://schemas.opengis.net/gml/2.1.2/feature.xsd>.

element `gml:_Feature` defines the basic structuring and XML markup for a feature.

For interoperable exchange of geodata, “GML 2.1 provides support for building feature collections. An element in an application schema that plays the role of a feature collection must derive from `gml:AbstractFeatureCollectionType` and declare that it can substitute for the (abstract) `gml:_FeatureCollection` element. A feature collection can use the `featureMember` property to show containment of other features and/or feature collections.” [OGC 2002-069].

Each abstract element `gml:_Feature` and `gml:_FeatureCollection` has the optional attribute `fid`, which holds the unique identification string (feature ID) for the feature, resp. the feature collection. Figure 2.6 shows the XML markup for these elements.

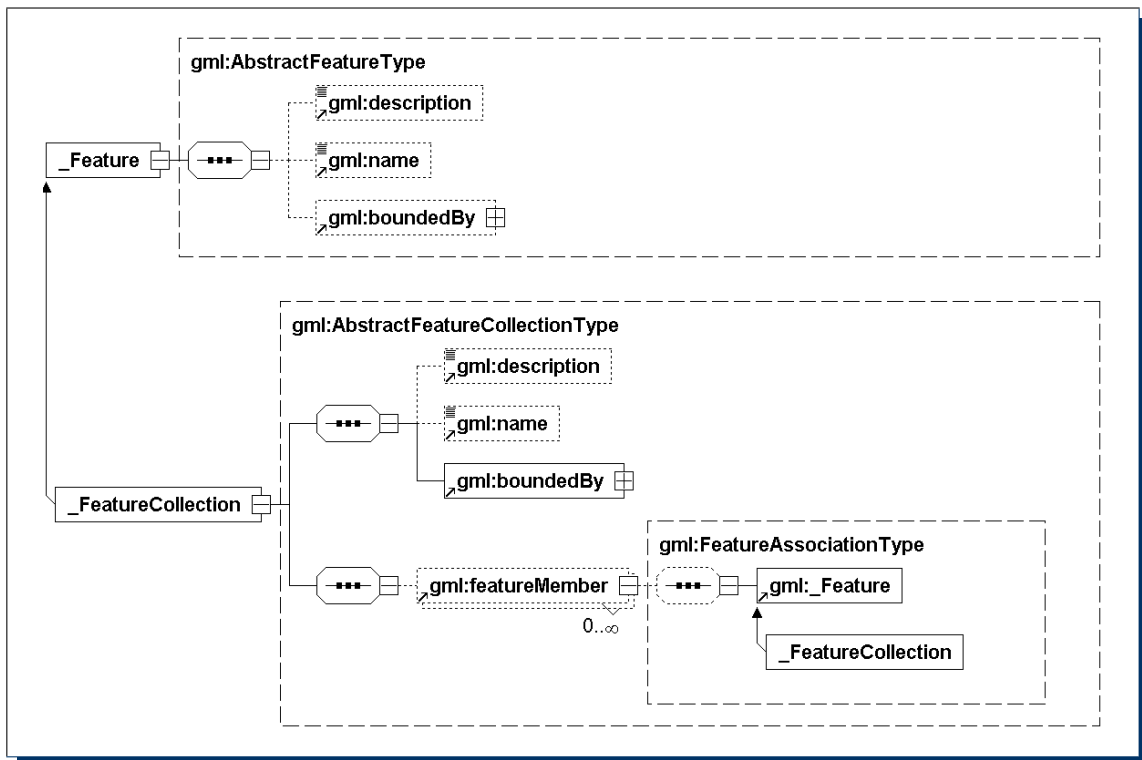


Figure 2.6: GML 2.1.2 feature schema defines the XML markup for a feature and a feature collection

GML Geometry

GML also provides XML markup for geometry, which is defined in the geometry schema⁶. The abstract base class of the geometry schema is `gml:_Geometry`. It has the optional attributes `gid` and `srsName`, which are inherited by all geometry classes. The attribute `gid` holds the unique identifier for the geometry and the attribute `srsName` holds the identifier of the used CRS. GML 2.1.2 defines a set of basic geometry classes, from which the following are used for this work: `Point`, `LineString`, `LinearRing` and `Polygon`.

⁶The geometry schema is included in the file “geometry.xsd”, which is available for download from the OGC Web site: <http://schemas.opengis.net/gml/2.1.2/geometry.xsd>.

Point The GML encoding of the simple geometry `gml:Point` is shown in listing 2.3. The location of the point can either be encoded using the element `gml:coord` or `gml:coordinates`. The `coord` element keeps the pair (X,Y,Z), where Y and Z are optional. Alternatively, the `coordinates` element can be used, whose value is a formatted string that keeps the values for X, Y and Z. The format of the string is fixed by the attributes `decimal`, `cs` and `ts`. The attribute `decimal` defines the character for the decimal separator, `cs` is the coordinate separator and `ts` is the tuple separator. The default coordinate separator is the comma (","), the default decimal separator is the dot (".") and the default tuple separator is the space character (" ").

```

1 <element name="Point" type="gml:PointType" substitutionGroup="gml:_Geometry"/>
2 <complexType name="PointType">
3   <annotation><documentation>
4     A Point is defined by a single coordinate tuple.</documentation>
5   </annotation><complexContent>
6     <extension base="gml:AbstractGeometryType">
7       <sequence><choice>
8         <element ref="gml:coord"/>
9         <element ref="gml:coordinates"/>
10      </choice></sequence>
11    </extension>
12  </complexContent>
13 </complexType>

```

Listing 2.3: GML encoding of the simple geometry `Point`

gml:LineString The GML encoding of the simple geometry `gml:LineString` shows listing 2.4. The sequence of two or more `coord` elements hold the points of the line string. The simple geometry `Line` can be encoded by a `LineString` with two `coord` elements.

```

1 <element name="LineString" type="gml:LineStringType" substitutionGroup="gml:_Geometry"/>
2 <complexType name="LineStringType">
3   <annotation><documentation>
4     A LineString is defined by two or more coordinate tuples, with linear interpolation between them.
5   </documentation></annotation>
6   <complexContent>
7     <extension base="gml:AbstractGeometryType">
8       <sequence><choice>
9         <element ref="gml:coord" minOccurs="2" maxOccurs="unbounded"/>
10        <element ref="gml:coordinates"/>
11      </choice></sequence>
12    </extension></complexContent>
13 </complexType>

```

Listing 2.4: GML encoding of the simple geometry `LineString`

gml:LinearRing The GML encoding of the simple geometry `gml:LinearRing` shows listing 2.5. The sequence of four or more `coord` elements hold the points of the line string. It is essential that the linear ring has a closed topology. Therefore, the first and the last coordinate must be identical. But this is not enforced by GML.

gml:Polygon In GML, there is no direct encoding of the geometry `Surface`. Instead, the derived class `Polygon` is used (see listing 2.6). The element `gml:outerBoundaryIs` holds

```

1 <element name="LinearRing" type="gml:LinearRingType" substitutionGroup="gml:_Geometry"/>
2 <complexType name="LinearRingType">
3   <annotation><documentation>
4     A LinearRing is defined by four or more coordinate tuples, with linear interpolation between them;
5     the first and last coordinates must be coincident.
6   </documentation></annotation>
7   <complexContent>
8     <extension base="gml:AbstractGeometryType">
9       <sequence><choice>
10        <element ref="gml:coord" minOccurs="4" maxOccurs="unbounded"/>
11        <element ref="gml:coordinates"/>
12      </choice></sequence>
13    </extension>
14  </complexContent>
15 </complexType>

```

Listing 2.5: GML encoding of the simple geometry `LinearRing`

a `gml:LinearRing` geometry that describes the outer boundary of the `gml:Polygon`. The boundaries of the optional holes are encoded as a sequence of `gml:innerBoundaryIs` elements. Each `gml:innerBoundaryIs` element holds the geometry, which is encoded as a `gml:LinearRing`.

```

1 <element name="Polygon" type="gml:PolygonType" substitutionGroup="gml:_Geometry"/>
2 <complexType name="PolygonType">
3   <annotation><documentation>
4     A Polygon is defined by an outer boundary and zero or more inner boundaries
5     which are in turn defined by LinearRings.
6   </documentation></annotation>
7   <complexContent>
8     <extension base="gml:AbstractGeometryType">
9       <sequence>
10        <element name="outerBoundaryIs">
11          <complexType>
12            <sequence><element ref="gml:LinearRing"/></sequence>
13          </complexType>
14        </element>
15        <element name="innerBoundaryIs" minOccurs="0" maxOccurs="unbounded">
16          <complexType>
17            <sequence><element ref="gml:LinearRing"/></sequence>
18          </complexType>
19        </element>
20      </sequence>
21    </extension>
22  </complexContent>
23 </complexType>

```

Listing 2.6: GML encoding of the simple geometry `Polygon`

2.3.2 Encoding of an Application Specific Data Model

The GML defines a common XML encoding for features and geometries to ensure syntactical interoperability. The defined markup has also fixed semantics to ensure semantical interoperability. These constructs must be used as defined and may not be changed. In order to markup application specific data structures and data models, these constructs from the

`feature.xsd` and `geometry.xsd` files must be extended by defining an application specific schema; the GML application schema. It includes the GML basic definitions and defines the specific markup of the application's object-oriented data model. It is important to notice that the application specific definitions may not use the GML namespace.

Because the exchange of GML marked-up geodata is based on feature collections, the application schema's root element must be declared as a substitutable to the element `gml:FeatureCollection` and its type extends the GML type `gml:AbstractFeatureCollectionType`. The root element can hold features through the `gml:featureMember` element, which is marked-up as a sequence. Listing 2.7 shows a simple example of a feature collection that can contain instances of the feature `LandParcel`.

```

1 <xs:element name="LandparcelCollection" type="am:LPCTYPE"
2   substitutionGroup="gml:FeatureCollection"/>
3 <xs:complexType name="LPCTYPE">
4   <xs:complexContent>
5     <xs:extension base="gml:AbstractFeatureCollectionType"/>
6   </xs:complexContent>
7 </xs:complexType>
8 <!-- Defintion of feautres -->
9 <xs:element name="LandParcel" type="am:LPTYPE"
10  substitutionGroup="gml:Feature"/>
11 <xs:complexType name="LPTYPE">
12   <xs:complexContent>
13     <xs:extension base="gml:AbstractFeatureType">
14       <xs:sequence minOccurs="0">
15         <xs:element name="Identifier"/>
16         <xs:element name="shape" type="gml:LinearRingType"/>
17       </xs:sequence>
18     </xs:extension>
19   </xs:complexContent>
20 </xs:complexType>

```

Listing 2.7: Simple GML application schema that defines a feature collection that can hold one feature type, named `LandParcel`

The City Model Example

The City Model represents a geodata content to be used later for illustrating theoretical concepts and problems in the fields of declaration and enforcement of access restrictions. In this section, it's used to illustrate the development of a GML application schema. Later, the illustration of declaration and enforcement of access restrictions is based on that application schema. The development uses the object-oriented data model from figure 2.7.

The data model comprises of the classes `Building`, `Intersection` and `Street`. Each class has properties, which hold the characteristics of the object. The geometry of the `Building` class is described by the simple geometry class `Surface` and the `Street` geometry is described by the class `LineString`. The data content (the objects) of this model is displayed in table 2.4. It comprises of two instances of the class `Building`, six instances of the class `Street` and five instances of the class `Intersection`. All geospatial information objects are included in the model to illustrate spatial restrictions on 0D, 1D and 2D geometry. The building's shape is defined by a 2D geometry, the street's path is defined by a 1D geometry and finally the

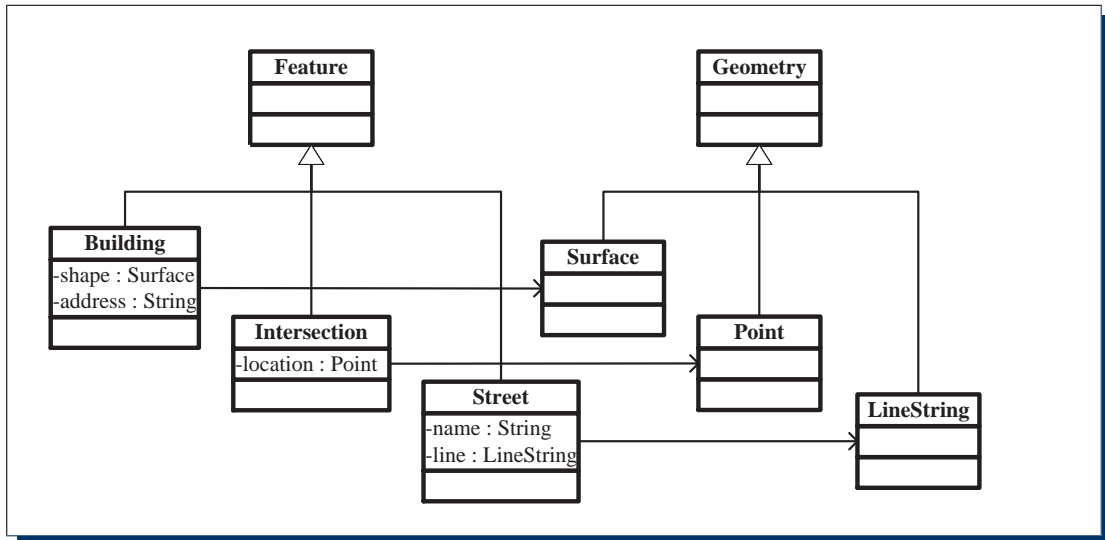


Figure 2.7: The City Model example data model

intersection's location is defined by a 0D geometry.

class	id	non-spatial properties	geometry
Building	HouseA	address=String(3 Street A)	shape=Surface(foo, -1 2,0 2,0 3,-1 3)
	HouseB	address=String(5 Street D)	shape=Surface(foo, 5 4,6 4,6 5,5 5)
Street	StreetA	name=String(Street A)	line=LineString(foo, 0 0,0 4)
	StreetB	name=String(Street B)	line=LineString(foo, 0 4,3 4)
	StreetC	name=String(Street C)	line=LineString(foo, 0 0,3 0)
	StreetD	name=String(Street D)	line=LineString(foo, 3 4,6 4,6 3)
	StreetE	name=String(Street E)	line=LineString(foo, 3 0,6 3)
	StreetF	name=String(Street F)	line=LineString(foo, 3 0,2 2,3 4)
Intersection	XA	name=String(X A)	location=Point(foo, 0 0)
	XB	name=String(X B)	location=Point(foo, 0 4)
	XC	name=String(X C)	location=Point(foo, 3 0)
	XD	name=String(X D)	location=Point(foo, 3 4)
	XE	name=String(X E)	location=Point(foo, 6 3)

Table 2.4: Geospatial information objects of the City Model

The development of an application schema requires the definition of a namespace. For the GML encoding of the City Model, the namespace `am` is used. The GML encoding further requires to define a root element for the feature collection. This element must be a substitutable for the `gml:FeatureCollection`. The type definition must extend the `gml:AbstractFeatureCollectionType`. Each feature of the model is named according to the class name and substitutable to `gml:Feature`. The geometries of the features must be encoded using GML simple geometry definitions.

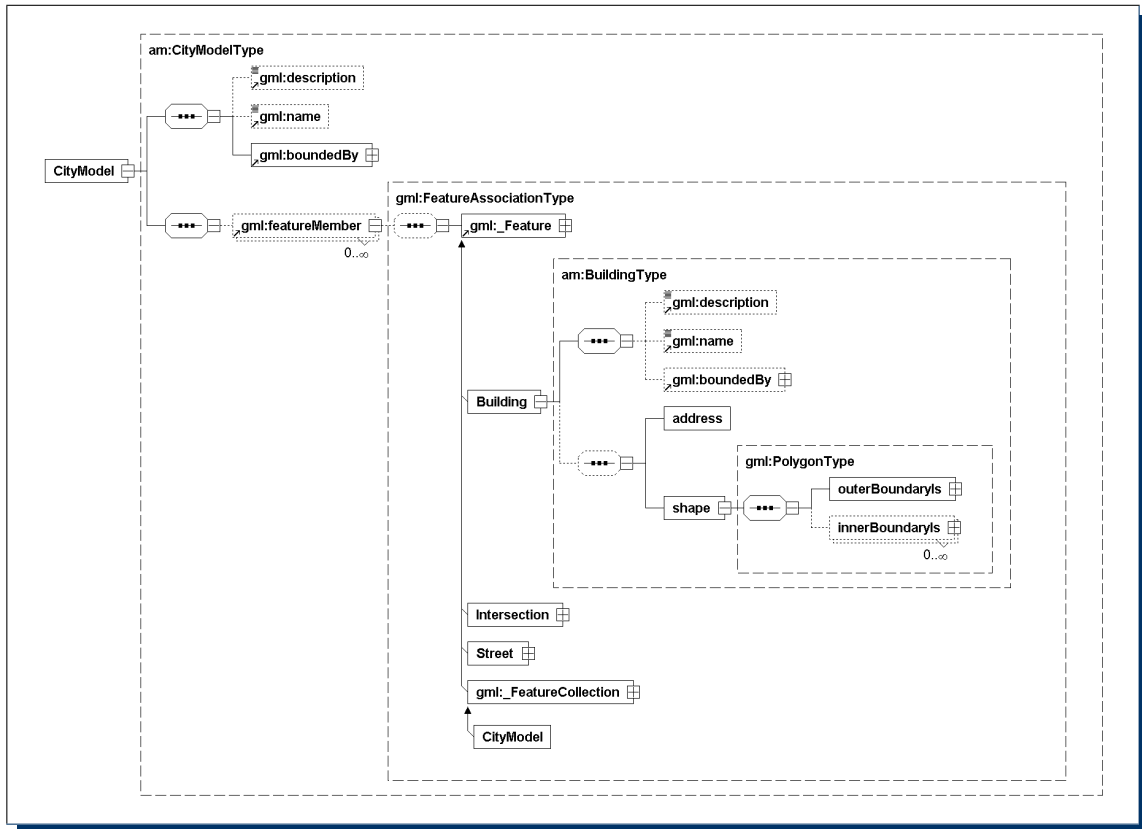


Figure 2.8: GML application schema for the city model example

The City Model application schema⁷ defines the tag `CityModel` as the top level element. This element is a substitute for the `gml:FeatureCollection` and therefore defines a GML feature collection. According to GML, it has an optional `fid` attribute, the mandatory element `gml:boundedBy` and a sequence of `gml:featureMember` elements. The element `gml:boundedBy` is a rectangular box, which represents the convex hull of the geometries of the features, contained in the feature collection. Each `gml:featureMember` element holds one element, which represents a GML feature. The elements `Building`, `Street` and `Intersection` are GML features. Please note that this is just one possible way to encode the City Model in GML. A part of the city model application schema is shown in figure 2.8.

According to that City Model application schema, the content can be marked-up in GML. A part⁸ of a valid GML feature collection encoding of the geospatial information objects from the City Model is shown in listing 2.8.

2.3.3 GML and Interoperability

It is wrong to say that GML is interoperable. But it is true to say that GML provides the means and XML markup which can be used to encode geodata in an interoperable way. Thus, interoperability can only be achieved if all involved parties agree on used GML application

⁷The City Model application schema is printed in appendix B.2.

⁸A complete feature collection for the City Model is provided in section B.3.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <CityModel xmlns="http://www.in.tum.de/am" xmlns:am="http://www.in.tum.de/am"
3   xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://http://www.in.tum.de/am CityModel.xsd" fid="CityModel" >
6   <gml:boundedBy><gml:Box gid="box1" srsName="foo">
7     <gml:coord><gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
8     <gml:coord><gml:X>6</gml:X><gml:Y>5</gml:Y></gml:coord>
9   </gml:Box></gml:boundedBy>
10  ...
11  <gml:featureMember>
12    <Building fid="HouseA" >
13      <address>3 Street A</address>
14      <shape srsName="foo">
15        <gml:outerBoundaryIs>
16          <gml:LinearRing>
17            <gml:coord><gml:X>-1</gml:X><gml:Y>2</gml:Y></gml:coord>
18            <gml:coord><gml:X>0</gml:X><gml:Y>2</gml:Y></gml:coord>
19            <gml:coord><gml:X>0</gml:X><gml:Y>3</gml:Y></gml:coord>
20            <gml:coord><gml:X>-1</gml:X><gml:Y>3</gml:Y></gml:coord>
21            <gml:coord><gml:X>-1</gml:X><gml:Y>2</gml:Y></gml:coord>
22          </gml:LinearRing>
23        </gml:outerBoundaryIs>
24      </shape>
25    </Building>
26  </gml:featureMember>
27  ...
28 </CityModel>

```

Listing 2.8: GML feature collection snippet of the City Model

schemas. In that sense, it is essential that all parties agree on the used XML markup and the meaning of the elements. In order to keep the GML predefined semantics for elements and type definitions, it is not permitted to use the gml namespace as the target namespace of the application schema and it is also not permitted to change GML pre-defined elements. For a common understanding, the use of GML pre-defined elements shall only be used in the sense, expressed by the OGC.

2.4 Interoperable Use of Distributed and Heterogenous Geodata

One of the first approaches of interoperable access to distributed and heterogenous geodata was started by the OGC in 1993. *“The Open GeoData Interoperability Specification (OGIS) ... is an attempt to design methods that provide an object-oriented architectural framework for access to geodata, independent of the specific data structures and file formats used to model the data. From the user’s point of view, OGIS allows access to geodata at remote locations, no matter the format. From an application developer’s point of view, OGIS provides a set of network services to identify, interpret, and represent dataset from a geodata server to a geoprocessing client.”* [Kenn Gardeks]. This approach results in different OGC specifications, which describe required aspects in detail.

2.4.1 The Open GIS Consortium and their Interoperability Specifications

The OGC has released abstract and implementation specifications. The purpose of the abstract specifications is to provide a general, but sufficient model for different issues of interoperable geoprocessing. Among other topics, the following OGC abstract and implementation specifications are relevant for this work⁹:

Topic 5 – Features [OGC 1999-105r2]: A feature is a software representation of a real world phenomenon, which may have a location relative to the Earth. Each feature has an identity which is unique in the domain it belongs to. The encoding of the feature's location and extend is defined in topic 1.

Topic 10 – Feature Collections [OGC 1999-110]: A feature collection is a set of features instances, encoded according to the associated feature and projection schema. The feature schema describes the structure of the features and the projection schema describes the structure of the feature collection.

Topic 1 – Feature Geometry [OGC 2001-101]: This international standard provides conceptual schemas for describing and manipulating the geospatial characteristics of features. The standard specification defines the encoding of feature locations and extends, which is relevant in the context of defining and enforcing spatial access restrictions.

Topic 12 – The OpenGIS Service Architecture [OGC 2002-112]: This international standard describes the taxonomy of services for processing geospatial information. It also describes three basic models for service chaining: transparent, translucent and opaque chaining.

Web Map Service (WMS) [OGC 2001-068r3]: The WMS is a Geo Web Service that produces maps of geospatial data. The content of the map comprises of a set of geographic features. Each feature is represented by a graphical symbol, whose position on the map is related to the location of the feature. Different input parameters control the result of the service: the map.

Web Feature Service (WFS) [OGC 2002-058]: The WFS is a Geo Web Service that provides operations which allow read, change, create and delete features from an associated feature store, typically a database. The service result is a feature collection that is primarily serialized using GML. Different input parameters control the behavior of the service in order to exchange geospatial information objects.

Geographic Markup Language (GML) [OGC 2002-069]: *“The Geography Markup Language (GML) is an XML encoding for the transport and storage of geographic information, including both the spatial and non-spatial properties of geographic features.”* [OGC 2002-069, p. 2]. A valid GML document shall be compliant to an application schema that defines the structure of features and the structure of the GML document.

Simple Features - SQL [OGC 1999-049]: The standard defines a geographic model for simple feature geometry and operators for the different geographic types. This model and the defined operators and their semantics becomes relevant when enforcing spatial access restrictions.

⁹The interested reader may find all original specifications on the OGC Web site at www.opengis.org.

2.4.2 Definition of Interoperability

Interoperable is an adjective, which definition is ‘‘able to operate in conjunction’’ (Concise Oxford Dictionary, 9th Edition). Another definition of the term interoperability is ‘‘...the ability of a system or a product to work with other systems or products without special effort on the part of the customer.’’ (www.whatis.com). As described in [Miller 2000], interoperability has many flavors: Technical, semantic, political/human, inter-community, international and legal interoperability.

The technical interoperability focuses on the ability to overcome the challenge to describe system interfaces and processing in such a way that they can work together in a seamless way. It also focuses on the requirements to describe the messages, sent between the systems in such a way that they can be processed without intervention of the customer or user.

The semantic interoperability focus on the challenge to give defined message structures, systems and their interfaces an unambiguous meaning. The semantic interoperability copes with the problem to share a common meaning, known as the rhetorical theory *triangle of meaning*, as introduced by C.K. Ogden and I.A. Richards [Richards 1979]. One approach to overcome the semantic challenge is the Semantic Web approach, where concepts and ontologies are the central building blocks.

The feature of a system or the exchanged data to be interoperable becomes more important today as it was in the past. This results in the globalization, where the borders between companies and government agencies are flattered. In the geographic field, many companies and government agencies have collected and maintained large amounts of geodata. Each repository does not have a great value, but the combination has. This drives a continuous process to become interoperable and to make information available to others so that all can benefit from it: The user benefits from the interoperability, because complex decision making becomes available and the data providers benefit from interoperability as it opens new market potential to get more paying users. In order to have interoperability for this to happen, superordinate organizations such as the W3C, OASIS and the OGC are important as they define how to ensure interoperability on different levels.

Interoperability in this work focuses on three different technical levels: data interoperability, service interoperability and access control interoperability. Different viewpoints on interoperability exist that reflect these different levels.

Geodata Interoperability

Data interoperability is essential for data exchange or processing between parties. It is based on a mutual agreement of all involved parties in respect of syntax and semantics, as stated in [OGC 2002-112, p. 5]. It is the informational viewpoint that focuses on the syntax and semantics of the exchanged data, required for processing.

The syntax defines the structure of the data and the semantics focus on the meaning of used terms. For the object-oriented approach, used terms refer to the name of class and their properties, as well as the type of the properties. The semantics refer to a common understanding what the classes, properties and types mean.

In this respect, GML defines the foundation for interoperability storage and exchange of

geodata. It provides means for an XML encoding of the object-oriented model for simple features and different types of geometry.

Geographic interoperability can be defined as an extension to this general data interoperability, because it also defines the requirements for the exchange and processing of geometry information. Because geometry relies on a particular CRS, it is important to use unique identifiers for the CRS. GML provides a placeholder (attribute `srsName`) for a unique identification of the used CRS. This attribute can keep any identifier string such as the EPSG-codes, which provide a unique identifier for Coordinate Reference Systems.

In order to guarantee interoperability over domains, it is essential that the used definitions can be referenced in a unique and unambiguous way. Using GML, the GML namespace guarantees that pre-defined constructs can be referenced in such respect.

Service Interoperability

Service interoperability is a very broad term. The IEEE defines it as *“the ability of two or more systems or components to exchange information and to use the information that has been exchanged”* [IEEE 1990]. A more detailed definition is that *“interoperability is the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units.”* [OGC 2002-112].

The OGC Service Architecture ([OGC 2002-112]) specification defines service interoperability based on the Reference Model of Open Distributed Processing (ISO/IEC 10746). That specification approaches service interoperability from different viewpoints.

The enterprise viewpoint focuses on the purpose and scope of the services. The engineering viewpoint is concerned with the environment for distribution of the services and the network infrastructure. The technology viewpoint describes the hardware and software issues for implementation. The implementation constraints take the cost and availability issues under consideration.

Important for this work is the communication viewpoint, which is concerned with the interaction pattern for a service, manifested by their interfaces. In this respect, a service is a functional unit that implements a particular set of interfaces. This is an important viewpoint for the enforcement process. It is essential to understand the parameters of a service and the service response in order to form an appropriate input for the authorization process.

Access Control Interoperability

Access control interoperability is important for enterprise wide declaration, modification and enforcement of access restrictions.

Declaration of access restrictions throughout the enterprise requires (i) a common language to encode the restrictions and (ii) enterprise wide identical procedures to enforce the declared restrictions. Even the enterprise wide management of the access restrictions is an important topic, it is not covered in this work.

Access control interoperability from the user’s point of view ensures that she/he can access

restricted resources without the awareness that an enforcement of declared access restrictions exist. This assumes that the user already has a unique identifier or has been authenticated by assertions, for which appropriate access rights exist.

Access control interoperability from the communication point of view ensures that different systems (clients or services) can communicate with each other, without specific knowledge of the internals of the access control.

2.4.3 The Web Map Service (WMS) Implementation Specification

This open standard [OGC 2001-068r3] describes a set of generic service interfaces. One interface returns the capabilities of the service, the second interface is capable of controlling transformation of geodata into map representations. The third interface is optional. It returns metadata for selected features on a requested map.

GetCapabilities: The **GetCapabilities** interface can be used to retrieve the capabilities of the service. This interface returns service metadata as a valid XML document, the capabilities document. It can be processed by a WMS compatible client or any other application in order to provide 'on-the-fly' configurable client functions. The content of the document can also be transformed into HTML for user readability.

GetMap: The **GetMap** interface supports the request of map representation of geodata. Different geospatial and dimensional parameters control this transformation.

SRS (Spatial Reference System) is a unique string representation of the Coordinate Reference System (CRS) for which the map shall be generated. Possible values are listed in the capabilities document. An example string for this parameter is *EPSG:4326*, which denotes the CRS WGS84.

BBOX describes the area of interest for which the map is to be generated. The format of the parameter is a comma separated string of the coordinate tuples for the left lower and upper right corner of the area of interest. The values fit to the used SRS string. An example for the BBOX parameter is the string '-97.105, 24.913, -78.794, 36.358'. This BBOX describes the area of interest for the map from figure 1.1. The coordinate of the left lower corner is ($W97.105^\circ, N24.913^\circ$) and of the upper right corner is ($W78.794^\circ, N36.358^\circ$). The maximum area, for which a WMS instance provides maps and metadata is described in the capabilities document.

WIDTH Gives the width of the requested map in pixels (x-axis for the Cartesian Coordinate System).

HEIGHT Gives the height of the requested map in pixels (y-axis for the Cartesian Coordinate System).

FORMAT is a string, representing the MIME encoding for the map format. For example, the Compuserve GIF format is referenced by the string *image/gif*. The supported formats are listed in the capabilities document'.

LAYERS refers to the service internal structuring of the WMS geodata. When thinking of an object-oriented approach, a layer is equivalent to a class or in GML terminology a spatial feature type. Instances of the class build the resources of the Web Map Service, which can only be fetched according to their geometry.

GetFeatureInfo: The `GetFeatureInfo` interface is optional. It can be used to retrieve particular metadata about a feature, presented on the map. Because the WMS is a stateless service, it does not know the map for which the user likes to retrieve additional information. Therefore, the use of the `GetFeatureInfo` interface requires to attach the parameters of the `GetMap` request plus the coordinate of the point of interest, using the Cartesian Coordinate System. The WMS calculates the absolute (real-world) location according to the CRS from that location. This interface uses three important parameters to control the fetching of the metadata.

QUERY_LAYERS A `QUERY_LAYER` is a layer of the `GetMap` interface for which metadata about features can be queried. Which layers can be queried is listed in the capabilities document.

X,Y These parameters define the location of the point of interest, based on a previously requested map. The values define the position in pixels, according to the Cartesian Coordinate System.

In Figure 5.3, page 170, the calculation of the feature's real-world location, based on the given parameters is illustrated.

Implications to Access Control

With respect to access control, it is essential to understand how the combination of `GetMap` parameters control the service output. Let's take the WMS `GetMap` request for the example map from listing 2.9:

```

1 http://a-map-co.com/mapserver.cgi?VERSION=1.1.0&REQUEST=GetMap&
2 SRS=EPSG:4326&BBOX=-97.105,24.913,-78.794,36.358&WIDTH=560&HEIGHT=350&
3 LAYERS=AVHRR-09-27&STYLES=&FORMAT=image/png&BGCOLOR=0xFFFFFFFF&
4 TRANSPARENT=TRUE&EXCEPTIONS=application/vnd.ogc.se.inimage
5
6 http://b-maps.com/map.cgi?VERSION=1.1.0&REQUEST=GetMap&
7 SRS=EPSG:4326&BBOX=-97.105,24.913,78.794,36.358&WIDTH=560&HEIGHT=350&
8 LAYERS=BUILTUPA_1M,COASTL_1M,POLBNDL_1M&STYLES=0xFF8080,0X101040,BLACK&
9 FORMAT=image/png&BGCOLOR=0xFFFFFFFF&TRANSPARENT=TRUE&
10 EXCEPTIONS=application/vnd.ogc.se.inimage

```

Listing 2.9: WMS `GetMap` requests for retrieving the map from figure 1.1

The map format is `image/png`, which denotes a binary image. Therefore, the authorization decision cannot be based on the response of the service. The `SRS` and `BBOX` parameters declare the area of interest. For this area, all geospatial information objects from the layer `AVHRR-09-27` are selected. The identities of the included objects are not known. The authorization decision can only be based on the area of interest and the layer information.

An authorization decision for the `GetFeatureInfo` interface can be based on similar information. The parameter `QUERY_LAYERS` denotes the classes and the `X` and `Y` parameters denote the point of interest on the displayed map.

The `GetCapabilities` interface is not subject for access control because it returns the capabilities of the service and not the personalized capabilities of the user. In order to have a user request his capabilities from the WMS, an additional interface (`GetUserCapabilities(user-ID)`) is recommended.

2.4.4 The Web Feature Service (WFS) Implementation Specification

This international standard [OGC 2002-058] is very complex. It supports an ‘SQL-like’ transformation of geodata into structured text. The actual version of this specification is 1.0.0. It supports the simple feature model of GML 2.1.2, as introduced in section 2.3.

In more detail, a Web Feature Service implementation defines the `GetFeature` interface to transform an object-oriented structure of (geospatial) information objects into a GML encoded feature collection, as introduced in section 2.3. The WFS also has the mandatory interfaces `GetCapabilities` and `DescribeFeatureType` and the optional interfaces `LockFeature` and `Transaction`. A service that just implements the mandatory interfaces is called a basic WFS. A service that implements all interfaces is called a transactional WFS.

The Basic Web Feature Service

GetCapabilities: The `GetCapabilities` interface has the same semantics as for the WMS. It returns an XML encoded capabilities document of the service metadata.

DescribeFeatureType: The `DescribeFeatureType` interface allows the request of a valid GML application schema that defines the XML markup for the requested feature type. The mandatory parameter for this interface is `TypeName`.

TypeName defines the feature type for which the schema definition is to be returned. If this parameter is missing or empty, the WFS returns the schema definition for all supported feature types.

GetFeature: The `GetFeature` interface returns a `gml:FeatureCollection`, which per default is a valid GML document. The markup of that feature collection is defined by a specific GML application schema. This application schema, which includes the GML schemas for simple features and geometry, has two duties: First, it defines the markup of the (geospatial) features by a XML Schema type declaration. Each feature has mandatory attributes and elements, inherited from the GML basic types and application specific attributes and elements. Second, it defines the structure (markup) of the feature collection. A valid GML feature collection is shown in listing 2.8.

Regarding the WFS response (a feature collection), the parameters of the `GetFeature` interface can be separated into two different categories. The first category of parameters defines, which feature types are included in the response and what properties of the features are fetched. This is controlled by the parameters `TypeName` and `PropertyName`.

TypeName is the parameter that represents a supported feature type. All supported feature types are listed in the capabilities document.

PropertyName is the parameter that controls what properties of a feature type (specified by `TypeName`) are to be included in the feature. If the `PropertyName` parameter is not being used or empty, all properties of the feature type are included.

The second category of parameters control which feature instances are included into the response, the feature collection. The three mutual exclusive parameters are `FeatureID`, `FILTER` and `BBOX`.

FeatureID is the parameter to query a particular feature, identified by its feature ID (attribute `fid`). This assumes that a WFS associates unique fids to all accessible features and always uses the same fid for the same feature. This is particularly important for the updating of features, using the `Transaction` interface. A fid remains unchanged for the lifetime of the feature.

FILTER is the parameter that holds an XML encoding of the OGC filter implementation specification [OGC 02-059]. The filter encoding ‘... defines an XML encoding for filter expressions based on the BNF definition of the OpenGIS Common Catalog Query Language as described in the OpenGIS Catalog Interface Implementation Specification, Version 1.0.’ [OGC 02-087r3]. It can be compared with the `WHERE` section of an SQL2 ([ISO 1992]) based query. In relation to the `FeatureID` parameter, a filter supports the selection of features, based on their characteristics, expressed as properties.

BBOX is the parameter that allows to define an area of interest for which features of a particular type (specified by `TypeName`) are fetched. Thus, this parameter fetches spatial features according to their geospatial characteristics. The geometry of the area of interest is represented by a rectangular bounding box. The format of the parameter is equivalent to the parameter `BBOX` of the WMS implementation specification.

The Optional (Transactional) Interfaces of the Web Feature Service

A transactional Web Feature Service implements the optional interfaces `LockFeature` and `Transaction`. In order to support sequential write operations, the service can no longer be stateless. Therefore, the use of the `LockFeature` interface can be used to lock feature instances of the same type for a specific time. After the time is expired, the WFS releases the lock. During the locking time, modification operations can be carried out on the locked features. The lock ID must be supplied with the operations in order to refer to the same lock. The `Filter` parameter can be used to select multiple features, according to their characteristics. The state diagram of the transactional WFS ([OGC 2002-058, figure 5, p. 3] shows the mechanism of locking.

During an active lock, the `Transaction` interface can be used to modify the data store of a WFS. The interface supports the `Delete`, `Update` and `Create` operations. The associated parameters depend on the used operation. For example, the `Delete` operation takes a `Filter` parameter that specifies the feature instances to be deleted. The parameter `releaseAction` determines what the WFS shall do after a successful completion of the invoked operation.

Implications to Access Control

With respect to an access control for distributed geospatial information objects, it is important to keep in mind that the WFS supports an object-oriented organization of the geodata. For this work, it is assumed that the identities of the accessible features are unique. Regarding the authorization decision, it is important to understand how the WFS fetches the features of the output, based on the input parameters.

Depending on the service operation and the parameters of a request, the authorization decision can only be based on the service response. Even this seems to be rather awkward,

it is inevitably correct processing. The features (geospatial information objects), for which declared access restrictions must be enforced is dynamically created by the service through the request parameter. However, certain service operation/parameter combinations allow or require to derive an authorization decision upon the request parameter already.

For example, the use of the service operation `Transaction` allows to invoke operations that modify the repository. Before such a request can be executed by the service, the authorization of the subject must be evaluated. More detailed investigations about deriving an authorization decision on the request parameters are deferred until chapter 5.

2.4.5 Conclusion on Interoperability and Implications to Access Control

Putting the building blocks together brings interoperability from the geodata store to the combined use in a client application. On the service level, implementation specifications like the WMS and the WFS allow geodata processing by generic and interoperable interfaces. If these interfaces are implemented by services, which can be invoked via Internet Technology, they are called Geo Web Services. On the data exchange level, interoperability is ensured by using a common format or markup of the geographic information. For map representations, different international -non OGC- formats are available. Different binary formats like GIF, JPEG, etc. and vector formats like SVG enable the combined use of maps. For the textual representation of geodata, GML provides the means of interoperability through a common XML markup, based on an object-oriented model for simple features and geometries.

The example use case from chapter 1 (figure 1.1) to issue a gale warning for the area of the Gulf of Mexico actually uses the combination of maps from two different data providers. The combined use is powered by two Web Map Service implementations, where the maps are requested and combined in the client.

2.5 Introduction to Access Control

Access control is a framework that allows to regulate the access to protected resources. This is based on the framework's capabilities to express the protection in access permissions. For a concrete request, the access control system evaluates the available permissions if the initiator of the request has adequate access rights. If so, the request is accepted and rejected otherwise. In short, an access control system regulates 'who may do what'.

2.5.1 Used Terminology

Before continuing, the used terminology is specified in order to help the reader to understand relevant aspects.

Authentication is the verification of the identity of a person or process, initiating a request. It is a prerequisite for access control, because it is required by the authorization process.

Authorization is the legislative of an access control system. It is the process that tries to confirm that a subject is allowed or disallowed to access a resource in the intended way.

Enforcement is the executive of an access control system. It is the process that intercepts the communication between the subject and the protected resources. It fetches required information from the request, essential to the authorization process. It requests an authorization decision from the authorization process. Based on that decision, the current request is accepted or rejected.

Subject is the activator or initiator of a request. This can be a user or a software agent, uniquely identified through the authentication process. One possible representation of the identity is to use assertions.

Assertions in this context are used to verify that the assumptions of a subject made about its identity are actually valid. Assertions can be represented by a set of attribute value pairs that uniquely identify a subject. The validity is guaranteed by the authentication.

Operation (or action) is the method or function that a subject intends to invoke on the resource. For geospatial information objects, different standard operations such as `read`, `write`, `delete`, `create` and `print` are meaningful. Due to the geospatial aspect, operations such as `map` or `transform` can be defined. The `map` operation allows to create a graphical representation of the geodata and `transform` recalculates the geometry from a given CRS to a target CRS.

Resource is the entity that is protected by the access control system. In the context of this work it is geographic information, accessible in an object-oriented model. In the environment of this work, a resource can only be accessed via a service.

Permission is a machine read- and interpretable expression for making the declaration protection to resources explicit. It can define an allowance or a disallowance for a particular subject to use a designated operation on an intended resource. For detailed usage, this work refers to an allowance as a positive permission (P^+) and to a disallowance as a negative access statement (P^-).

P^+ is a machine read- and interpretable expression that explicitly permits an identified subject the access to a designated resource(s) using a particular operation.

P^- is a machine read- and interpretable expression that explicitly denies an identified subject the access to a designated resource(s) using a particular operation.

2.5.2 Different Strategies for Managing Access Rights

Access rights are positive permissions that give subjects the ability to access certain resources with designated operations. The management of access rights can follow the access control or information flow strategy.

The information flow strategy, better known as the **Bell-La Padula** model, defines classifications to resources and privileges to subjects. In short, the management of access rights can be defined as ‘‘No read up - no write down’’. This strategy ensures that information, which has a certain classification level cannot be made less classified by writing. Also no subject can get read access to higher classified information as it corresponds to his privilege level.

Because this strategy is typically important for military use, it is not considered a relevant topic in this work.

The access control strategy for managing the assignment of access rights to users differentiates between the Mandatory Access Control (MAC) and Discretionary Access Control (DAC) strategy. The DAC is based on the owner principle. Each owner of a resource can decide, who else gets access and if that subject also gets the right to pass access rights on to other users as well. This strategy is very complex to handle and bears the problem that the owner of a resource loses control, resp. cannot verify, what subjects have actually access to owned resources. In contrast to DAC, the MAC strategy defines system-wide restrictions on resources. These restrictions are typically managed by a responsible person – the policy writer –, who has the duty to declare restrictions in a way that the access control system can enforce them.

This work does not cope with the strategies of managing access rights. However, it assumes that access permissions are encoded by a policy writer, who is responsible to fix existing flaws. Therefore, the results of testing existing permissions for inconsistencies can be presented to that policy writer.

2.5.3 The Basic Access Control System

The implementation of an access control system requires three components: authentication, authorization and enforcement. This work excludes the authentication component. It is assumed that each subject, issuing a request to protected resources is uniquely identified by assertions, validated and guaranteed by a trusted authentication component. The basic functionality of the access control system with the components authorization and enforcement illustrates figure 2.9.

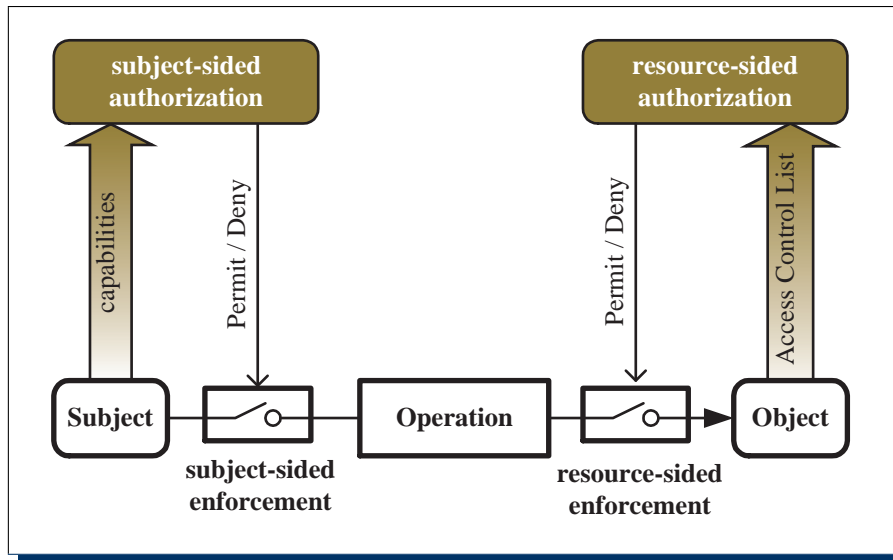


Figure 2.9: The basic functionality of an access control system

The authorization component is the legislative of the access control system. For deriving authorization decisions, a set of permissions is available that can be evaluated. The basic

model for the declaration of permissions uses positive permissions. Every access is denied, unless an explicit positive permission is available that matches the request, represented by the subject assertions, the operation and the resource(s). For simplicity, the subject assertions are restricted to a simple identity string. The declared access rights can be modeled, using an access control matrix. This matrix has the structure that the subjects occupy the rows of the matrix and the resource occupies the columns of the matrix. The fields of the matrix express the allowed operations. The following matrix shows a simple example, where Bob is allowed to write and read the resource Building and Alice is allowed to read the resource Building and Intersection.

$$AccessControlMatrix = \begin{pmatrix} & \text{Building} & \text{Intersection} \\ \text{Alice} & \textit{read} & \textit{read} \\ \text{Bob} & \textit{write, read} & \end{pmatrix}$$

The implementation of an enforcement component can be achieved in two ways: implementation on the subject or resource side.

For the implementation of the enforcement component on the subject side, the capabilities of a subject must be extracted from the matrix. This can be achieved by selecting the matrix fields of the corresponding row. This results in a so-called capabilities list of two-valued tuples: {object, operation}. For the above example, the capabilities list of the subject Bob contains the tuples {Building, read} and {Building, write}.

For the implementation of the enforcement component on the resource side, the so-called access control list is extracted from the matrix. It contains the two-valued tuples {subject, operation} for a resource. This can be achieved by selecting the matrix fields of the corresponding column. For the above example, the access control list of the resource Intersection results in the single entry {Alice, read}.

The primary concern of an access control system is to protect resources. Two major reasons exist, why the resource sided enforcement is widely used: The access restrictions on the resources is defined by a resource provider. When using the subject sided enforcement, the provider must trust that the enforcement is done correctly. Also, once a subject has received a valid capabilities list, it is difficult to revoke that list.

The major concern of a resource provider is that the enforcement is error-free. One aspect of the resource sided implementation of the enforcement component is that the changing of subjects require a modification/update of the access control lists. Role Based Access Control addresses this issue.

2.5.4 Introduction to Role Based Access Control

A Role Based Access Control (RBAC) addresses the challenge to maintain a consistent set of access rights. The resource sided implementation of the enforcement component, based on access control lists has one major disadvantage: The change of subjects has direct influence to the access control lists and requires its update. Because such an update has the potential to result in an inconsistent set of policies, it is desired to separate the subject from the access control list. This can be achieved by the role construct. The role is used as an indirection between the resource and subject side. It replaces the subject in the access control list.

Because the role defines a distinguishing set of access rights, required to perform specific tasks, the change of the role construct is less likely than the change of subjects. In particular in companies, the role can be associated with the job assignment in the company as defined in an organization chart. This enables a stable environment on the access control list.

In order to have subjects associated with particular access rights, a certain set of rules must be associated with the subject. Because the access to a resource is granted on the presentation of a particular role, the subject is required to have that role active in order to gain access.

The use of RBAC has -in addition to the previous issues- the advantage that it allows to model access rights based on job functions. A job function can be represented by a role that comprises required access rights. This enables the modelling of employee access rights by associating them to one or multiple roles, according to the job function.

Different RBAC extensions to the original model, known as RBAC₀, have been developed: RBAC₁ allows the structuring of roles by inheritance. One role can inherit all permissions from the superior role. RBAC₂ allows the definition of constraints that determine under which conditions a subject can make a passive role active. RBAC₃ supports the assignment of mutually exclusive roles and role hierarchy.

Even the RBAC is a very important model for access control, its capabilities do not provide the flexibility to declare the introduced access control requirements. Therefore, the introduced model is based on Rule Based Access Control. The permissions can be associated to subject assertions, which provides the flexibility to incorporate with RBAC by using the assertion Role¹⁰.

2.5.5 Introduction to Rule Based Access Control

A Rule Based Access Control system uses the Rule construct as its basic building block. It allows the declaration of positive and negative permissions. It can be represented as a four-valued tuple of the elements Subject, Operation, Resource and Condition, which effect is Permit or Deny.

$$Rule := \{SM_R, OM_R, RM_R, C\} \rightarrow \{Deny, Permit\} \quad (2.15)$$

The elements of the rule have the following meaning:

Subject (SM_R) represents a set of assertions, identifying the subject to which this rule is associated. An assertion can be an attribute value pair, represented by the triplet Name(value) := Type. The assertion about a role can be modeled as role(administrator) := RoleType. For simplicity, this work uses an abbreviation for the unique identification of subjects. Only the value is written but the name of the attribute and the type is suppressed. Instead of writing identity(Bob) := String, only the value Bob is being used.

Operation (OM_R) is an assertion with of a fixed name and type: operation(value) := OperationType. For simplicity, this work uses just the string, when referring to an operation. For example, write is short for operation(write) := OperationType.

¹⁰The introduced model is based on XACML, which allows to use a role based profile as explained in [OASIS 2004d].

Resource (RM_R) is a placeholder for an Xpath expression. Because resources are encoded in XML, a resource can be represented by the triplet $Xpath(value) := String$. For simplification, just the value is used.

Condition (C) is the construct that keeps a Boolean expression ($Condition \rightarrow \{True, False\}$). The result of that statement must result in True or False. The condition allows to declare complex constraints on all available information, such as environment, subject, operation or resource information. Examples are date, time or the geometry of a resource object.

The Rule construct allows to declare positive and negative access statements in a very flexible way. For example, the allowance that the subject **Bob** can write the resource **Building** if the address is '1600 Pennsylvania Avenue NW, Washington, DC 20500' can be declared by the following rule:

$$\{Bob, write, //Building, if(./address == '1600 Pennsylvania Avenue NW, Washington, DC 20500')\} \rightarrow Permit$$

Even the Rule construct is the basic building block of this type of access control, it may not live in isolation. A set of rules are combined by the Policy construct, using a particular combining algorithm. The combining algorithm compresses the outcomes¹¹ of a rule to one single value. In such a respect, a Policy can be modeled as a five-valued tuple. The first three elements are Subject, Operation and Resource. Similar to the Rule construct, these elements define the applicability of the policy. The fourth element is the CombiningAlgorithm that has the function to compress the outcomes of the rules to one single result. The fifth element represents a list of Rule constructs, where at least one rule is required.

$$Policy := \{SM_P, OM_P, RM_P, CA_P, Rule^+\} \quad (2.16)$$

The effect of a Policy cannot be defined. It results from the outcomes of the comprised rules and the combining algorithm. A combining algorithm is a function that transforms a vector of outcome (\vec{o}) values to one single outcome. Equation 2.17 formally defines the logic of the combining algorithm. Because the order of the outcomes is essential for some combining algorithms, the input must be defined as a vector and not as a set. The output of the algorithm is a single value.

$$CombiningAlgorithm := f(\vec{o}) \rightarrow \{N/A, Deny, Permit, Indeterminate\} \quad (2.17)$$

Table 2.5, page 48 defines different combining algorithms, which can be used for this model.

2.5.6 Enforcement of Declared Permissions

The enforcement of declared permissions can be characterized as the executive process in the access control system. It relies on the authorization process, which can be seen as the

¹¹The outcome of a rule is different from the effect (see section 2.5.8).

legislative. The enforcement process fetches required information from a particular request and forms an authorization decision request, which is sent to the authorization process. The response of the authorization process is the authorization decision, which instructs the enforcement process to block or accept the current request.

Therefore, the authorization process has the duty to derive an authorization decision based on a particular authorization decision request and a set of permissions. The introduced model encodes the permissions as a hierarchy of Policy and Rule constructs. Both constructs use a {Subject, Operation, Resource} (SOR)-tuple to define their applicability. The SOR_P -Tuple represents the applicability of the Policy construct and the SOR_R -Tuple represents the applicability of the Rule construct.

In order to derive an authorization decision, the authorization process must find applicable policies and rules for a given request. This can be achieved by matching Subject, Operation and Resource from a Policy or Rule construct with the request. This is possible, because the request can be characterized as a SOR-tuple.

$$Request := SOR = \{S, O, R\} \quad (2.18)$$

2.5.7 Finding Applicable Policies and Rules

The finding of applicable policies and rules is based on matching the SOR-tuples from the request and the policies and rules. This matching takes place using different matching functions. What matching functions are available depend on the implementation of the access control system. Examples of simple matching functions are `string-equal` or `regular-expression-equal`. The distinguishing difference between them is that `string-equal` only accepts two string values. The possible existence of regular expressions is not resolved. In contrast, the `regular-expression-equal` matching function processes regular expressions. This difference is important in respect to define valid values for the SOR-tuple.

The introduced model supports the identification of subjects by assertions. An assertion can be represented as an attribute-value-pair. The name of the attribute must be unique and the value must be valid for the type of the attribute. Examples of assertions are `name="Bob"` or `email="matheus@in.tum.de"`. Thus, the Subject element of the Request-tuple keeps a set of attribute-value-pairs that identify the issuer of the request. In contrast to that, the operation is identified by one specific attribute with a unique name. The value of that attribute keeps the name of the operation. Therefore, the Operation element of the Request-tuple is a regular string. The resource is identified by a valid Xpath expression, because one assumption of the model is that resources are encoded in XML, resp. GML.

For the purpose of finding matching policies and rules for a given request, the existence of matching algorithms is assumed. Each matching algorithm accepts two parameters and returns a Boolean value. The algorithm returns `True` for a positive match and `False` in case that no matching can be determined. The first parameter represents the Subject, Operation or Resource of the request and the second parameter the according element of the Policy or

Rule.

$$\begin{aligned} Match_S := Match(S, s) \Leftrightarrow \\ \forall assertion : Match(assertion(S), assertion(s)) \rightarrow True, s \in \{S_P, S_R\} \end{aligned} \quad (2.19)$$

$$Match_O := Match(O, o) \Leftrightarrow O \equiv o, o \in \{O_P, O_R\} \quad (2.20)$$

$$Match_R := Match(R, r) \Leftrightarrow XQuery(r, R) \neq \emptyset, r \in \{R_P, R_R\} \quad (2.21)$$

- 2.19:** $Match_S$ returns a *True* value if all assertions of the **Subject** element of the **Request** tuple and the **SOR** tuple of a rule or policy match. The generic **Match** algorithm is a placeholder for the actual defined matching algorithm. If one of the assertions does not satisfy the match criteria, $Match_S$ returns a *False* value.
- 2.20:** $Match_O$ returns a *True* value, if the string values of the **Operation** element of the **Request** and the **SOR** are identical.
- 2.21:** $Match_R$ queries the given resource content, represented by the **R** element for XML nodes, based on the Xpath expression of the resource element of the **SOR** tuple. If the set of fetched XML nodes is not the empty set, a *True* value is returned. In case that the resource content does not contain any XML nodes that match the Xpath expression, the value *False* is returned. **XQuery** represents the actual call to fetch the nodes of the resource XML document.

Based on the previous definitions, the matching of policies and rules for a given request can be defined in the following manner: A **Policy** or **Rule** is applicable to a given request if $Match_S$, $Match_O$ and $Match_R$ all return the value *True*. The set of applicable policies and rules build a subset to the overall available policies and rules.

$$\mathcal{P} \supseteq \mathcal{P}_A = \bigcup_i Policy_i \mid (Match_S \wedge Match_O \wedge Match_R) \rightarrow True \quad (2.22)$$

$$\mathcal{R} \supseteq \mathcal{R}_A = \bigcup_j Rule_j \mid (Match_S \wedge Match_O \wedge Match_R) \rightarrow True \quad (2.23)$$

- 2.22:** \mathcal{P} is the set of available policies. The set of applicable policies \mathcal{P}_A is defined as a subset of \mathcal{P} . It is defined as the union of all policies that satisfy the matching conditions for the **Subject**, **Operation** and **Resource** element. In case that all available policies match the request, the subset equals the set of available policies ($\mathcal{P}_A \equiv \mathcal{P}$).
- 2.23:** \mathcal{R} is the set of existing rules of a policy. The set of applicable rules \mathcal{R}_A is defined as a subset of \mathcal{R} . It is defined as the union of all rules that satisfy the matching conditions for the **Subject**, **Operation** and **Resource** element. In case that all existing rules match the request, the subset equals the set of available rules ($\mathcal{R}_A \equiv \mathcal{R}$).

The optional **Condition** of a **Rule** is not considered for finding applicable rules. The result from the evaluation of the condition is used for the determination of the effect of the **Rule**. According to the definition, a **Rule** returns the effect *N/A* if either the **SOR**-tuple does not match or the condition is not satisfied. Thus, the matching of the rule's **SOR**-tuple is the sufficient and essential pre-condition, before evaluating the condition.

2.5.8 Deriving an Authorization Decision

The procedure of deriving an authorization decision for a rule based access control model is based on two distinguishing characteristics: the outcome of a rule and the combining algorithm.

- The outcome of a rule can be characterized as the basic driver for the authorization process. The outcomes of enforceable rules are combined to one single authorization decision, based on the logic of the combining algorithm.

According to the introduced model, a Rule can have two different effects: Permit or Deny. The effect Permit represents a positive access (the specified subject and operation can be invoked on the resources) and the effect Deny represents a negative access (the specified subject cannot invoke the operation on the resources). From the matching process, the outcome of a rule can also be N/A, which represents the situation that either the SOR-tuple of the Rule does not match the given request or the Condition could not be satisfied¹².

This results in the definition of the set of enforceable rules, which is a subset of the applicable rules. Let \mathcal{R}_E be the set of enforceable rules, Outcome(Rule) be the function that returns the outcome of a rule, then \vec{o} is the vector of the outcomes from the enforceable rules.

$$\mathcal{R}_A \supseteq \mathcal{R}_E = \bigcup_i^{|\mathcal{R}_A|} \text{Rule}_i \mid \text{Condition} \rightarrow \text{True}, \text{Rule}_i \in \mathcal{R}_A \quad (2.24)$$

$$\vec{o} := \bigcup_j^{|\mathcal{R}_E|} \text{Outcome}(\text{Rule}_j), \text{Rule}_j \in \mathcal{R}_E \quad (2.25)$$

The set of enforceable rules, defined as \mathcal{R}_E contains all rules, which satisfy the matching criteria and where the Condition satisfies to true. This set of rules drives the authorization decision process.

- A combining algorithm has two purposes: First, it controls the processing of rules and second, it combines the outcomes of matching rules to one single outcome.

Depending on the logic of a combining algorithm, one or multiple rules are processed in order to determine the matching and the outcome. Some combining algorithms return the outcome of the first matching rule. Other compress the outcome of multiple matching rules to one single outcome, dependent or independent from the sequence of the rules. Therefore, the outcome depends on the three factors, from which two can be correlated: the logic of the algorithm, the sequence of the rules and the outcome of enforceable rules. Table 2.5 gives a list of different combining algorithms.

2.5.9 Illustrating the Decision Process

A policy is comprised of one combining algorithm and multiple rules. Each rule can have an optional condition, which declares additional constraints that control the enforceability of the

¹²An implementation can also define additional outcomes, which represent processing errors.

Combining algorithm	Semantics
deny-overrides	This combining algorithm returns Deny in case that at least one of the matching rules have the outcome Deny . This algorithm interrupts the processing if the first rule with a Deny outcome is found.
permit-overrides	This combining algorithm returns Permit in case that at least one of the matching rules have the outcome Permit . This algorithm interrupts the processing if the first rule with a Permit outcome is found.
first-applicable	This combining algorithm returns the outcome of the first matching rule and thus interrupts further processing of rules.
only-one-applicable	This algorithm processes all rules. It creates a set of enforceable rules and then determines if the order is greater one ($ \mathcal{R}_E > 1$). If that is the case, the combining algorithm returns a processing error. In case that the order is exactly one (1), the outcome of the enforceable rule is returned.
ordered-deny-overrides	This combining algorithm is identical to the deny-overrides algorithm with the exception that the processing of the rules is according to their sequence.
ordered-permit-overrides	This combining algorithm is identical to the permit-overrides algorithm with the exception that the processing of the rules is according to their sequence.
specific-general	This combining algorithm has the same logic as the first-applicable combining algorithm. But, it carries different semantics: The rules that are combined by this algorithm are correlated and declare a general/exceptional permission.

Table 2.5: Semantics of combining algorithms

rule. This structure defines two flows: the flow for matching purposes (control flow) and the flow for decision making (result flow). The flow for matching determines applicable/enforceable rules. The flow starts at the policy and continues to the comprised rules. The decision flow starts at the rules and continues to the combining algorithm, which returns one single outcome.

The visualization of such a structure can be based on a tree representation of a policy. Each node of the tree represents either a policy, rule or combining algorithm. The root node is restricted to be the start node. It defines the entry node for the control flow of the authorization decision. The leaves of the node represent the conditions of the rules. The edges of the tree have two directions. The direction from the root node to the leaves represent the control flow. The edges therefore keep the **SOR**-tuples for a policy (SOR_P) or a rule (SOR_R). The direction from the leaves to the combining algorithm represent the result flow and therefore keep the outcome from the rules / combining algorithm. For illustration

purposes, the following example policy and rule definitions are used:

$$P1 = \{*, *, \textit{Building}, \textit{first-applicable}, R1, R2\}$$

$$R1 = \{\textit{Bob}, \textit{write}, \textit{Building}, \textit{if}(\textit{address} == \textit{'3 Street A'})\} \rightarrow \textit{Permit}$$

$$R2 = \{*, \textit{read}, \textit{Building}\} \rightarrow \textit{Permit}$$

Policy P1 declares that it is applicable for all Request-tuples, that refer to the resource **Building**. It does not define any matching constraints to the Subject and Operation elements. It defines that the outcome of the containing rules are compressed, using the combining algorithm **first-applicable**. Rule R1 declares the access right that **Bob** is entitled to invoke the write operation on the resource **Building** if the resource has the address ‘3 Street A’. Rule R2 declares the access right that any subject can read the **Building** resource.

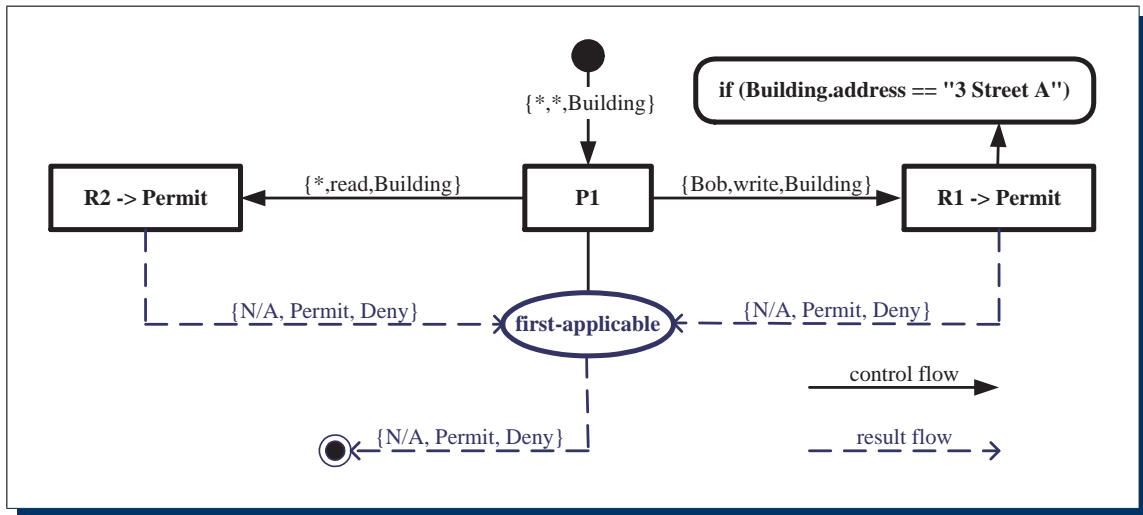


Figure 2.10: Representation of the example policy structure

This policy structure, as visualized in figure 2.10 can be used to demonstrate the deriving of an authorization decision. In order to do this, the following example Request-tuples are used:

$$Req1 = \{\textit{Bob}, \textit{write}, \textit{Building}(\textit{address} = \textit{'3 Street A'})\}$$

$$Req2 = \{\textit{Alice}, \textit{read}, \textit{Building}(\textit{address} = \textit{'3 Street A'})\}$$

$$Req3 = \{\textit{Alice}, \textit{write}, \textit{Building}(\textit{address} = \textit{'3 Street A'})\}$$

Deriving an authorization decision for request Req1

First, the request Req1 is matched against the policy P1, which results in a positive match. Therefore, the further processing of the request is controlled by the combining algorithm **first-applicable**. Next, the request is matched against Rule R1. This matching is positive, which requires the processing of the condition. The evaluation of the condition can be satisfied, which triggers the rule R1 to return the outcome *Permit* to the combining algorithm. Because the combining algorithm **first-applicable** stops after receiving one outcome of a rule, the rule R2 is not processed. This results in the policy outcome *Permit*.

Deriving an authorization decision for request Req2

First, the request Req1 is matched against the policy P1, which results in a positive match. Therefore, the further processing of the request is controlled by the combining algorithm *first-applicable*. Next, the request is matched against Rule R1. This matching is positive, which requires the processing of the condition. Because the condition cannot be satisfied, the rule returns *N/A* to the combining algorithm. This requires the processing of the rule R2. Here, the matching is positive and because the satisfaction of the condition is always given, the rule returns the outcome *Permit* to the combining algorithm. This results in the combining of the values *N/A* from rule R1 and *Permit* from the rule R2, which results in a final *Permit*.

Deriving an authorization decision for request Req3

First, the request Req1 is matched against the policy P1, which results in a positive match. Therefore, the further processing of the request is controlled by the combining algorithm *first-applicable*. Next, the request is matched against Rule R1. This matching is negative, which triggers the rule R1 to return the outcome *N/A*. Next, the request is matched against the rule R2 which also results in the outcome *N/A*. Because no more rules exist, the combining algorithm stops processing. The combination of the outcome values *N/A* and *N/A* results in a final *N/A* result.

2.5.10 Different Strategies for the Declaration of Access Restrictions

Any access control system must regulate the access to protected resources, according to the declared permissions. For a distributed access control system, where the enforcement is separated from the authorization, it is important that the authorization decision is unambiguous. Because the enforcement can either block or accept a request, the authorization decision must be in a binary form. If more than two values are allowed for the authorization decision, the enforcement must understand the value and map it to block or accept. This requires additional information, such as a mapping table. For example, a unique authorization decision can carry the values *Permit* or *Deny*. If the value *N/A* is present, the enforcement must choose a fail-safe action, which typically results in blocking the request.

The introduced model of Rule Based Access Control supports three different strategies for the declaration of the permissions. Each strategy has pros and cons, which are discussed in more detail:

1. The *all-permitted* strategy assumes that access to resources is permitted if not explicitly denied through declared negative permissions ($\{\text{Subject, Operation, Resource, Condition}\} \rightarrow \text{Deny}$). This implies that the enforcement process must interpret all *N/A* authorization decisions as *Permit*.

This strategy is favored if most of the access is permitted. Thus, a very small set of negative permissions is required. However, this strategy keeps an important risk: all errors in the system, regardless of the origin can result in an erroneous acceptance of the request. Not all existing errors must exploit to an unintended access, but bare a potential, which must be eliminated under all possible circumstances. The two main risks are that

- not all requests, which are to be denied are clearly defined or known. Erroneous and missing permissions result in a N/A decision and therefore provide unallowed and unintended access.
- a spelling error in the policy, rule or condition can result in a N/A decision. This would also result in unallowed access.

Another contra to this strategy is that it is impossible to test an existing set of policies for completeness. This is because all unmatched requests result in the not applicable outcome. The correctness can only be tested according to defined test cases. If the list of test cases is incomplete or incorrect, the access control system might be incorrect.

2. The all-denied strategy assumes that access to resources is denied if not explicitly permitted through positive permissions ($\{\text{Subject, Operation, Resource, Condition}\} \rightarrow \text{Permit}$). This implies that all N/A authorization decisions are interpreted as Deny.

This strategy is favored if most of the access is prohibited. The distinctive strength of this strategy is that it is fail-safe. All errors if not resulting in a Permit decision due to an erroneous permission, result in the blocking of the current request.

Similar to the all-permit strategy, it is not possible to verify the correctness unless using defined test cases. But, the fail-safe characteristics does not exploit protected resources. A missing positive rule can result in an erroneous blocking, which can be announced by the user.

3. The all-explicit strategy assumes that all possible access requests result in a Permit or Deny authorization decision. The N/A authorization decision is not admitted.

Thus, this declaration strategy requires the most policies and rules. However, it has the important strength that a N/A authorization decision indicates an error. This can trigger the authorization process to generate an error message for the policy writer to correct the cause of the error. In order to have the enforcement process take the fail-safe action, the authentication result can be altered to Deny, before send to the enforcement process.

For a distributed access control system, where the authorization decision and the enforcement process are separated, the enforcement process does not know the access control strategy upon which the decision process derives the decision results. For the introduced declaration strategies, the N/A decision must be interpreted in a different way: For the all-permit/all-deny strategy, the interpretation is accept/block of the request. For the all-explicit strategy, it indicates an error and must therefore result in the fail-safe block of the request. In an environment, where an enforcement process depends on different authorization decisions, different strategies can drive the authorization decisions. In cases, where the all-permit and all-deny strategies are allowed, additional metadata must be added to the authorization decision. This metadata controls the mapping of different authorization decision values to the actions block or accept.

2.6 Introduction to Distributed Access Control

Distributed access control differentiates from the traditional access control in such a way that the authentication, authorization and enforcement processes are implemented as autonomous components. The resulting infrastructure and the information flow is illustrated in figure 2.11.

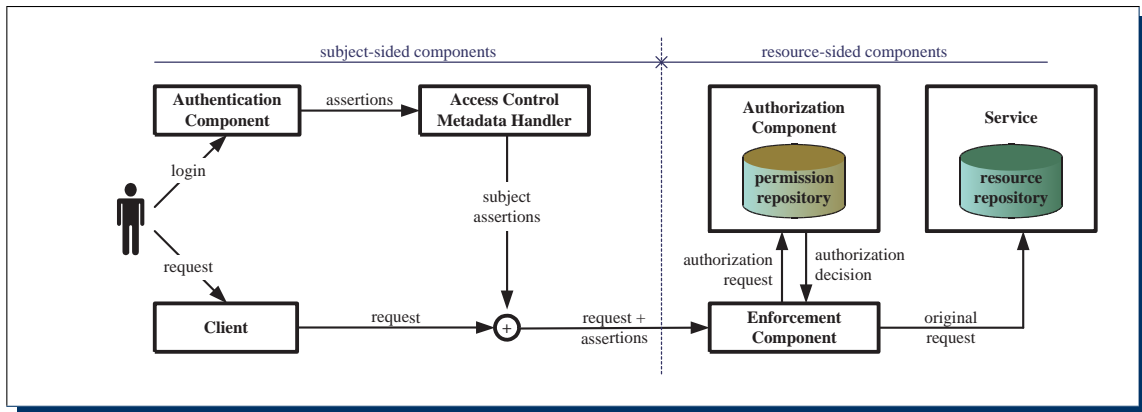


Figure 2.11: Infrastructure and information flow of distributed access control

The use of such an infrastructure results in different constraints and requirements that must be satisfied.

The separation of the Authentication Component presents a single point, where the user can login in. The Authentication Component provides a set of assertions about the user. They are either be forwarded to a particular handler or kept for later enquiries. This work does not cope with the authentication itself. It simply uses the assertions for matching permissions. For simplicity, it is assumed that the assertions are forwarded to an appropriate handler, such as the Access Control Metadata Component. This component has the duty to fetch the original request, issued by a client on behalf of the user in order to add relevant access control metadata. Here, the adding of subject assertions is required. In case that conditions exist, which require more environment information such as date, time, etc., this information must also be added by this handler. The output of this handler is the actual request as it is send to the target service.

Before the request reaches the target service, the request is fetched by the Enforcement Component. It detaches the access control metadata from the original request. Based on information from the metadata and the original request, an authorization decision request is created. This is send to the Authorization Component, which derives an authorization decision. That decision is returned to the Enforcement Component and used to block or accept the original request. In case that the request is blocked, the Enforcement Component returns an error message to the client. In case that the request is accepted, it is forwarded to the actual service. The service response is then forwarded to the client.

From this functional overview, the requirements for a distributed access control can be derived:

- The communication between the components must use a well defined syntax and se-

mantics in order to guarantee the appropriate interpretation of message content. This results in the demand for a language, which defines that syntax and semantics.

- Trusted relationships must exist between the components. The Authorization Component relies on the correctness of the authorization decision requests. Vice versa, the Enforcement Component relies on the authorization decision of the Authorization Component. Enforcement and authorization rely on the subject assertions, created by the Authentication Component.
- The communication security is also very important. It must be established in connection with the trust relationships, in order to guarantee a certain level of overall security.
- Unique identification of subjects and resources is required.

Even these aspects are important to establish a distributed access control, security related aspects are outside the scope of this work. The interested reader is redirected to different standards, which enable the establishment of the highlighted topics:

- XML Encryption by W3C [[W3C 2002a](#)]
- XML Signature by W3C [[W3C 2002b](#)]
- SOAP Message Security standard by OASIS [[OASIS 2004f](#)]
- X.509 Certificate Token Profile 3 by OASIS [[OASIS 2004g](#)]
- UsernameToken Profile 1.0 3 by OASIS [[OASIS 2004e](#)]
- SAML by OASIS [[OASIS 2004b](#)]

[[Chen 2004](#)] provides in his work a classification of communication threads for an environment of distributed Web Services and how to establish communication security based on the existing standards.

The focus of this work is the declaration and enforcement of access restrictions for geospatial information objects. In respect to distributed access control, one requirement results from the aspect that one Authorization Component can derive decisions for multiple Enforcement Components. In such an environment, it must be possible to associate a particular authorization request with the corresponding set of permissions. This aspect must be taken into account, when developing the access control model.

Because the declaration and enforcement relies on the subject assertions provided by the Authentication Component, a particular syntax and semantics must be defined for the markup of these assertions.

2.6.1 Standardized Language for Assertions

Due to the separation of the authentication component and the authorization component, a common language is required to define assertion attributes and their structure. The assertions, issued by the Authentication Component are used by the Authorization Component

to match policies and rules. This can only work out if both sides obey the definition of an assertion attribute, the type and their structure. For example, if the assertion for an email address is defined as $\text{email}(\text{matheus@in.tum.de}) := \text{RFC822Name}$, the syntax and the meaning is unambiguous. Assuming the attribute email is a property of the Subject element, an example usage in a rule can be like this:

$$\text{Rule} = \{\text{email}(\text{matheus@in.tum.de}) := \text{RFC822Name}, \dots, \dots\} \rightarrow \dots$$

The declared rule matches a request if this assertion matches. This requires the definition of matching algorithms for the different assertion types. In order to match the email assertion, a RFC822Name matching algorithm is required. The next chapter introduces different standards that provide the capabilities to define assertions and their markup in XML.

Because the Authorization Component relies on the assertion attributes, issued by the Authentication Component, it is essential to establish trust. As described in [Johnston et.al. 1998], this can be achieved using certificates. They state that widely distributed access control requires the authentication by certificates. In their motivating example, different stakeholder exist that specify access restrictions on common resources.

2.6.2 Standardized Permission Language

In the distributed access control environment, the standardized permission language has two duties:

1. Define common language constructs to express the access statements in an unambiguous way. The policy writer must understand and agree to the language constructs, their meaning and their structure. This is essential that the authorization component processes the permissions in a foreseeable way.
2. Define common processing of the permissions in order to derive an authorization decision. This refers in particular to the logic of combining algorithms: How they control the processing of permissions and which result they return for a certain input.

The next chapter evaluates related standards to determine in how far this requirement is met.

2.6.3 Standardized Communication Between Components

Due to the separation of the Enforcement Component and the Authorization Component, the need for a standardized language arrives that defines a structure of the authorization request and response messages. The authorization request, which is sent from the Enforcement Component to the Authorization Component contains the information about the subject assertions, the operation and the resource content. The resource content is an XML encoding of the resources, which are addressed by the request.

The next chapter evaluates related standards to determine in how far this requirement is met.

Chapter 3

Access Control Requirements and Related Standards and Systems

This chapter starts with introducing the access control requirements as they result from an informal poll at the InterGeo 2002. It further evaluates different standards and quasi standards, in how far they support the requirements as they resulted from the poll. Also, capabilities of different systems are evaluated in how far they support the infrastructure constraints and access control requirements. This chapter concludes with the comparison and quantifies the standard's usability.

3.1 Access Control Requirements

The demands for restrictions of data providers for an access control system have been collected from talks to government and private geodata providers on the InterGeo 2002. The resulting access control restrictions are condensed in the following subsections.

3.1.1 Enforce Restrictions on the Resources, Independent from the Service

For a service based geodata infrastructure, the declaration and enforcement of access restrictions must be independent from the used services. This is because one service operation can carry different operations to be invoked on the resource(s). In such respect, the same operation can obtain access to different resources using different access modes, depending on the operation parameters.

One example of such a service is the Web Feature Service ([OGC 2002-058]). The service operation *Transaction* allows to invoke the operations *Delete*, *Create* and *Update* on the accessible resources.

3.1.2 Class- and Object-Based Requirements

It must be possible to declare and enforce access restrictions based on an object-oriented data model. In such respect, restrictions can be declared for a class or individual object(s).

The class-based restrictions must be enforced for all instances of the class. The object-based restrictions must be enforced for particular objects, as referenced by the declared permission. Assuming an object-oriented data model of geodata, accessible through the services infrastructure, the service response/request information must also be modeled according to an object-oriented data model. This is given by the GML encoding of the exchanged geodata, according to the simple feature specification ([OGC 1999-100r1]).

Based on GML markup, objects are represented by features. The characteristics of a feature is modeled by recursive structured sub-elements and attributes. Because GML uses the XML encoding, the required referencing to classes (feature types) and particular objects (features) can be achieved by using Xpath ([W3C 1999]) expressions.

An example for a class-based access permission is that Bob can read all objects of class `Building`.

$$\{Bob, read, //Building, \epsilon\} \rightarrow Permit$$

An example for an object-based access permission is that Alice can write information to the instance of the class `Building`, where the address equals "1600 Pennsylvania Avenue NW".

$$\{Alice, write, //Building, if(address == "1600PennsylvaniaAvenueNW")\} \rightarrow Permit$$

3.1.3 Spatial Requirements

An obvious requirement results from the geographic characteristics of the geospatial information objects. Geographic or geospatial information objects describe real world entities, which have a spatial reference; hence a location relative to the surface of the Earth and a shape, expressed by geometry. The interviewee named the spatial access control requirement as essential for this type of data. The requirement is that spatial restrictions allow to declare permissions, based on the spatial relation between a feature's geometry and a given geometry; the permission geometry. Two important spatial relations were named in the informal poll: `Within` and `Touches`.

The `Within` relation allows to restrict access inside a given area; the permission area. For example, employees of a municipal government are allowed to access particular resource objects if their geometry satisfy the given topological relation with the geometry of their sovereign territory. The other spatial relation is `Touches`, which restricts the access to objects, where both geometries are adjacent to each other. An example use of this restriction is that a subject, which has access to a particular land parcel can also access all land parcels if both geometries satisfy the given topological relation.

The spatial relations are not declared in isolation. They must be combined with a class-based restriction. In such a respect, the spatial restriction can be seen as a constraint to the class-based access restriction, which limits the access to objects, based on their geometry.

An example, according to the spatial access restriction is that Bob can read all objects of class `Building`, if the geometry (expressed by the property `shape`) is within the area `{foo, 0 0,3 0,3 4,0 4,0 0}`.

$$\{Bob, read, //Building, if(Within(shape, \{foo, 0 0, 3 0, 3 4, 0 4, 0 0\}))\} \rightarrow Permit$$

3.1.4 Representational Requirements

This kind of requirements focus on the representation of the geodata, primarily as a map. For example, geodata that is represented as a binary map image can be unrestricted if the scale does not exceed a particular threshold. An example access right of this kind that Bob can request a binary map with a scale less than 1:25.000.

Another kind of representational requirement is that particular rendering styles are restricted. For example, the request of b/w weather maps is free, but the request of colored weather maps is restricted.

In addition to the graphical representation, the data format can result in a restriction. A map in a vector format can potentially be used for further processing. Therefore, request of maps in vector format can be restricted: Bob can request maps in format image/svg.

3.1.5 Temporal Requirements

Some of the interviewed data providers argued that it is relevant to restrict the access based on time and date. Some said that access to the service (and hence to the geodata) may only be allowed between business hours from Monday to Friday, 9.00 am to 5.00 p.m.¹ Another temporal requirement is to restrict access relative to the current local system time. As example, the request of weather or doppler-radar maps is for free if older than a certain time window. In contrast, the request of actual maps can be restricted.

3.1.6 Communication Security Requirements

Some interviewed people argued that the request of confidential or personal information can only be granted if particular communication security aspects are met. The classes of an object-oriented data model can be classified according to the security requirements: confidentiality, integrity and accountability. If objects of classified classes are accessed, the corresponding security aspects must be satisfied.

An example for this kind of restriction is that objects of class **Building** can only be accessed if the connection is based on HTTPS. Another, more complex example is that objects of class **Building** can only be accessed if the service output is confidential for the subject, which must use an X.509 certificate for identification, issued by a certain trust center.

3.2 Framing the Problem Space and Identifying Infrastructure Constraints

Many important kinds of geographic data are collected and maintained by different government agencies and organizations. The combined use of that distributed and heterogeneous geodata build the fundamental basis for decision making and geoprocessing for different purposes. Therefore, it is the key issue to make the distributed and heterogeneous geodata

¹The time is the local service time, independent from the local time of the user.

available in an interoperable way. This work is based on the service oriented infrastructure ([OGC 1999-100r1]), implemented as Geo Web Services.

The owner of the geodata defines access restrictions for protected geodata. Because the geodata is accessible through Geo Web Services, which have the capability to transform the local data model and structure into an interoperable data model and structure on the interface level, these restrictions must be declared for the service result, the resource content. Because this work assumes the GML markup of object-oriented geodata, the access permissions can reference the protected resource objects through Xpath expressions. This enables the declaration and enforcement of class-based and object-based restrictions. The declaration and enforcement of spatial access restrictions can be based on functions, testing the satisfaction of particular topological relations between two geometries.

The distributed constraint requires that a standardized language provides a vocabulary for the declaration of access restrictions in an unambiguous way and defines the process of enforcement of the declared access restrictions in a transparent way.

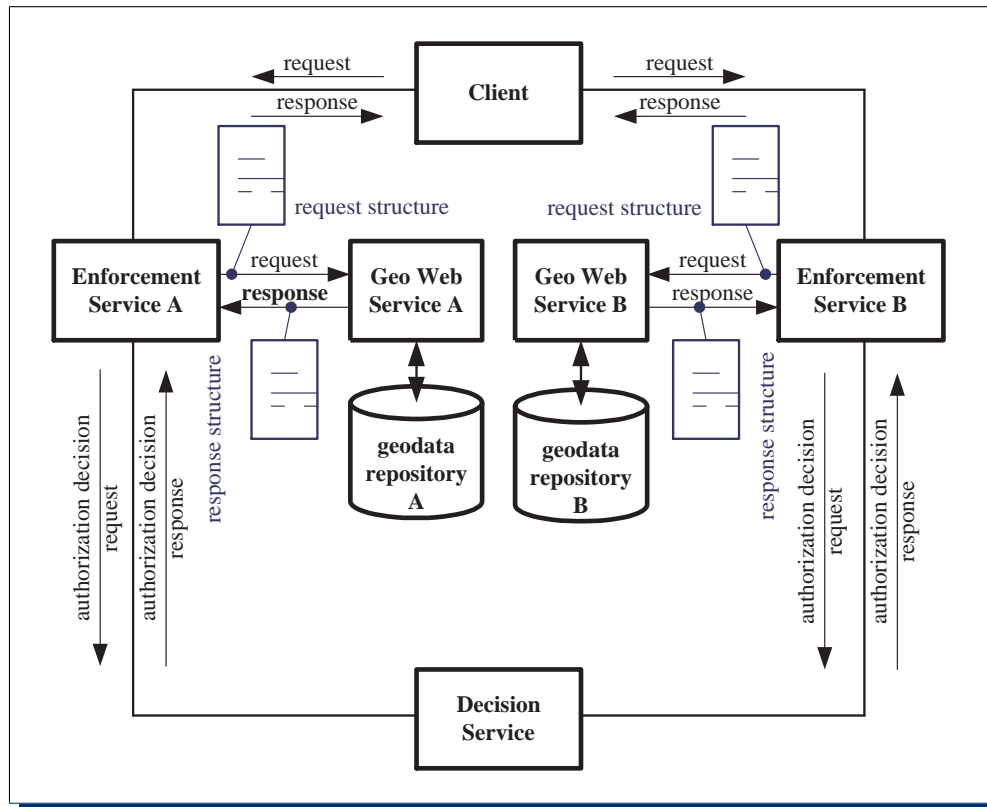


Figure 3.1: The service infrastructure for online access to protected resources

Figure 3.1 illustrates the initial infrastructure of online access through Geo Web Services. Each service accepts a designated set of parameters on a particular operation. The operation provides the functionality to access the geodata store and fulfills the request. The Enforcement Service intercepts the request to the service, isolates the authorization metadata and service operation parameters from the request. An authorization decision is either requested from the information of the request or the service result. The Authorization Service answers the

request with a Permit or Deny statement, upon which the Enforcement Service makes the service result available to the requestor.

The requirements for an access control standard are summarized in table 3.1.

Requirement	Explanation
Unambiguous vocabulary	The language that supports the declaration of permissions must support a naming convention, which is extendable and unambiguous. This is required because all involved parties must understand and agree on the syntax and semantics of the language.
Extendable	Required, to declare and enforce access restrictions which are problem domain specific. In this context, the declaration and enforcement of spatial restrictions is a domain specific problem.
Define authorization process	The access control language must define a generic processing of authorization decision requests. For the introduced rules and policies, this can be achieved through the combining algorithm, the matching and the declaration of conditions.
Define language structure and processing rules for authorization decision request and response	In an environment, where the enforcement and decision process are separated, it is required that the access control language defines the structure of authorization decision requests and responses as well as their processing. In addition, it is important that the assertions and the resource content can be defined in an unambiguous way.
Define vocabulary and format for authorization metadata	The access request to a service, issued by the client is intercepted by the Enforcement Service. The request contains service specific parameters and authorization metadata, which is relevant for the authorization decision request. Therefore, the Enforcement Service must understand the used vocabulary and structure of the authorization metadata.
Capable for declaration and enforcement of named restrictions.	Among other requirements, the capabilities of the access control requirements must allow to declare and enforce the restriction, as highlighted in the previous section. In particular the requirements for class- and object-based restrictions as well as spatial restrictions must be realizable.

Table 3.1: Requirements for access control standards

3.2.1 Constraints from the Distributed Aspect

The enterprise wide enforcement of declared access restrictions can be based on multiple or one central Authorization Service. The policy language must provide the capability to relate access permissions to data, exchanged with a particular service. This is required for the centralized authorization, because different enforcement services request authorization decisions: each service for a particular resource content.

In order to ensure reliable access control in a distributed environment, it is essential that

the Subject, Operation and Resource are uniquely identified. Because it is difficult to ensure globally unique identifiers, an access control domain can be defined. All subjects, operations and resource objects, which belong to that domain have a unique identifier. The authentication process assigns a unique identifier to each user. The uniqueness of operations and resource objects is simply based on an agreement between cooperating parties. In [OGC 1999-049] an informative section describes how to ensure unique identifiers for the environment of distributed Geo Web Services, providing access to distributed geospatial information objects. Assuming that resources have a unique local and domain wide identifier, the policy language must support to express this identifier.

3.3 Standards for Distributed Access Control

The focus of this work is the declaration and enforcement of access restrictions based on a distributed service infrastructure. For the declaration, it requires a language to formulate the introduced restrictions (see 3.1). For the enforcement, an Authorization Service is required, which delivers an access decision for a particular request. In particular, an authorization process must be implemented that derives the authorization decision based on a particular request and the set of permissions, available in the permission repository.

In this regard, the following standards are evaluated, what capabilities they provide for the declaration and enforcement. This chapter concludes with a comparison of the introduced standards and judges their useability for this work.

3.3.1 Security Access Control Markup Language (SAML)

The Security Access Control Markup Language (SAML) ([OASIS 2004b]) is an international standard of The Organization for the Advancement of Structured Information Standards (OASIS) ([OASIS 2004c]).

SAML defines a framework for exchanging security information between online business partners. For this purpose, SAML defines XML schemas that define the markup of the security information, called assertions. The assertions are being used for exchanging authentication and authorization information. There is always an asserting party and a relying party. The asserting party creates assertions about subjects which are being used by the relying party. For example, an assertion about a subject can be that its email address is *matheus@in.tum.de* and that this was proven by an X.509 certificate. The relying party can use this assertion for an authorization decision. Thus, a trust relationship must exist between the relying and the asserting party. This enables the Single Sign On (SSO): A subject's identity is proven once by the asserting party and all relying parties base the access control enforcement on the assertion. Thus, the user does not have to authenticate itself with each relying party.

The SAML framework is based on different key concepts.

Assertions define an information set that includes one or more statements about Authentication, Attribute and Authorization decisions. The Authentication assertion gives the information that a subject was authenticated by a specific means at a particular time. The Attribute assertion gives more characteristics of the subject, like his email address.

The Authorization decision assertion identifies what the subject is entitled to do. SAML defines the XML Schema for XML markup of assertions.

Protocol: SAML defines a request/response protocol mechanism for querying and obtaining assertions. A SAML request can either request for known assertions or for a particular Authentication, Attribute or Authorization assertion. SAML defines the XML schema for XML markup of the request/response messages.

Bindings specify, how SAML assertions are mapped onto transport specific protocol, such as HTTP Post or SOAP.

Profile defines the sequence of SAML messages for a particular purpose.

According to the pre-requisite of declaration and enforcement of access restrictions, SAML can be used to add the access control required metadata to the request from the client to the service. SAML can also be used for establishing interoperable communication between the Enforcement Services and the Authorization Service.

However, SAML cannot be used for the declaration of access restrictions. Hence writing the permissions, which are used by the Authorization Service to derive an access decision is not possible.

3.3.2 XML Access Control Language (XACL)

XML Access Control Language (XACL) is a research project at IBM ([[IBM et. al. 2002](#)]). It provides the means for declaring access restrictions on XML encoded resources. XACL also defines the Provisional Authorization Architecture, which allows the definition of certain actions to be taken by the Enforcement Service before or after the access to the resource is allowed. XACL is not an extendable language.

The policy is a container for permissions, based on the {Subject, Operation, Resource, Condition} tuple, as introduced in equation 2.15. The XACL DTD defines the exact structure of the policy.

Subject: A subject must be identified by a name or optionally by a group or role.

Resource: Resources are selected by an Xpath expression.

Operation: The allowed operations (actions) are limited to read, write, create and delete.

Condition: A Boolean expression that evaluates to *True* or *False*.

Effect: The outcome of a policy can either be *grant* or *deny*.

XACL defines the means for declaration and enforcement of restrictions to XML encoded resources. In this respect, XACL can be used to protect the access for GML encoded geo-data. However, XACL lacks about extensibility, which prevents the declaration of spatial restrictions. XACL does also not declare the message structures, exchanged between an Authorization Service and the Enforcement Services and a client and a Geo Web Service.

3.3.3 eXtensible Access Control Markup Language (XACML)

The eXtensible Access Control Markup Language (XACML) is an international standard from OASIS ([OASIS 2003]). It describes a general purpose policy system, which supports the most common needs to establish an authorization decision in a distributed environment. The distributed environment is similar to the environment introduced in figure 3.1, page 58. In XACML terminology, the Enforcement Service is called the Policy Enforcement Point (PEP) and the Authorization Service is called the Policy Decision Point (PDP).

XACML supports the declaration and enforcement of access restrictions through policies, based on three types of definitions:

Policy syntax: XACML defines XML Schema syntax for a policy language, which supports the declaration of access restrictions in an interoperable way. Interoperability is ensured by the use of well known data types and functions. XACML supports simple datatypes and provides extension points for the definition of complex (structured) data types, according to the XACML Schema capabilities. XACML also supports different functions and provides extension points for the definition of new, problem specific functions. This gives the freedom to adapt XACML to express almost all access restrictions.

Policy processing: XACML defines the semantics for processing existing access permissions in order to derive an authorization decision. It also defines the processing of policy obligations in the PEP. Obligations are certain actions, which are to be taken by the PEP after receiving a decision response from the PDP.

Request/response format: XACML defines the structure of messages, exchanged between the PEP to the PDP. The authorization decision request is send from a PEP to the PDP. It comprises of attributes that describe the current request and an XML encoded resource content. After the PDP has derived an authorization decision, the decision result is send back to the PEP.

The declaration of access restrictions is expressed by using attribute value tuples. This gives the freedom to either use pre-defined attributes or to define new, problem specific attributes. This allows the association of permissions to subjects through any attributes. The association of permissions to a subject identity can happen by using the XACML attribute `subject-id`. It is also possible to define a RBAC, as illustrated in [OASIS 2004d]. XACML defines two different mechanisms, how to resolve attributes in the policy logic: `AttributeDesignator` and `AttributeSelector`. The `AttributeDesignator` allows the selection of simple attributes from the decision request message and the `AttributeSelector` enables the selection of attributes from an XML formatted request content, using Xpath expressions.

Intercommunication between PEP and PDP

XACML 'at work' can be explained, using figure 3.1. The workflow of XACML empowered transactions start, when the PEP (Enforcement Service in figure 3.1) receives a request. The PEP extracts all available access control relevant metadata from the request. In particular, the PEP fetches the information about the initiator -the Subject-, the resources and the operation to be invoked on the resources. In addition, other information such as the date

and time of the request an other environmental information can also be fetched. The PEP then creates the decision request message, which is to be send to the PDP (Authorization Service in figure 3.1). The XML encoded structure of the authorization decision message, according to the XACML schema definition is shown in figure 3.2. The PDP tries to match permissions against the information from the decision request. The PDP returns a decision to the PEP, which might include obligations. After the PEP has received the decision result, it interprets the information and either forwards the request to the resource or service, it denies the request or it allows the request with processing of the obligations. The PDP can request that the obligations are carried out before the PEP forwards the request or after the request has completed and before the result is forwarded to the initiator. The PEP function can be characterized as a resource specific mediator, intercepting communication between clients and a service, providing access to protected resources.

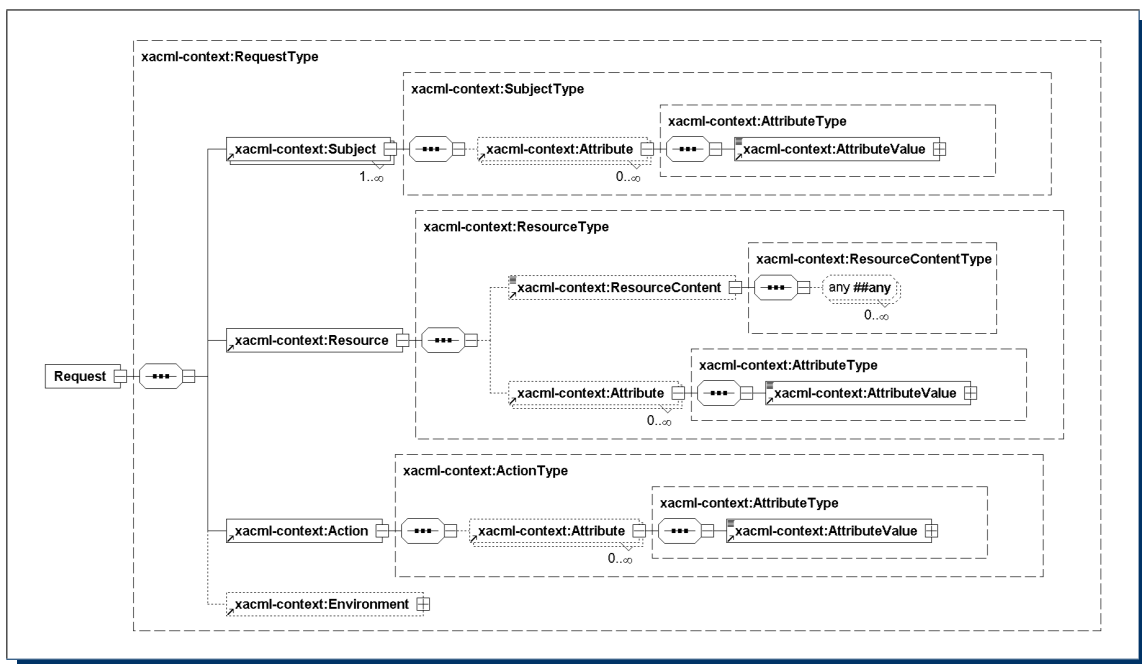


Figure 3.2: XML Schema definition of the XACML authorization decision request

The PolicySet, Policy and Rule Hierarchy

XACML defines XML tags, which are involved in the declaration of permissions. The basic building block is the Rule element, which holds a permission, according to definition 2.15, page 43. A sequence of Rule constructs is comprised in a Policy and a set of Policy elements is comprised in the PolicySet element. The coherence between the elements is shown in figure 3.3.

The Target element, which is defined for the PolicySet, Policy and the Rule element hold the {Subject, Operation, Resource} tuple through the elements Subject, Action, Resource. The Target element is being used for matching applicable policy sets, policies and rules to a given request.

The matching is controlled by attribute value pairs, which can be an AttributeSelector

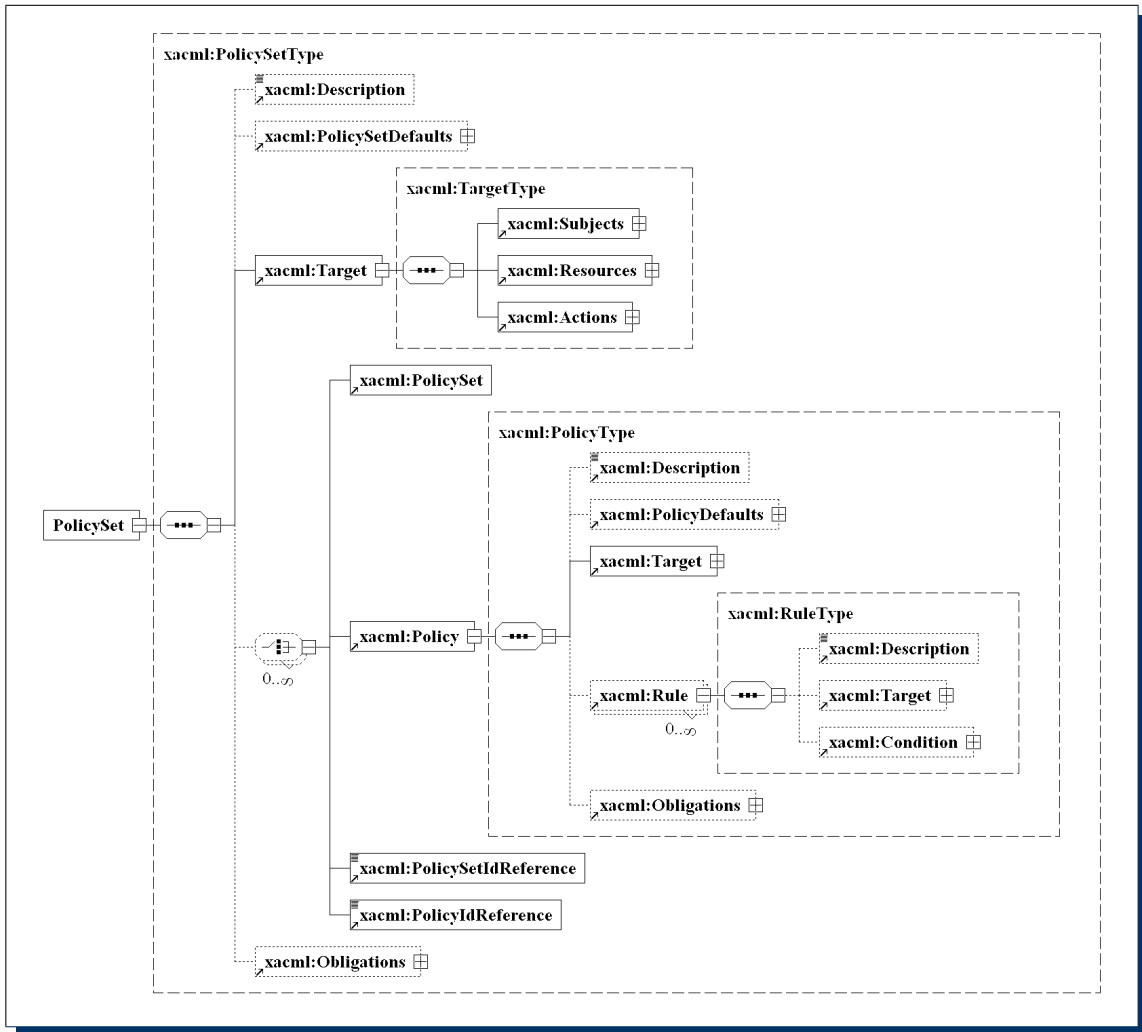


Figure 3.3: XML Schema definition of the XACML PolicySet element

or `SubjectAttributeDesignator`. The `AttributeSelector` allows the fetching of simple attribute values from the `ResourceContent` element, which is part of the authorization request. The `SubjectAttributeDesignator` allows the fetching of simple `AttributeValue` elements from the `Subject` element and match the value against the attribute value of the request. Figure 3.4 shows the `Target` element structure and in detail the internal structure for the `Subject` element. The structure for the elements `Resource` and `Action` is similar to the `Subject` element, but the name of the designator elements is different: The `SubjectAttributeSelector` is named `ResourceAttributeSelector` for fetching attributes from the `Resource` element, resp. `ActionAttributeSelector` for fetching attributes from the `Action` element. The subject matching is shown in figure 3.4.

The matching algorithm for comparing the values of the `AttributeValue` and the `AttributeSelector`, resp. `AttributeDesignator` can be declared by using the `MatchId` attribute of the surrounding element `SubjectMatch`, `ActionMatch` or `ResourceMatch`.

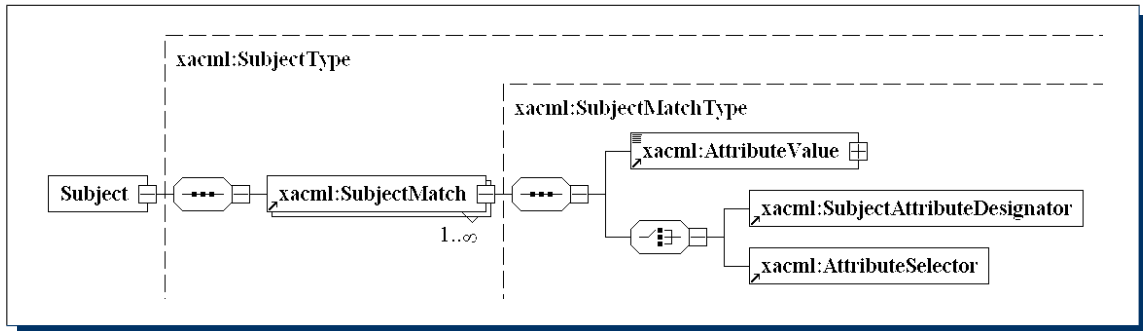


Figure 3.4: XML Schema definition of the Subject matching

The Condition Element

The Condition element allows the declaration of complex conditions, which drive the outcome of a rule. Listing 3.1 shows the XML Schema structure of that XACML element. Here, just a brief introduction highlights the capabilities as required in this work. For further details, please see [OASIS 2003].

```

1 <xs:element name="Condition" type="xacml:ApplyType" />
2 <xs:complexType name="ApplyType" >
3   <xs:choice minOccurs="0" maxOccurs="unbounded" >
4     <xs:element ref="xacml:Apply" />
5     <xs:element ref="xacml:Function" />
6     <xs:element ref="xacml:AttributeValue" />
7     <xs:element ref="xacml:SubjectAttributeDesignator" />
8     <xs:element ref="xacml:ResourceAttributeDesignator" />
9     <xs:element ref="xacml:ActionAttributeDesignator" />
10    <xs:element ref="xacml:EnvironmentAttributeDesignator" />
11    <xs:element ref="xacml:AttributeSelector" />
12  </xs:choice>
13  <xs:attribute name="FunctionId" type="xs:anyURI" use="required" />
14 </xs:complexType>

```

Listing 3.1: XML Schema definition of the <Condition> element

Unlike the PolicySet, Policy and Rule element, the Condition element supports the definition of constraints on all types of attributes by the SubjectAttributeDesignator, ActionAttributeDesignator, ResourceAttributeDesignator and EnvironmentAttributeDesignator. Also, complex conditions can be defined, using the Apply and Function elements. In contrast to this mechanism, the matching on the Target element supports only simple string matching. This is reflected by XACML by declaring the applicability of the matching functions (see [OASIS 2003][A.14]).

Listing 3.2 presents the example, where the resource objects are checked if at least one particular instance exists: The instance of the class Building, where the value of the sub-element Name is equal to House A.

Lines 5-7 select the string values from the property Name of all resource objects of class Building. Lines 3-4 define the string value for the matching: "House A". Line 2 defines the matching function for the result from lines 5-7 and lines 3-4. In line 1, the Boolean match function is defined that is applied to the matching results from the function, defined in line

```

1 <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
2   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
3   <AttributeValue
4     DataType="http://www.w3.org/2001/XMLSchema#string">House A</AttributeValue>
5   <AttributeSelector
6     RequestContextPath="//CityModel/gml:featureMember/Building/name/text()"
7     DataType="http://www.w3.org/2001/XMLSchema#string"/>
8 </Condition>

```

Listing 3.2: XACML `<Condition>` element expressing an example condition on the existence of a particular instance of class `Building`

2. If the result is *True*, the effect of the rule is considered in the process for deriving an authorization decision.

Obligations

XACML defines obligations as processing instructions for the PEP, which are included in the authorization decision. The authorization decision can request the processing of obligations either before the actual service request is made or afterwards. The processing of obligations before invoking the information service can be used to modify the original request and the processing of obligations after the PEP has fetched the service result can be used to modify the result according to access rights of the subject. More details on the use of obligations is highlighted in the outlook section (6.3, page 189).

A Tree Representation for Deriving an Authorization Decision

Based on the hierarchy of `PolicySet`, `Policy` and `Rule`, the authorization decision is driven by the outcome of the rules. This depends on the satisfaction of the rule's condition. The authorization decision is influenced by that hierarchy and the combining algorithms: A sequence of `Rule` constructs is comprised in a `Policy`. The compression of the rule outcomes is achieved by a rule combining algorithm, encoded by the `RuleCombiningAlgorithm` element. The same combining mechanism is defined for policies. The outcome from a set of `Policy` elements of one `PolicySet` is combined by the `PolicyCombiningAlgorithm`. Finally, the outcomes of a sequence of policy sets result in the authorization decision.

These hierarchy and processing rules can be described by a tree. The root node of the tree is the decision node. This node holds a combining algorithm and the available `PolicySets` of the permission repository. Each of the `PolicySet` nodes hold one `CombiningAlgorithm` node and at least one `Policy` node. In the same fashion, the `Policy` node holds one `CombiningAlgorithm` and at least one `Rule` node. Each `Rule` node holds one `Condition` node, which represents a leaf of the tree. The existence of a condition is optional. A `Rule` without a condition is an implicit declaration for an existing rule that is always satisfied, e.g. *if(True)*.

This tree structure can be defined by BNF production rules, as shown in listing 3.3.

The process of deriving an authorization decision can be separated into two steps: (i) Finding enforceable rules, policies and policy sets and (ii) calculating a final outcome of the enforceable constructs.

```

1 DecisionNode ::= CombiningAlgorithm PolicySets | CombiningAlgorithm Policies
2 PolicySets ::= PolicySet | PolicySets
3 PolicySet ::= PolicyCombiningAlgorithm Policies
4 Policies ::= Policy | Policies
5 Policy ::= RuleCombiningAlgorithm Rules
6 Rules ::= Rule | Rules
7 Rule ::= Condition

```

Listing 3.3: BNF description of the tree representation of the *Authorization Decision*

The finding of enforceable permissions depends on the matching of the Request tuple and the Target element from the PolicySet, Policy and Rule. Using the tree representation, this can be achieved by traversing the tree from the root node to the leaves, following the edges of the tree.

The calculating of an authorization decision can be achieved by processing the tree in the upward direction. Starting at the enforceable rules, the outcome on each level of the tree is compressed by the combining algorithm. One single outcome is calculated, when the root node is reached. This outcome represents the authorization decision.

For the XACML version 1.1, the combining algorithm of the root node (decision node) is restricted to **only-one-applicable**. This requires an appropriate structuring of the PolicySet, Policy and Rule constructs as described in chapter 4.

RBAC Encoding with XACML

The support for Role Based Access Control is described in [OASIS 2004d]. XACML natively supports a RBAC model if an application specific attribute is defined that has the semantic of a role. The value of that attribute define the used role. As an example, the URI `http://www.andreas-matheus.de/geoxacml/1.0/attribute#role` can be used to associate a permission to a role. The permission is associated to a role by using the SubjectMatch construct and the previous defined URI. For example, the following SubjectMatch defines a permission, associated to the role manager.

```

1 <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal" >
2   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string" >
3     employee
4   </AttributeValue>
5   <SubjectAttributeDesignator
6     AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/attribute#role"
7     DataType="http://www.w3.org/2001/XMLSchema#string" />
8 </SubjectMatch>

```

Listing 3.4: Associating a permission to a role

XACML also supports role hierarchy, as defined in RBAC₁. This can be achieved by using the PolicySetIdReference/PolicyIdReference constructs. For example, the superior role *manager* has all the permissions from the subordinated role *employee*. This can be achieved by declaring a *PolicySet* with the id *Role:Employee* defines the permissions for the role *employee*. The *PolicySet*, declaring the permissions for the manager role declares the associated

permissions and inserts the permissions of the role employee by using the reference construct. This is shown in listing 3.5.

```

1 <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy" PolicySetId="Role:Manager"
2   PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides" >
3   <Target>
4     <Subjects>
5       <Subject>
6         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal" >
7           <AttributeValue
8             DataType="http://www.w3.org/2001/XMLSchema#string" >manager</AttributeValue>
9           <SubjectAttributeDesignator
10            AttributeId="urn:www.in.tum.de/am:attribute:role"
11            DataType="http://www.w3.org/2001/XMLSchema#string" />
12         </SubjectMatch>
13       </Subject>
14     </Subjects>
15     <Resources><AnyResource/></Resources>
16     <Actions><AnyAction/></Actions>
17   </Target>
18
19   <!-- Use permissions associated with the employee role -->
20   <PolicySetIdReference>Role:Employee</PolicySetIdReference>
21 </PolicySet>
22
23 <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy" PolicySetId="Role:Employee"
24   PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides" >
25   <Target>
26     ...
27   </Target>
28   ...
29 </PolicySet>

```

Listing 3.5: Declaring permissions for a role hierarchy in XACML

3.3.4 Digital Rights Management (DRM)

Digital Rights Management (DRM) is about copyright protection. Because a digital content can be copied in equal quality and modified for own interest, the enforcement of licensed use is important. DRM is a platform to protect and enforce licensed use of a delivered (offline) digital content. In order to enforce licensed use, the content is encrypted before transmitted to the (end) user. The use of the content, such as playback is only possible with a certified software that obtains a key from a key delivery system. The delivery of the key is protected through an authorization process. It takes assertions about the requesting user (subject) and checks for sufficient rights. If the appropriate rights exist, the key is delivered and the user can use the content. Figure 3.5 illustrates the DRM processes, which can be separated into seven steps:

1. **Packaging** is the process where the digital content is encrypted and locked with a *Key*. The decryption key is stored and the information, how and where the encryption key can be requested is added to the content.
2. **Distribution** is the process, where the encrypted content is made available on a CD or for online download.

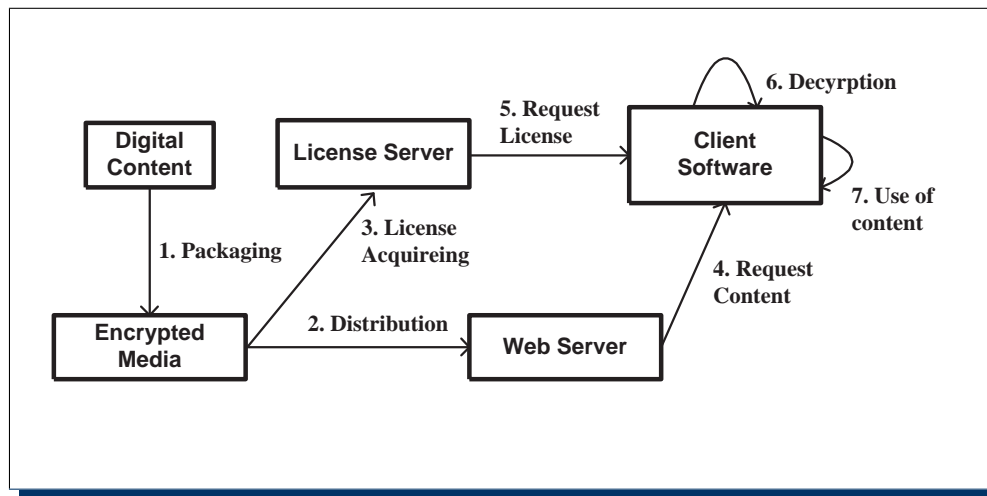


Figure 3.5: The different Digital Rights Management processes

3. **License Acquisition** is the process, where license information and the encryption key is forwarded to the license server. A user, who likes to use the content requests a license from the license server.
4. **Request Content** is the process, where the user requests the encrypted content for use. The content can be used online or offline. Before the use begins, a license must be requested. The user can initiate this or the client software does this autonomously.
5. **Request License** is the process, where the license is requested from the license server, required to use the encrypted content. The license contains the constraints of use such as how long the license is valid or how often the content can be used. Also, time constraints can be included in the license which can regulate at what times and days the content can be played. A received license can be saved to the local computer for further use.
6. **Decryption** is the process, where the client software decrypts the content and allows the licensed use. The client software contains an access control system that enforces the usage constraints, declared in the license.
7. **Content use** is the process, where the client software makes the content useable.

This general scenario is used by different companies to protect their digital audio or video files for unlicensed playback. For example, Microsoft licenses the use of Windows Media through the Windows Media Rights Manager. This scenario is shown in figure 3.6 [Microsoft 2004]. The encrypted (protected) content can only be played with the Windows Media Player and the appropriate license, including the decryption key.

The Digital Rights Management cannot be used for implementing the introduced requirements, because it is limited to playback a protected audio or video media. It does not support the declaration of fine-grained permissions, as required.

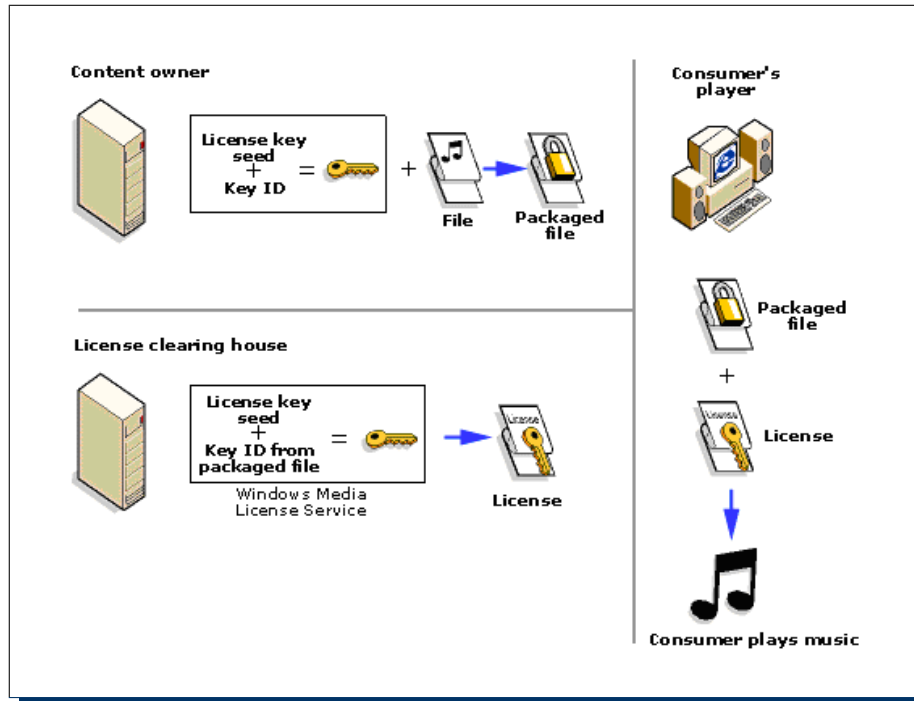


Figure 3.6: How to play an encrypted Windows Media with Microsoft DRM [Microsoft 2004]

3.3.5 EXtensible Rights Markup Language (XrML)

The eXtensible rights Markup Language (XrML) is a quasi standard from ContentGuard ([ContentGuard 2004]) for expressing rules and conditions of licensed use. It is a general-purpose language, using the XML markup to express rules and conditions. The encryption of a digital content enables DRM in the first place. XrML also defines mechanisms to ensure message integrity and entity authentication.

Even though it is not an international standard yet², Microsoft supports it by their Rights Management Services (RMS). Implementations in the Windows Media Player and Office 2003 aim to enforce licensed use of digital content such as audio and video files as well as Office documents. Windows Media Rights Manager (version 9) supports the licensed use of Windows Media content. The newest initiatives in Office 2003 are about to prevent the illegal ‘copy and paste’ usage of valuable text and graphics from Office documents. Beside Microsoft, other major companies support XrML in their products and services such as IBM, HP und VeriSign.

3.4 Systems, Implementing Distributed Access Control

Many systems exist that ‘deal’ with access control in the broader term and authorization in the narrower term. This work is interested in the existence of systems that support the decla-

²The XrML version 2.0 is frozen and submitted to the International Organization for Standardization (ISO). Based on XrML, ISO has specified the Rights Expression Language (REL).

ration and enforcement of the introduced requirements for access restrictions in a distributed environment.

3.4.1 Shibboleth

'Shibboleth is an initiative to develop an open, standards-based solution to the needs for organizations to exchange information about their users in a secure, and privacy-preserving manner.' [Internet2]. The privacy-preserving is enabled by the use of subject attributes (assertions) rather than subject identity.

Shibboleth [Erdos et. al. 2002] is a very specific system that provides web-based authentication and authorization. The main use case behind it is to secure interactions between sites: A student, using a Web Browser likes to access course material on a Web Server. The request of the Web Browser does not carry any user identity information. Therefore, an authentication system is used, where the user 'logs in'. After the request is issued, a Shibboleth component fetches subject attributes from the authentication system. Upon the fetched attributes, an authorization decision is derived. The system uses the XACML policy syntax for the declaration of the policies. An open source PDP (sunxacml.sourceforge.net) is used, which implements the XACML semantics of policy processing.

3.4.2 Cardea

Cardea [Lepro 2003] is a distributed authorization system. It separates the authentication and authorization process and establishes communication between them using SAML. Upon an authorization decision is received, the decision process starts receiving subject assertions from an Attribute Authority. In order to establish a trusted communication, the messages are encrypted using the XMLDSig standard.

Policies in Cardea are written in XACML. The authorization decision is derived from processing the policies, using the dynamically obtained subject assertions.

The system does not cover how to deal with an incomplete or inconsistent policy set. The system does not allow the declaration and enforcement of spatial restrictions and therefore does not support one major requirement for this work.

The dynamic request of subject assertions provides the basis for enabling subject privacy. On the Attribute Authority, access restrictions can control, which assertions can be requested under which context.

3.5 Conclusion of Useability and Implications for this Work

The focus of this work is to declare and enforce access restrictions for distributed geospatial information objects. As previously introduced, the environment is an infrastructure of distributed services, which provide the means of access and bring interoperability on the data structure through GML markup. For this infrastructure it must be possible to declare and enforce the introduced requirements for class-based, object-based and spatial restrictions.

Under this focus, non of the evaluated standards and systems provide such capabilities.

Therefore, the question is whether one of the standards provide a basis for the development of the required capabilities.

SAML provides the means for declaring user assertions, which is required to travel as metadata to the request to the Enforcement Service. In a similar fashion, SAML can be used to describe the decision request and response messages between the Enforcement Service and the Authorization Service. However, SAML does not provide the means for the declaration of permissions. It also does not declare the semantics how an authorization decision can be derived.

In this respect, SAML cannot be used for the declaration and enforcement of the introduced access restriction requirements.

XACL supports the declaration of access restrictions based on the elements and attributes of an XML document. But, XACL does not allow extensions. It can therefore not be used to declare the spatial restrictions.

XACML is an international standard that supports the declaration of “fine-grained” access restrictions for XML structured resources. Even the standard does not support the declaration of spatial restrictions, this capability can become available by extending the standard. This extension must define the required attributes and functions to support the declaration and enforcement of spatial restrictions.

XACML declares the semantics for processing a policy store in order to derive an authorization decision. The described process has its limits if declaring object type and instance restrictions as well as spatial restrictions.

XACML also defines message structures for the decision request and response messages, exchanged between the PEP and the PDP.

XACML is therefore the standard, which has been selected to be used as a base for developing a system that supports the declaration and enforcement of access restrictions, as introduced in section 3.1.

DRM declares processes that can be used to enforce licensed use for a static content. But the assumed environment comprises of services, which dynamically create a content, upon which access restrictions must be enforced. In that respect, DRM can only be used to enforce restrictions on the entire content.

XrML is a specification that enables the processes of DRM. It is being used to protect copyrights on a static content. As explained earlier, it is not suitable for this work.

Shibboleth and Cardea are specific implementations to given use case scenarios. Both do not fully support the introduced requirements. However, the architecture of Cardea is very similar to the intended architecture of the prototype implementation of this work. Cardea dynamically requests the subject assertions whilst deriving an authorization decision. But the prototype implementation of this work assumes that all authorization metadata is sent with the service request. Therefore, no dynamic loading of subject assertions is performed.

Chapter 4

Declaration and Enforcement of Access Restrictions

This chapter focuses on the presentation of a model for declaration and enforcement of the highlighted class-based, object-based and spatial access restrictions. This is achieved by using the XACML standard for declaration and enforcement of class- and object-based restrictions. For the declaration and enforcement of spatial restrictions, XACML is extended to GeoXACML. The presented solution assumes that the protected resources are accessible via an infrastructure of distributed services. It is also assumed that the requested resources -the resource content- is dynamically created by the access request and that its structure obeys to an object-oriented model, encoded in GML¹.

Because an error-free enforcement of access restrictions relies on the encoded permissions that declare all desired restrictions in a correct and complete way, this chapter introduces also two kinds of detections: Approximate and exact detection of inconsistent permissions. Inconsistent permissions are unreachable, incomplete or contrary permissions. The approximate detection does not make any assumptions about possible requests and the resulting resource content as the exact detection does.

4.1 Declaration of Restrictions

The class- and object-based restrictions are fundamental for an access control system to protect resources, using an object-oriented data model. As stated in the requirements, a class-based restriction protects all instances of a class, whereas the object-based restriction protects individual objects. A spatial restriction allows to enforce the restricted access to resources, based on their geometry.

In order to formalize the permissions, which encode the different restriction kinds using XACML, resp. GeoXACML language constructs, the previous definition from the Rule Based Access Control section must be refined. This is because XACML distinguishes the matching for two type of resources: Resource attributes, as they are explicitly comprised in the

¹This convention is not necessary but it simplifies the explanation of the interconnections between referencing resources in the policy and the resource content, comprised in the authorization decision request.

authorization decision request and the resource objects, defining the resource content.

4.1.1 The XACML Request Tuple

The XACML authorization decision request contains information about subjects, operations, resource attributes and resource objects, describing the resource content encoded in XML; hence GML for this work. This requires to represent the request as a four-valued tuple, differentiating between the resources, encoded as attribute-value-pairs (\mathcal{RA}) and the resource objects (resource content) (\mathcal{RO}):

$$Request := \{S, O, \mathcal{RA}, \mathcal{RO}\} \quad (4.1)$$

In the Request, S represents the identification of a subject and O represents the intended operation. Because \mathcal{RA} and \mathcal{RO} represent the protected resources, declared access restrictions must be enforced for the union of the \mathcal{RA} and \mathcal{RO} ($\mathcal{RA} \cup \mathcal{RO}$).

4.1.2 Class-Based Restrictions

The class-based restriction limits the Resource to refer to a class of the object-oriented data model. The GML equivalent is a feature type, as defined in the GML application schema. In the introduced model, the superscript ^C refers to a class-based permission and is used to define the Resource matching element (RM_R) and the Resource Content match element (XM_R^C), which holds an Xpath expression for matching GML feature types. The Condition of a rule may not be used, which is represented by the ϵ . For all other cases, representing a class-based restriction, C^C can be represented by a three-valued tuple. XF represents an Xpath node-set function, XM_C is the Xpath expression to be used with XF and \mathcal{V} defines a set of string values to be used for matching against the result of the Xpath node-set function. The condition evaluates to *True*, if the result of the Xpath node-set function matches the values of \mathcal{V} and returns *False* otherwise.

$$Rule^C := \{SM_R, OM_R, RM_R, XM_R^C, C^C\} \rightarrow \{Deny, Permit\} \quad (4.2)$$

$$C^C := \begin{cases} \epsilon & \rightarrow True \\ \{XF, XM_C, \mathcal{V}\} & \rightarrow \begin{cases} True, & XF(XM_C) \equiv \mathcal{V} \\ False, & else \end{cases} \end{cases} \quad (4.3)$$

For the services based infrastructure, the access control system must protect resources, based on GML markup. This can be achieved if the value of the XM_R^C element keeps an Xpath expression. For the class-based restriction, this expression must be limited to reference XML elements, which represent a class of the underlying object-oriented data model, a feature type in GML. The GML application schema defines a feature type as a global XML element, which is substitutable to `gml:Feature`. Therefore, a valid Xpath expression for XM_R^C can be evaluated in the following way: Let *Name* be the local name of the XML element that is fetched with the Xpath expression from the XM_R^C construct. Then, a correct reference to a GML feature can be verified, if used for matching based on the GML application schema².

²It is an implementation issue to access the correct GML application schema.

Let $\text{local-name}(\dots)$ be the Xpath node-set function that returns the local name of a node, then the following expressions verify, if XM_R^C refers to a GML feature type.

$$\text{tag-name} = \text{local-name}(\text{XM}_R^C) \quad (4.4)$$

$$\begin{aligned} \text{'element'} &\equiv \text{local-name}(/ \text{schema}/ \text{element}[\text{@name} = \text{tag-name} \\ &\quad \text{and } \text{@substitutionGroup} = \text{'gml:Feature'}]) \\ &\Rightarrow \text{XM}_R^C \text{ refers to a GML feature type} \end{aligned} \quad (4.5)$$

For example, the City Model application schema (see B.2) defines the classes **Building**, **Street** and **Intersection**. These XML elements are defined globally and substitutable for the XML Schema element `gml:Feature`. Applying the previous definitions to the example allows to validate the XM_R^C value `"/am:CityModel/gml:featureMember/am:Building"`:

$$\begin{aligned} \text{Building} &= \text{local-name}(/ \text{am:CityModel}/ \text{gml:featureMember}/ \text{am:Building}) \\ \text{'element'} &\stackrel{?}{=} \text{local-name}(/ \text{schema}/ \text{element}[\text{@name} = \text{'Building'} \\ &\quad \text{and } \text{@substitutionGroup} = \text{'gml:Feature'}]) \rightarrow \text{True} \\ &\Rightarrow \text{XM}_R^C \text{ refers to a GML feature type} \end{aligned}$$

Table 4.1 enumerates example Xpath expressions for the XM_R^C element and according verification expressions based on the City Model application schema.

Class	XM_R^C	Xpath verification expression
Building	<code>//am:Building</code>	<code>local-name(/xs:schema/xs:element[@name='Building' and @substitutionGroup='gml:Feature'])</code>
Street	<code>//am:Street</code>	<code>local-name(/xs:schema/xs:element[@name='Street' and @substitutionGroup='gml:Feature'])</code>
Intersection	<code>//am:Intersection</code>	<code>local-name(/xs:schema/xs:element[@name='Intersection' and @substitutionGroup='gml:Feature'])</code>

Table 4.1: Xpath expressions for class-based restrictions and verification expressions based on the City Model example

Using the definitions 4.2 and 4.3, example class-based restrictions can be declared. The access right that **Bob** can read objects of the class **Building**, based on the City Model example GML markup can be expressed as:

$$R^C = \{ \text{Bob}, \text{read}, \epsilon, / \text{am:CityModel}/ \text{gml:featureMember}/ \text{am:Building}, \epsilon \} \rightarrow \text{Permit}$$

For the Rule that encodes a class-based restriction, the condition C^C is not mandatory, because the matching of XM_R^C must not be refined. However, the condition must be used to encode a permission that applies to a resource content, not containing objects of a particular class. This is important in order to express complete matching as pointed out in section 2.5.10. For declaring permission according to the all-explicit strategy, it is important to allow the declaration of permissions for a resource content, not containing particular resource objects.

In this case, the regular matching -as supported by the XM_R^C - can not be used, because it associates a rule to particular existing resource objects and not to the absence of particular resource objects. Therefore, the condition C must be used to express the required matching.

For example, the following rule and condition declare that the `read` access to a resource content is unrestricted if it does not contain objects of the class `Building`. The matching of the rule `R1` provides matching for all subjects, the operation `read` and all resource content objects. The condition C refines the matching for the resource content objects in that sense that it counts the existence of the objects of class `Building`. If the count equals zero, the condition returns `True`, which triggers the rule to return `Permit`.

$$R^C = \{*, read, \epsilon, //*, C^C\} \rightarrow Permit$$

$$C^C = \{count, /am:CityModel/gml:featureMember/am:Building, \{0\}\}$$

4.1.3 Object-Based Restrictions

The object-based restriction is an extension to the class-based restriction. Whereas the class-based restriction must be enforced for all instances of the class, the object-based restriction is to be enforced for particular objects only. In this model, an object-based restriction is encoded as permissions, which use the superscript O as identification. The object-based restriction can reference one single object or a set of objects, as expressed by the Xpath. In XACML, the Resource Content matching is restricted to simple string matching, which requires the combined use of the XM_R^C and Condition matching elements. The XM_R^C references to the XML element, representing the class and the Xpath matching element of the Condition XM_C^O defines a condition to restrict the matches for particular instances. For the object-based restriction, the condition function XF can be any of the valid Xpath functions.

$$Rule^O = \{SM_R, OM_R, RM_R, XM_R^C, C^O\} \rightarrow \{Deny, Permit\} \quad (4.6)$$

$$C^O = \{XF, XM_C^O, \mathcal{V}\} \rightarrow \begin{cases} True, & XF(XM_C^O) \equiv \mathcal{V} \\ False, & else \end{cases} \quad (4.7)$$

Table 4.2 enumerates example conditional Xpath expressions, based on the City Model example.

Class	Object restriction	XM_R^C	XF	XM_C^O
Building	address="3 Street A"	//am:Building	boolean	am:address='3 Street A'
Building	address="3 Street A" or address="5 Street D"	//am:Building	boolean	am:address='3 Street A' or am:address='5 Street D'
Building	fid="HouseA"	//am:Building	boolean	./@fid='HouseA'

Table 4.2: Xpath example expressions for object-based restrictions, based on the City Model example, 2.3.2, page 28⁴

⁴The namespace for the XML document is `am` and the root element is `CityModel` as defined in the GML application schema 2.8, page 30.

Using the definitions 4.6 and 4.7, example object-based restrictions can be declared. The access right that Bob can read the object of the class `Building` at the address `5 Street D` based on the City Model example GML markup can be expressed as:

$$R^O = \{Bob, read, \epsilon, //am:CityModel/gml:featureMember/am:Building, C^O\} \rightarrow Permit$$

$$C^O = \{boolean, ./am:address="5 Street D", \{True\}\}$$

4.1.4 Spatial Restrictions

Condensing the polled spatial access control requirements from section 3.1.3, page 56 results in the capabilities of the access control model to extend the class-based restriction with a spatial condition. In this model, an object-based restriction is encoded as permissions, which use the superscript ^S as identification. The spatial condition expresses complex constraints based on a topological relation between a permission geometry and the resource object's geometries.

According to the introduced requirements for spatial access restrictions (3.1.3, page 56), permissions can be declared for resource classes satisfying a particular spatial condition. The Topological Condition Function (TCF) can be expressed by defining a Boolean expression, using the defined spatial methods `Disjoint`, `Touches`, `Crosses`, `Within`, `Overlaps`, `Intersects` and `Equals` (2.2.3, page 21) on the resource and permission geometry. Even these spatial methods can be applied to combinations of different geometry dimensions, only the case of a 2D restriction geometry was named to be practical. This results in the initial situation that a permission can be defined for a 2D surface. For this model, the geometry of a surface can be defined, using the definition 2.6, page 21 and encoded by using a GML `Polygon`.

The requirement to express a spatial permission in combination with a class resource is supported by the Rule construct (2.15, page 43). In that definition, SM_R , OM_R and RM_R define the matchmaking for the rule and the C^S defines a general expression for refining the resource content matching expressed by XM_R^C . For the declaration of spatial conditions, the *Condition* definition must be specialized in order to express the constellation between the permission geometry (G_P) and a resource geometry (G_R). This can be achieved by defining a spatial condition C^S , which is a three-valued tuple: $\{xpath\ to\ G_R\ (XM_R^C),\ spatial\ method\ (TCF),\ permission\ geometry\ (G_P)\}$. An alternative representation can be used, if the resource geometry is known: $\{TCF(G_R, G_P)\}$.

$$Rule^S := \{SM_R, OM_R, RM_R, XM_R^C, C^S\} \rightarrow \{Deny, Permit\} \quad (4.8)$$

$$C^S := \{XM_C^S, TCF, G_P\} = \{TCF(G_R, G_P)\} \quad (4.9)$$

$$TCF \in \{Disjoint, Touches, Crosses, Within, Overlaps, Intersects, Equals, \\ \neg Disjoint, \neg Touches, \neg Crosses, \neg Within, \neg Overlaps, \neg Intersects, \\ \neg Equals\} \quad (4.10)$$

$$G_R := XQuery^5(XM_C) \quad (4.11)$$

$$G_P := Polygon \quad (4.12)$$

In order to support the declaration of spatial restrictions that express a topological rela-

⁵XQuery is a function that fetches XML elements, based on the Xpath expression value, hold by XM_C^S .

tion condition in a negative way, the possible TCFs are extended to the logically negative: \neg Disjoint, \neg Touches, \neg Crosses, \neg Within, \neg Overlaps, \neg Intersects and \neg Equals.

For a rule, encoding a spatial restriction, the resource match RM_R can be used to correlate the CRS used for the encoding of the permission geometry and the CRS of the resource content object geometries.

One example for a spatial restriction that can be declared with the previous $Rule^S$ construct is that **Bob** is entitled to read objects of class **Building** if the geometry of the spatial property **shape** is **Within** the area, encoded by $G_P = \{\text{foo}, 0, 0, 10, 0, 10, 10, 0, 0\}$. Because the permission geometry is encoded by using the CRS *foo*, the matching of RM_R must reflect that.

In order to use appropriate Xpath expressions, the restriction encoded for the City Model example.

$$R^S = \{Bob, read, foo, //am:Building, C^S\} \rightarrow Permit$$

$$C^S = \{./am:shape, Within, \{\text{foo}, 0, 0, 10, 0, 10, 10, 0, 0\}\}$$

4.1.5 Complex Spatial Restrictions

A complex spatial restriction allows the declaration of permissions based on a combination of permitting or denying access rights for multiple permission geometries. As an example, access is granted if the geometry of the geospatial information object is neither within the restricted areas G_{P1} and G_{P2} .

A complex spatial restriction is declared through a Policy that combines subordinate Rule constructs, where each rule declares a spatial restriction as introduced in the previous section. In order to combine the outcome of the rules to one permission, the logical operators **AND**, resp. **OR**. can be used. Because the XACML standard -as of version 1.1- does not support the combining algorithms **AND**, resp. **OR**, they are defined by the GeoXACML extension as introduced in this work. The combining algorithms are defined by the following URIs:

$$\text{http://www.andreas-matheus.de/geoxacml/1.0/rule-combining-algorithm\#and} \quad (4.13)$$

$$\text{http://www.andreas-matheus.de/geoxacml/1.0/rule-combining-algorithm\#or} \quad (4.14)$$

The outcome of the combining algorithms **AND/OR** depends on the outcome of the subordinate Rule constructs. Because the combining algorithms do not compute on binary values (the rule's outcome can be N/A, Permit, Deny or Indeterminate), a truth table must be defined. The table 4.3 defines the combining algorithm's outcome, based on the outcome of two different rules (OR_1 and OR_2).

The semantics of these combining algorithms is explained in table 4.4. It is important to note that Deny and N/A are interpreted as False and Permit is interpreted as True.

The support for the logical combining algorithms **AND/OR** in combination with the logical **Negation** -as it is supported by the spatial condition- allows the declaration of advanced spatial restrictions. The capabilities are extended by the possibility that the permission areas can be a complex area.

Two simple examples of a complex spatial restriction is that Alice can read resource objects of class **Building** that **location** is not **Within** the restricted area G_{P1} and not **Within** the

OR1	OR2	OR1 \wedge OR2	OR1 \vee OR2
N/A	N/A	N/A	N/A
N/A	Permit	N/A	Permit
N/A	Deny	N/A	Deny
Permit	Deny	Indeterminate	Indeterminate
Permit	Permit	Permit	Permit
Deny	Deny	Deny	Deny

Table 4.3: Truth table for the logical AND/OR combination of rule outcomes

Combining algorithm	Semantics
or	This combining algorithm determines the outcome of the policy, based on the truth table 4.3. The outcome of the combining algorithm is <i>Indeterminate</i> if the processing of at least one of the subordinate rules results in an error.
and	This combining algorithm determines the outcome of the policy, based on the truth table 4.3. The outcome of the combining algorithm is <i>Indeterminate</i> if the processing of at least one of the subordinate rules results in an error.

Table 4.4: Semantics of the GeoXACML combining algorithms or and and

restricted area G_{P2} :

$$\begin{aligned}
 P1 &= \{*, *, \epsilon, //*, \text{and}, R1, R2\} \\
 R1 &= \{Alice, read, \epsilon, //Building, C1\} \rightarrow Permit \\
 C1 &= \{./location, \neg Within, G_{P1}\} \\
 R2 &= \{Alice, read, \epsilon, //Building, C2\} \rightarrow Permit \\
 C2 &= \{./location, \neg Within, G_{P2}\}
 \end{aligned}$$

A second simple example of a complex spatial restriction is that Alice can read resource objects of class `Building` that `location` is either `Within` the restricted area G_{P1} or `Within` the restricted area G_{P2} :

$$\begin{aligned}
 P1 &= \{*, *, \epsilon, //*, \text{or}, R1, R2\} \\
 R1 &= \{Alice, read, \epsilon, //Building, C1\} \rightarrow Permit \\
 C1 &= \{./location, Within, G_{P1}\} \\
 R2 &= \{Alice, read, \epsilon, //Building, C2\} \rightarrow Permit \\
 C2 &= \{./location, Within, G_{P2}\}
 \end{aligned}$$

4.1.6 Declaration of General/Exceptional Restrictions

The support of positive and negative permissions allow to declare exceptions to general permissions in a very compact way. As an example, a class-based restriction defines the general permissions that applies to all resource content objects of a class. Object-based and/or spatial restrictions can declare the exceptional permissions to an existing class-based restriction as they have the capability to make the rule apply to particular objects. In this respect, the effect of the rules that declare the general and exceptional permission is typically different. In order to distinguish this situation from a set of rules that declare contrary permissions, the combining algorithm *specific-general* may be used. Then, the situation of an exceptional permission can be declared by a sequence of rules, all subordinate to the same policy. The last rule in the sequence defines the broadest matching, which represents the general permission and all previous rules define a more specific matching, representing the exceptional restrictions.

For example, the following restriction must be declared: **Bob** can read all objects of class **Building**, but may not read the **Building** object with the address '1600 Pennsylvania Avenue NW, Washington, DC 20500'. The declaration requires one policy and two subordinate rules:

$$\begin{aligned}
 P1 &= \{*, *, \epsilon, //*, \text{specific-general}, R1, R2\} \\
 R1 &= \{Bob, r, \epsilon, C1\} \rightarrow Deny \\
 C1 &= \{//Building, ./am:address, \{"1600 Pennsylvania Avenue NW, Washington, DC 20500"\}\} \\
 R2 &= \{Bob, r, \epsilon, //Building, \epsilon\} \rightarrow Permit
 \end{aligned}$$

Without knowing that the declared rules R1 and R2 define correlated permissions for a general/specific restriction, this declaration can be misinterpreted as contrary permissions. Therefore, the combining algorithm *first-applicable* (see table 2.5, page 48) cannot be used, even it has the same processing logic.

4.2 Enforcement of Restrictions

In the infrastructure of distributed services, the resource content upon which an authorization decision must be derived is dynamically created by the service parameters of the request. This requires in general that the service request must be forwarded to the service and that the response of the service is used to represent the resource content. Depending on the format of the service output, it can directly be used for the resource content. For example, if the service output is a binary map, it cannot be used for the resource content. But, if the service output is marked-up in valid GML, it can be used as the resource content.

Under certain circumstances, the authorization decision can be based on the information of the request. This decision process is illustrated in figure 4.1. The intercepted service request contains, beside the access control metadata, information about the service operation and parameter to be used. Upon interception of the original request, the enforcement process checks if an authorization decision can or must be derived from the service operation parameters. If that is the case, the authorization decision is requested by the Authorization Service. In case that the received result is *Permit*, the original access request is forwarded to the service and the result is relayed to the requestor.

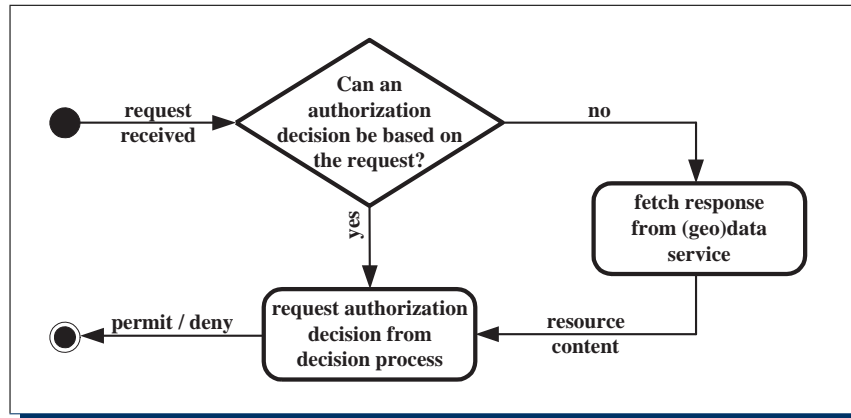


Figure 4.1: Activity diagram of the specific enforcement process

Situations, where the authorization decision must be derived from the access request exist with the Web Map Service, where no real object-oriented data model can be used to represent the service result and where the service result can be in binary format; a binary map. Another scenario, where the authorization decision must be derived from the request is for operations that result in a modification of the resource repository. If an authorization decision cannot be derived from the access request parameters, the Enforcement Service obtains the result from the (geo)data service and keeps a temporary copy. This result represents the resource content, upon which the authorization decision is derived. The resource content plus the authorization metadata is therefore used to form a decision request. After the enforcement service has received the authorization decision, the temporary copy of the service result is either forwarded to the requestor (Permit case) or deleted and an error message is sent to the requestor (Deny case).

It is quite obvious that much more processing is required if the authorization decision must be based on the service response and not on the service request parameters. Chapter 5 presents more details, how to evaluate the service parameters for the OGC Web Map Service and Web Feature Service. The next subsections illustrate requirements for the resource content, depending on the type of access restriction to be enforced.

4.2.1 Enforcement of Class-Based Restrictions

Class based restrictions refer to the structure of the resource content, marked-up in valid XML or GML. A class-based restriction is expressed by an Xpath expression, which is the value of the Resource element of a Rule construct. According to definition 2.15, page 43, a rule that expresses a class-based restriction must be included into the authorization decision, if the request subject matches the rule subject, the request operation matches the rule operation and the rule resource is part of the request resource; the resource content.

The Resource element of a Rule that expresses a class-based restriction is an Xpath expression that references an XML tag, which represents the class of an information object. For the enforcement, it is irrelevant if the resource content is comprised of one or multiple information objects of that class. If at least one object of a class is present, the matching can take place. Also, it is not compulsory that the information object is comprised of characterizing

values. The sole existence is sufficient that an existing rule can be matched.

Based on these considerations, the resource content must not necessarily be created from the actual service response. A template resource content can possibly be created from the service request parameter. The template resource content is comprised of XML tags that represent classes of information objects from the underlying object-oriented data model, according to the GML application schema definition.

For example, the listings 4.1 and 4.2 show different resource contents from the City Model example. The first resource content comprises of 'real' information objects and the other resource content of template objects.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <CityModel xmlns="http://www.in.tum.de/am" xmlns:am="http://www.in.tum.de/am"
3   xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://http://www.in.tum.de/am CityModel.xsd" fid="CityModel">
6   <gml:boundedBy><gml:Box gid="box1" srsName="foo">
7     <gml:coord><gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
8     <gml:coord><gml:X>6</gml:X><gml:Y>5</gml:Y></gml:coord>
9   </gml:Box></gml:boundedBy>
10  <gml:featureMember>
11    <Street fid="StreetA">
12      <name>Street of streets A,C</name>
13      <location srsName="foo">
14        <gml:coord><gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
15      </location>
16    </Street>
17  </gml:featureMember>
18  <gml:featureMember>
19    <Building fid="HouseA">
20      <address>3 Street A</address>
21      <shape srsName="foo">
22        <gml:outerBoundaryIs>
23          <gml:LinearRing>
24            <gml:coord><gml:X>-1</gml:X><gml:Y>2</gml:Y></gml:coord>
25            <gml:coord><gml:X>0</gml:X><gml:Y>2</gml:Y></gml:coord>
26            <gml:coord><gml:X>0</gml:X><gml:Y>3</gml:Y></gml:coord>
27            <gml:coord><gml:X>-1</gml:X><gml:Y>3</gml:Y></gml:coord>
28            <gml:coord><gml:X>-1</gml:X><gml:Y>2</gml:Y></gml:coord>
29          </gml:LinearRing>
30        </gml:outerBoundaryIs>
31      </shape>
32      <isOwnedBy xlink:role="simple" xlink:href="Alice"/>
33    </Building>
34  </gml:featureMember>
35 </CityModel>

```

Listing 4.1: Resource content example, comprising 'real' information objects

Both actual resource content and the resource template match for the same class-based restriction. Therefore, the use of a template resource content has one obvious advantage: the size of the resource content can be reduced dramatically. Using the resource template from listing 4.2 instead of the resource content from listing 4.1 reduces the size from 1179 to 434 bytes, which is a reduction of $\approx 63\%$ (non-space characters). This has two positive side effects. First, the transmission time of the authorization decision request, which is send from the Enforcement Service to the Decision Service, is much faster. Also, -assuming the same combining algorithm- the deriving of an authorization decision is faster, because the resource

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <CityModel xmlns="http://www.in.tum.de/am" xmlns:am="http://www.in.tum.de/am"
3     xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://http://www.in.tum.de/am CityModel.xsd" fid="CityModel" >
6   <gml:featureMember>
7     <Street/>
8   </gml:featureMember>
9   <gml:featureMember>
10    <Building/>
11  </gml:featureMember>
12 </CityModel>
```

Listing 4.2: Resource content example, comprising 'template' information objects

content is not comprised of redundant objects. Altogether, the reduction of the size of the resource content improves the overall processing time before an Enforcement Service receives an authorization decision.

4.2.2 Enforcement of Object-Based Restrictions

In contrast to the class-based restrictions, the enforcement of the object-based restrictions requires the resource content as it is generated by the service. This is because the object-based access restrictions relies on characteristics of object instances, such as the name or the address, as they cannot be created from the GML application schema.

4.2.3 Enforcement of Spatial Restrictions

In order to enforce spatial restrictions, the Access Control System must be able to determine, which Rule constructs applies to a given resource content. This is based on the geometry of the resource objects, comprising the resource content. The geometry encoding depends on the used Coordinate Reference System. Therefore, it must be ensured that a rule only matches those requests, where the geometry of the resource content objects is equivalent to the CRS, used to encode the permission geometry. With XACML, two options exist to control that matching: (i) Explicit encoding as an attribute value pair in the authorization decision request or (ii) based on the GML encoding of the resource object geometry.

The explicit encoding as an attribute value gives the opportunity of using the rule's resource matching element RM_R . This matching has the advantage that the applicability of the rule can be determined without evaluating the resource content. However, the Enforcement Service must know the CRS of the resource content. This is not a problem if the CRS is a request parameter. But, if the CRS is not a request parameter, it requires the processing and interpretation of the given resource content.

For all requests, where the used CRS is not encoded as an explicit attribute value, the processing of the Condition must provide the capabilities to verify the equivalence of the used CRSs.

Spatial restrictions can be encoded by using different topological relations between the permission and the resource object geometry. In order to illustrate the applicability of a

spatial restriction, different categories can be specified that are based on the dimension of resource geometry. For this access control model, the dimension of the permission geometry (G_P) is 2D ($\dim(G_P)=2$) and the dimension of the resource geometry (G_R) is either 0D, 1D or 2D ($\dim(G_R) \in \{0,1,2\}$).

Category one comprises all situations for a 0D resource geometry ($\dim(G_R)=0$). Respectively, category two comprises of all situations for the 1D resource geometry ($\dim(G_R)=1$) and category three contains all situations for the 2D resource geometry ($\dim(G_R)=2$). For each situation in each category, the topological relation between the resource and the permission geometry can be identified by using spatial methods. The spatial methods have the characteristics that they take two arguments: The first argument represents the resource geometry (G_R) and the second argument represents the permission geometry (G_P). The spatial methods return a boolean value: *True* if the topological relationship, represented by the spatial method is satisfied; *False* otherwise. A method may also return *False* if used inappropriate as stated in table 4.5.

Equals(G_R, G_P) The geometries are topological equal

Disjoint(G_R, G_P) The geometries have no point in common

Intersects(G_R, G_P) The geometries have at least one point in common

Touches(G_R, G_P) The geometries have at least one boundary point in common, but no interior points

Crosses(G_R, G_P) The geometries share some but not all interior points and the dimension of the resource geometry is 1D⁶ ($\dim(G_R) \equiv 1$).

Within(G_R, G_P) Geometry G_R lies in the interior of geometry G_P

Overlaps(G_R, G_P) The geometries share some but not all points in common and the intersection has the same dimension as the geometries themselves ($\dim(G_R) = \dim(G_P)$)

From the spatial methods (SM), a seven-valued spatial method vector \vec{sm} can be constructed

$$\vec{sm} = \left\{ \begin{array}{l} Equals(G_R, G_P) \\ Disjoint(G_R, G_P) \\ Intersects(G_R, G_P) \\ Touches(G_R, G_P) \\ Crosses(G_R, G_P) \\ Within(G_R, G_P) \\ Overlaps(G_R, G_P) \end{array} \right\}^T \quad (4.15)$$

Because the spatial methods return either *True* or *False*, an instance of the spatial method vector (\vec{sm}_I) can be interpreted as a Boolean tuple that characterizes a particular topological relationship. Because not all spatial methods can be used for all combinations of request and permission geometry dimensions, the tuple can also contain the value ϵ , expressing that this

⁶Based on the dimension of the permission geometry, which is 2D.

spatial method is inappropriate for testing. Which methods are appropriate, resp. inappropriate is stated in table 4.5.

$$\vec{sm}_I = \{sm_1, sm_2, sm_3, sm_4, sm_5, sm_6, sm_7\}, sm_i \in \{0, 1, \epsilon\}^7 \quad (4.16)$$

Method	dim(G _R)=0	dim(G _R)=1	dim(G _R)=2
Equals(G _R ,G _P)	N/A, because dim(G _P) ≠ dim(G _R)	N/A, because dim(G _P) ≠ dim(G _R)	Applicable
Disjoint(G _R ,G _P)	Applicable	Applicable	Applicable
Intersects(G _R ,G _P)	Applicable	Applicable	Applicable
Touches(G _R ,G _P)	Applicable	Applicable	Applicable
Crosses(G _R ,G _P)	N/A, because dim(G _P) ≠ dim(G _R)	Applicable	N/A, because dim(G _P)=dim(G _R)
Within(G _R ,G _P)	Applicable	Applicable	Applicable
Overlaps(G _R ,G _P)	N/A, because a 0D geometry can not overlap with a 2D geometry	N/A, because a 1D geometry can not overlap with a 2D geometry	Applicable, because two 2D geometries can overlap

Table 4.5: Which spatial methods can be used to determine a particular topological method for dim(G_P) = 2 and dim(G_R) ∈ {0,1,2}

Category 1: 0D Request Geometry, 2D Permission Geometry

All identified spatial method vectors (\vec{sm}_I) for dim(G_R)=0 are listed in this section. According to table 4.5, the usage of the spatial methods Equals, Crosses and Overlaps are inappropriate. The resulting spatial method vector has the following pattern:

$$\vec{sm}_{I,1} = \{\epsilon, sm_2, sm_3, sm_4, \epsilon, sm_6, \epsilon\} \quad (4.17)$$

The resource geometry is 0D (e.g. Point) and the permission geometry is characterized by a Polygon with zero, one or multiple holes. If the permission geometry is hole-free, the cases 1b and 1d do not apply. All possible situations and their spatial method tuples are illustrated in figures 4.2 to 4.6.

Situation 1a, 1b: In this situation, the resource geometry is outside of the permission geometry (see 4.2, 4.3). This can be detected by the spatial method vector

$$\vec{sm}_{I,1a} = \vec{sm}_{I,1b} = \{\epsilon, 1, 0, 0, \epsilon, 0, \epsilon\} \quad (4.18)$$

⁷O is equivalent to the Boolean value *False* and 1 is equivalent to the Boolean value *True*.

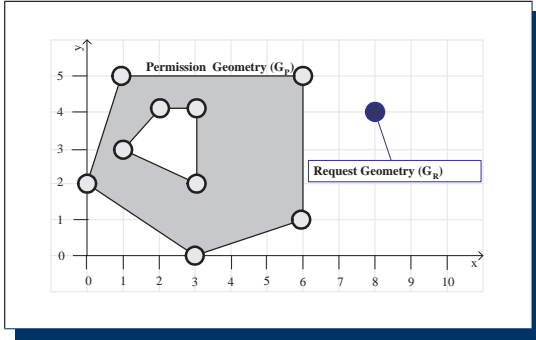


Figure 4.2: Situation 1a: G_R is outside of G_P

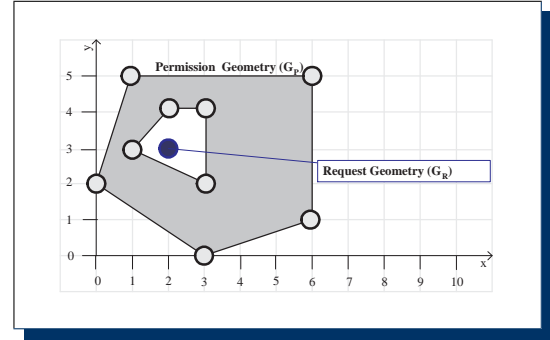


Figure 4.3: Situation 1b: G_R is inside the hole of G_P

Situation 1c, 1d: In this situation, the resource geometry is located on the inner/outer boundary of the permission geometry (see 4.4, 4.5). This can be detected by the spatial method vector

$$\vec{sm}_{I,11c} = \vec{sm}_{I,11d} = \{\epsilon, 0, 1, 1, \epsilon, 0, \epsilon\} \quad (4.19)$$

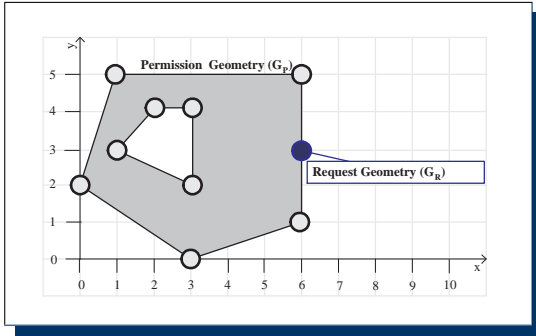


Figure 4.4: Situation 1c: G_R is on the outer boundary of G_P

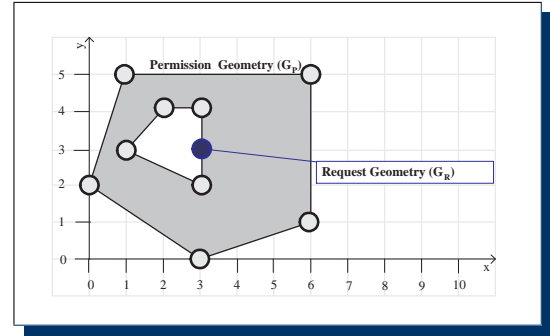


Figure 4.5: Situation 1d: G_R is on the inner boundary of G_P

Situation 1e: In this situation, the request is located in the ring of the permission geometry (see 4.6). This can be detected by the spatial method vector

$$\vec{sm}_{I,1e} = \{\epsilon, 0, 1, 0, \epsilon, 1, \epsilon\} \quad (4.20)$$

Category 2: 1D Request Geometry, 2D Permission Geometry

All identified method tuples (\vec{sm}_I) for $\dim(G_R)=1$ are listed in this section. According to table 4.5, the usage of the spatial methods, Equals and Overlaps are inappropriate. The resulting method tuple has the following pattern:

$$\vec{sm}_{I,2} = \{\epsilon, sm_2, sm_3, sm_4, sm_5, sm_6, \epsilon\} \quad (4.21)$$

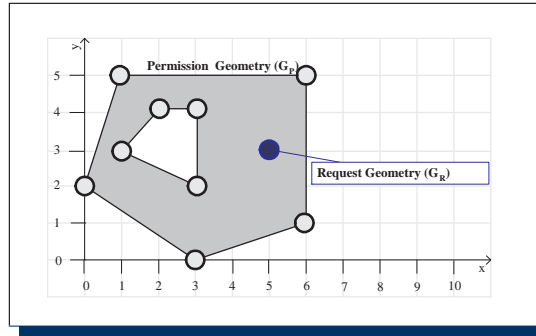


Figure 4.6: Situation 1e: G_R is inside the ring of G_P

The resource geometry is 1D (e.g. Line or LineString) and the permission geometry is characterized by a Polygon with zero, one or multiple holes. If the permission geometry is hole-free, the cases 2b, 2d, 2f, 2h, 2k, 2l and 2m do not apply. All other situations and their spatial method vectors are illustrated in figures 4.7 to 4.19.

Situation 2a, 2b: In this situation, the resource geometry is located completely outside the permission geometry or inside the hole (see 4.7, 4.8). This can be detected by the spatial method vector

$$\vec{sm}_{I,2a} = \vec{sm}_{I,2b} = \{\epsilon, 1, 0, 0, 0, 0, \epsilon\} \tag{4.22}$$

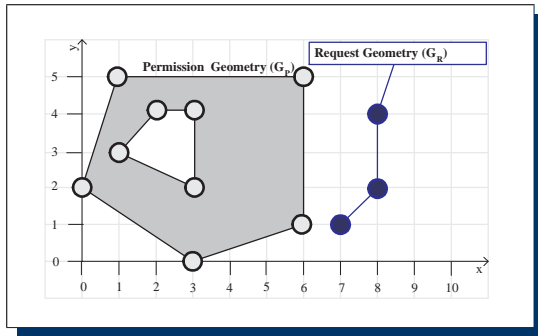


Figure 4.7: Situation 2a: G_R is outside of G_P

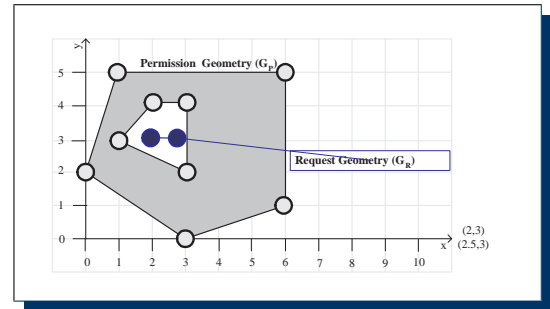


Figure 4.8: Situation 2b: G_R is inside the hole of G_P

Situation 2c, 2d: In this situation, the resource geometry is located completely outside the permission geometry or inside the hole, but is touching the permission outer/inner boundary with one start/end point of the LineString (see 4.9, 4.10). This can be detected by the spatial method tuple

$$\vec{sm}_{I,2c} = \vec{sm}_{I,2d} = \{\epsilon, 0, 1, 1, 0, 0, \epsilon\} \tag{4.23}$$

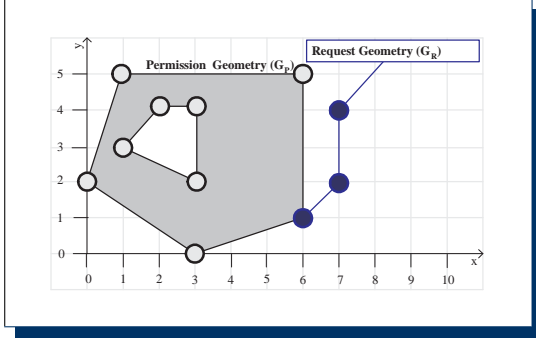


Figure 4.9: Situation 2c: G_R is outside but touching G_P

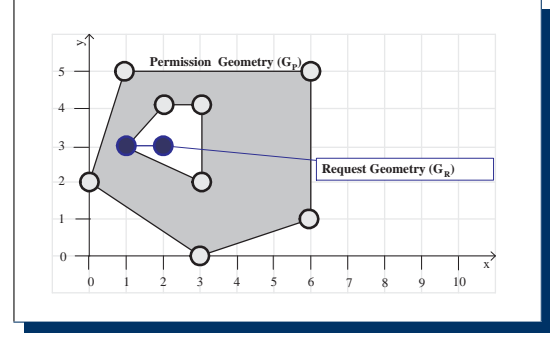


Figure 4.10: Situation 2c: G_R is inside the hole but touching G_P

Situation 2e, 2f: In this situation, the resource geometry is located on the permission outer/inner boundary (see 4.11, 4.12). This can be detected by the spatial method vector

$$\vec{sm}_{I,2e} = \vec{sm}_{I,2f} = \{\epsilon, 0, 1, 1, 0, 0, \epsilon\} \quad (4.24)$$

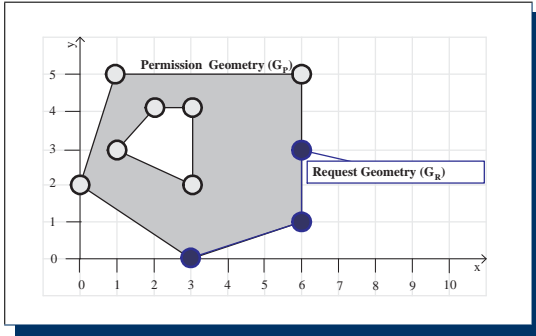


Figure 4.11: Situation 2e: G_R is on the outer boundary of G_P

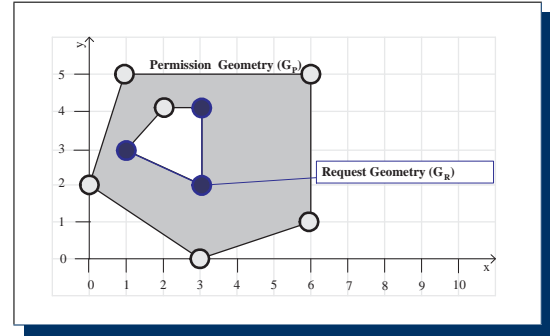


Figure 4.12: Situation 2f: G_R is on the inner boundary of G_P

Situation 2g, 2h: In this situation, the resource geometry is located in the ring of the permission geometry. However, the resource geometry and the permission boundary have one point in common (see 4.13 and 4.14). This can be detected by the spatial method vector

$$\vec{sm}_{I,2g} = \vec{sm}_{I,2h} = \{\epsilon, 0, 1, 0, 0, 1, \epsilon\} \quad (4.25)$$

Situation 2i: In this situation, the resource geometry is located in the ring of the permission geometry (see 4.15). This can be detected by the spatial method tuple

$$\vec{sm}_{I,2i} = \{\epsilon, 0, 1, 0, 0, 1, \epsilon\} \quad (4.26)$$

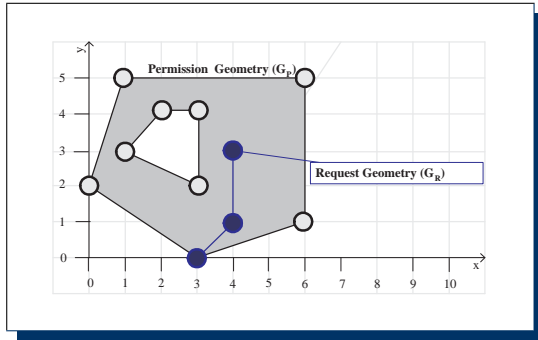


Figure 4.13: Situation 2g: G_R is inside the ring but touching G_P on the outer boundary

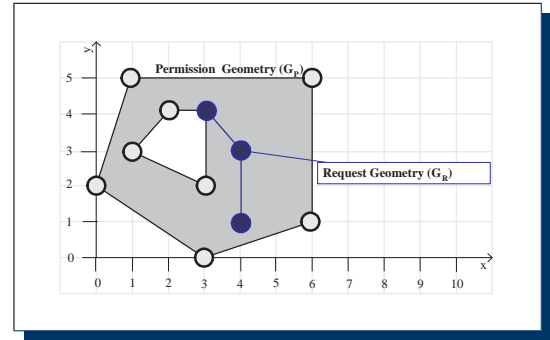


Figure 4.14: Situation 2h: G_R is inside the ring but touching G_P on the outer boundary

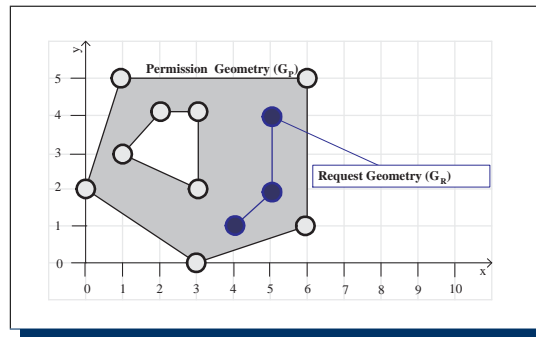


Figure 4.15: Situation 2i: G_R is inside the ring of G_P

Situation 2j, 2k: In this situation, the resource geometry is partly located in the ring of the permission geometry (see 4.16, 4.17). This can be detected by the spatial method vector

$$\vec{sm}_{I,2j} = \vec{sm}_{I,2k} = \{\epsilon, 0, 1, 0, 1, 0, \epsilon\} \tag{4.27}$$

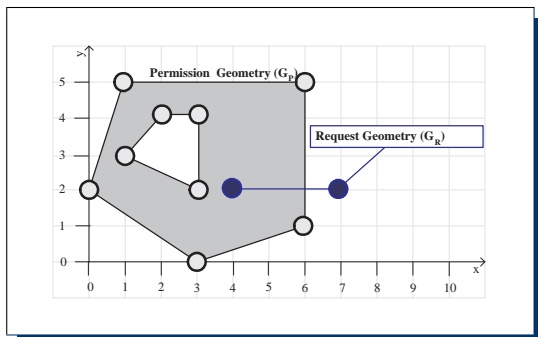


Figure 4.16: Situation 2j: G_R crosses G_P 's outer boundary

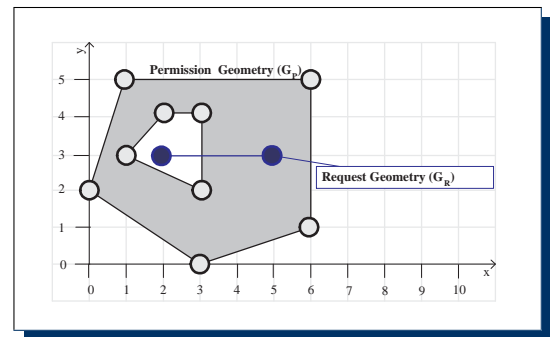


Figure 4.17: Situation 2k: G_R crosses G_P 's inner boundary

Situation 2l, 2m: In this situation, the resource geometry is crossing the permission geometry (see 4.18, 4.19). This can be detected by the spatial method tuple

$$\vec{sm}_{I,2l} = \vec{sm}_{I,2m} = \{\epsilon, 0, 1, 0, 1, 0, \epsilon\} \quad (4.28)$$

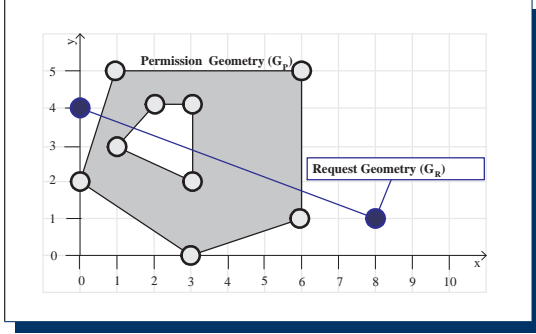


Figure 4.18: Situation 2l: G_R crosses G_P 's inner and outer boundary

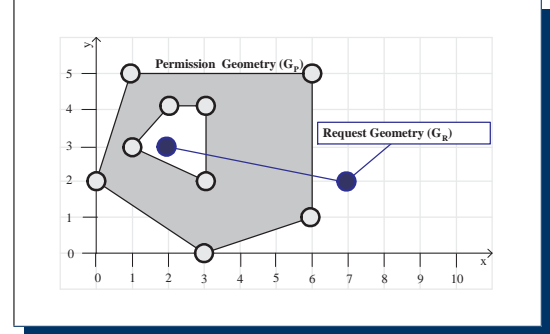


Figure 4.19: Situation 2m: G_R crosses G_P 's inner and outer boundary

Category 3: 2D Request Geometry, 2D Permission Geometry

All identified method tuples (\vec{sm}_I) are valid for $\dim(G_R)=2$. According to the table 4.5, the usage of the spatial method Crosses is not appropriate here. This results in the spatial method vector pattern

$$\vec{sm}_{I,3} = \{sm_1, sm_2, sm_3, sm_4, \epsilon, sm_6, sm_7\} \quad (4.29)$$

The resource geometry is 2D (e.g. Polygon) and the permission geometry is characterized by a Polygon with zero, one or multiple holes. If the permission geometry is hole-free, the cases 3b, 3d, 3f, 3h, 3j, 3m, 3n, 3p, 3q and 3s do not apply. For all other cases, possible methods are illustrated⁸ in figures 4.20 to 4.34.

Situation 3a, 3b: In this situation, the resource geometry is located completely outside the permission geometry or inside the hole (see 4.20, 4.21). This can be detected by the spatial method vector

$$\vec{sm}_{I,3a} = \vec{sm}_{I,3b} = \{0, 1, 0, 0, \epsilon, 0, 0\} \quad (4.30)$$

Situation 3c, 3d: In this situation, the resource geometry is located completely outside the permission geometry or inside the hole, but touching the permission outer/inner boundary with one point (see 4.22, 4.23). This can be detected by the spatial method vector

$$\vec{sm}_{I,3c} = \vec{sm}_{I,3d} = \{0, 0, 1, 1, \epsilon, 0, 0\} \quad (4.31)$$

⁸Situations 3q, 3r and 3s are not illustrated.

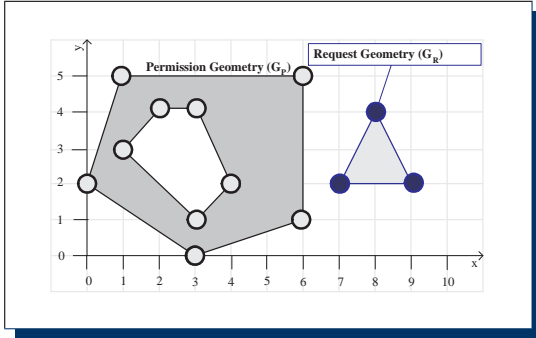


Figure 4.20: Situation 3a: G_R is outside of G_P

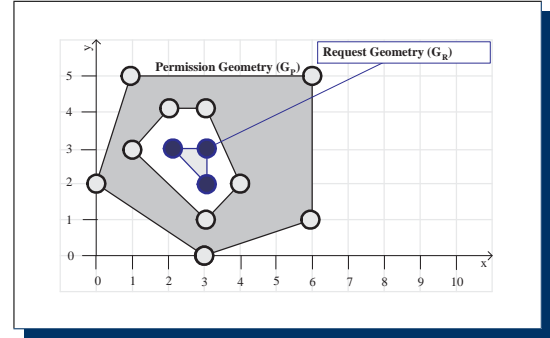


Figure 4.21: Situation 3b: G_R is inside the hole of G_P

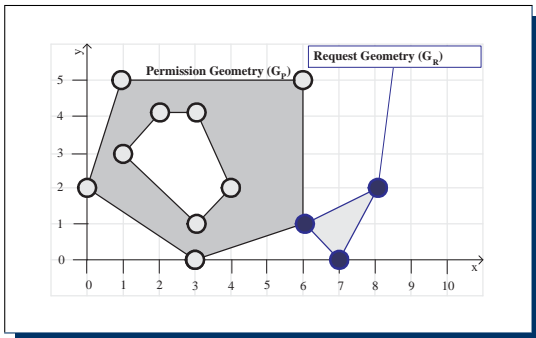


Figure 4.22: Situation 3c: G_R is outside but touching G_P

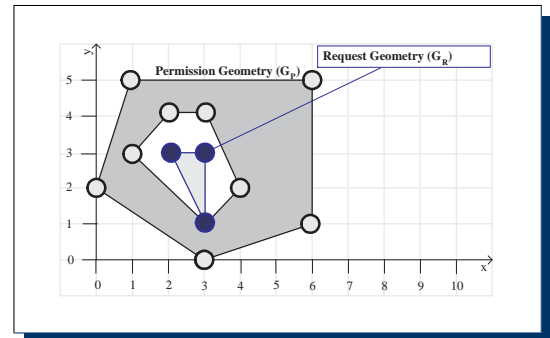


Figure 4.23: Situation 3d: G_R is inside the hole but touching G_P

Situation 3e, 3f: In this situation, the resource geometry is completely outside the permission geometry or inside the hole, but one boundary of the resource geometry is located on the outer- or inner-boundary of the permission geometry (see 4.24, 4.25). This can be detected by the spatial method tuple

$$\vec{sm}_{I,3e} = \vec{sm}_{I,3f} = \{0, 0, 1, 1, \epsilon, 0, 0\} \tag{4.32}$$

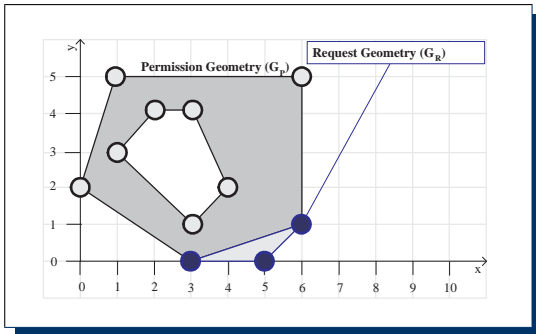


Figure 4.24: Situation 3e: G_R is outside but touching G_P

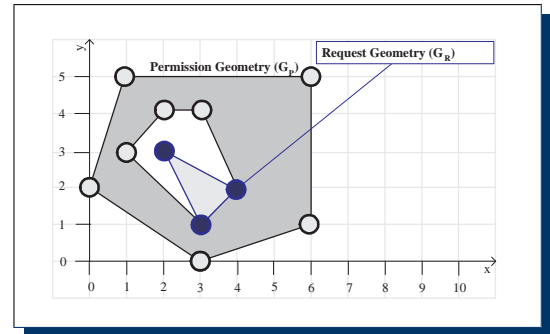


Figure 4.25: Situation 3f: G_R is inside the hole but touching G_P

Situation 3g, 3h: In this situation, the resource geometry is located in the ring of the permission geometry. However, the resource geometry and the permission boundary have one point in common (see 4.26, 4.27). This can be detected by the spatial method vector

$$\overrightarrow{sm}_{I,3g} = \overrightarrow{sm}_{I,3h} = \{0, 0, 1, 0, \epsilon, 1, 0\} \quad (4.33)$$

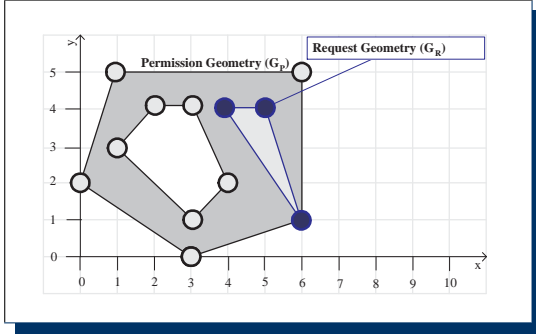


Figure 4.26: Situation 3g: G_R is inside the ring but touching G_P on the outer boundary

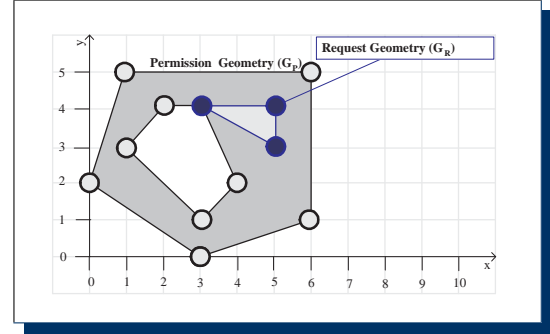


Figure 4.27: Situation 3g: G_R is inside the ring but touching G_P on the inner boundary

Situation 3i, 3j: In this situation, the resource geometry is located in the ring of the permission geometry. However, the resource geometry and the permission boundary have one line segment in common (see 4.28, 4.29). This can be detected by the spatial method vector

$$\overrightarrow{sm}_{I,3i} = \overrightarrow{sm}_{I,3j} = \{0, 0, 1, 0, \epsilon, 1, 0\} \quad (4.34)$$

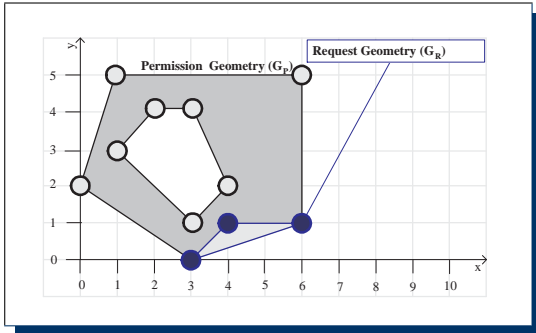


Figure 4.28: Situation 3i: G_R is inside the ring but touching G_P on the outer boundary

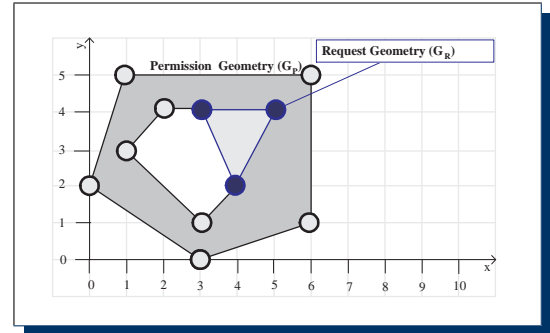
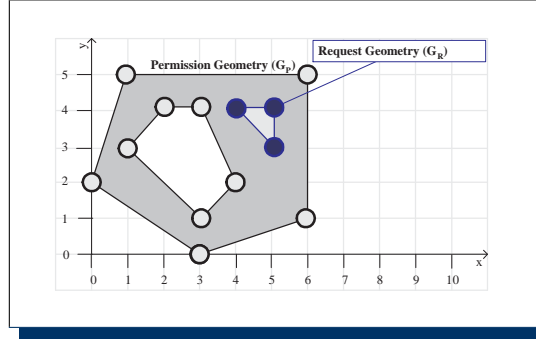


Figure 4.29: Situation 3j: G_R is inside the ring but touching G_P on the inner boundary

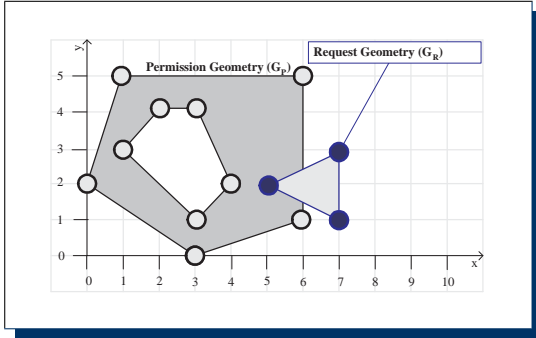
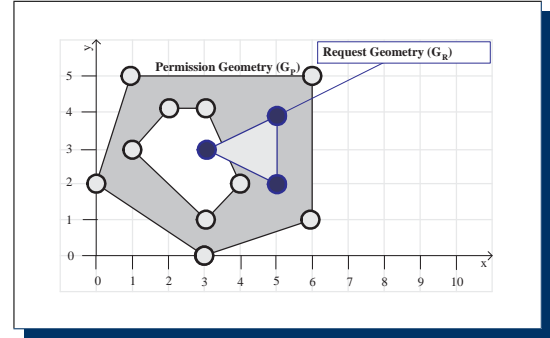
Situation 3k: In this situation, the resource geometry is located completely in the ring of the permission geometry (see 4.30). This can be detected by the spatial method tuple

$$\overrightarrow{sm}_{I,3k} = \{0, 0, 1, 0, \epsilon, 1, 0\} \quad (4.35)$$

Figure 4.30: Situation 3k: G_R is inside the ring of G_P

Situation 3l, 3m: In this situation, the resource geometry is overlapping the permission geometry (see 4.31, 4.32). This can be detected by the spatial method tuple

$$\vec{sm}_{I,3l} = \vec{sm}_{I,3m} = \{0, 0, 1, 0, \epsilon, 0, 1\} \quad (4.36)$$

Figure 4.31: Situation 3l: G_R is overlapping G_P on the outer boundaryFigure 4.32: Situation 3m: G_R is overlapping G_P on the inner boundary

Situation 3n, 3o: In this situation, the resource geometry is overlapping the permission geometry (see 4.33, 4.34). This can be detected by the spatial method tuple

$$\vec{sm}_{I,3n} = \vec{sm}_{I,3o} = \{0, 0, 1, 0, \epsilon, 1, 0\} \quad (4.37)$$

Situation 3p: In this situation, the resource geometry is equal to the permission geometry. This can be detected by the spatial method tuple

$$\vec{sm}_{I,3p} = \{1, 0, 1, 0, \epsilon, 1, 0\} \quad (4.38)$$

Situation 3q: In this situation, the resource geometry is equal to the outer boundary of the permission geometry; the resource geometry does not have the hole. This can be detected by the spatial method tuple

$$\vec{sm}_{I,3q} = \{0, 0, 1, 0, \epsilon, 0, 0\} \quad (4.39)$$

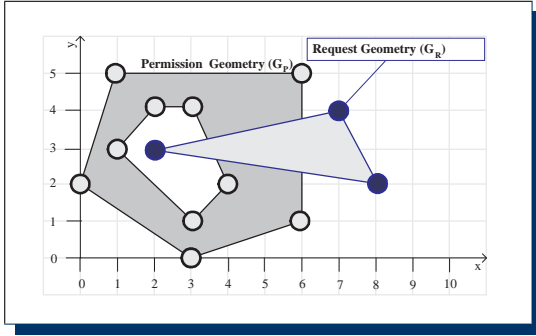


Figure 4.33: Situation 3n: G_R is overlapping G_P

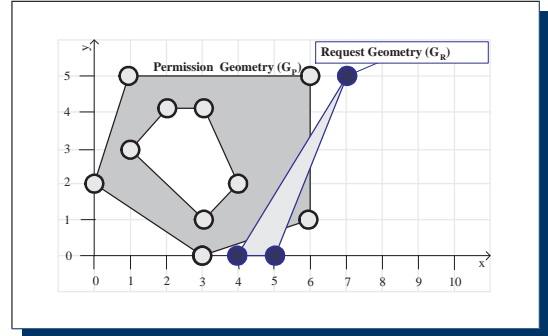


Figure 4.34: Situation 3o: G_R is overlapping G_P

Situation 3r: In this situation, the resource geometry is equal to the inner boundary of the permission geometry; the request geometry is equal to the hole. This can be detected by the spatial method tuple

$$\vec{sm}_{I,3s} = \{0, 0, 1, 1, \epsilon, 0, 0\} \quad (4.40)$$

4.3 Visualization of Access Restrictions

The visualization of access restrictions is an important support for the policy writer to understand the declared permissions. The different kinds of permissions can be visualized in different ways. The class- and object-based restrictions can be visualized as a tree (see 3.3, page 67). In addition, the spatial permissions can be visualized as a map, expressing their geospatial nature.

4.3.1 Visualization of the Permission Hierarchy

The declaration of permissions is based on the hierarchy as defined by XACML. The hierarchy can contain one or more different PolicySet levels, one Policy level, one Rule level and one Condition level. For each permission, a specific matching for requests determines its applicability. The combining algorithm of a PolicySet or Policy construct defines if one or all subordinate constructs must be evaluated, in order to derive an authorization decision.

The use of a combining algorithm that uses the effect of the first matching Rule -hence one rule- are first-applicable, only-one-applicable and specific-general. This processing behavior can be visualized as an OR-tree: The PolicySet or Policy construct is the root node, from which edges go to the subordinate constructs.

In contrast to the previously stated combining algorithms, the following combining algorithms process each subordinate permission, in order to derive an authorization decision: deny-overrides, permit-overrides, ordered-deny-overrides, ordered-permit-overrides, and and or. This processing behavior can be visualized as an AND-tree: The PolicySet or Policy construct is the root node, from which edges go to the subordinate constructs.

In order to illustrate the visualization, let's use the following example permission tree that

defines on PolicySet, two Policy constructs, using different combining algorithms. Each Policy has subordinate rules with spatial conditions:

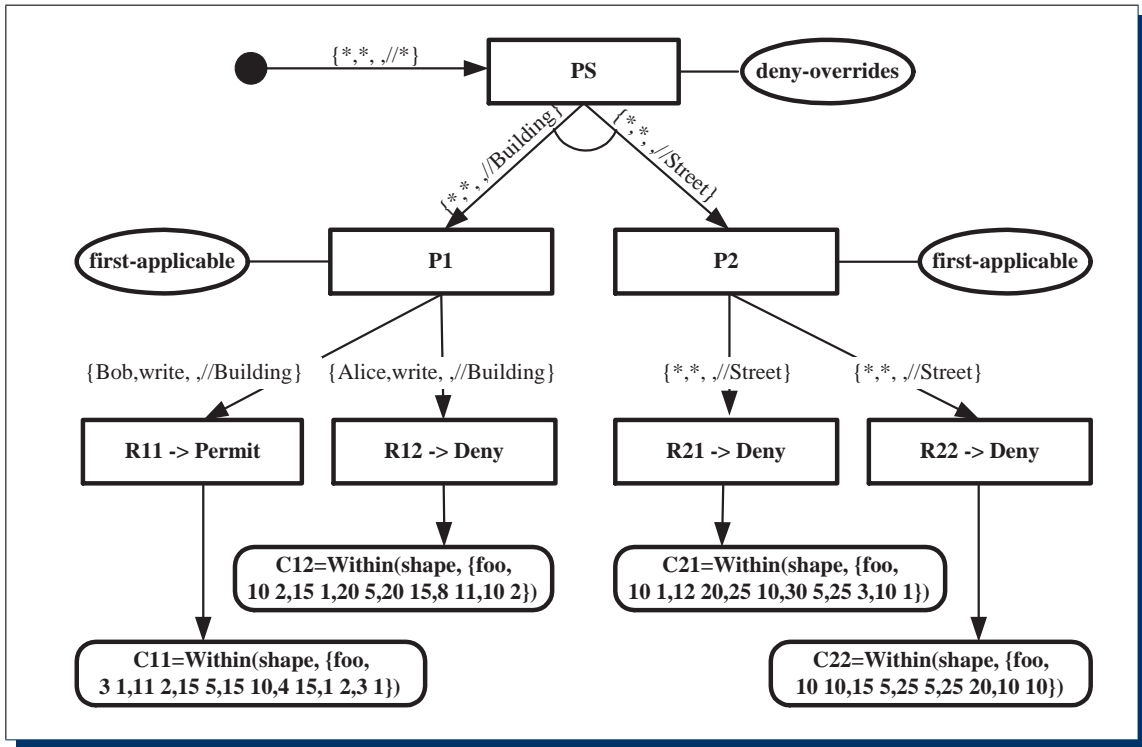
$$\begin{aligned}
 PS &= \{*, *, \epsilon, //*, \text{deny-overrides}, P1, P2\} \\
 P1 &= \{*, *, \epsilon, //Building, \text{first-applicable}, R11, R12\} \\
 P2 &= \{*, *, \epsilon, //Street, \text{first-applicable}, R21, R22\} \\
 R11 &= \{Bob, write, \epsilon, //Building, C11\} \rightarrow \text{Permit} \\
 C11 &= \{./am:shape, Within, \{foo, 3 1,11 2,15 5,15 10,4 15,1 2,3 1\}\} \\
 R12 &= \{Alice, write, \epsilon, //Building, C12\} \rightarrow \text{Deny} \\
 C12 &= \{./am:shape, Within, \{foo, 10 2,15 1,20 5,20 15,8 11,10 2\}\} \\
 R21 &= \{*, *, \epsilon, //Street, C21\} \rightarrow \text{Deny} \\
 C21 &= \{./am:shape, Within, \{foo, 10 1,12 20,25 10,30 5,25 3,10 1\}\} \\
 R22 &= \{*, *, \epsilon, //Street, C22\} \rightarrow \text{Deny} \\
 C22 &= \{./am:shape, Within, \{foo, 10 10,15 5,25 5,25 20,10 10\}\}
 \end{aligned}$$


Figure 4.35: Visualization of an example permission tree

Figure 4.35 shows an AND-tree that starts at the top level PolicySet. This is because the PolicySet PS uses the combining algorithm deny-overrides. Both subordinate Policy constructs use the combining algorithm first-applicable. The visualization of their subordinate rules can take place as OR-trees.

In respect, helping the policy writer to understand the declared permissions, such a tree representation can be used to illustrate the different kinds of errors as they are introduced

in this work: unreachable, incorrect or incomplete permissions. In order to do so, different coloring can be used to illustrate the different kinds of errors. Even different classifications of errors are possible by simply changing the color of edges and nodes. Examples for illustrating different kinds of permission errors are provided in later sections.

A tree visualization can not only be used to show existing permissions, but also be used to edit, create or delete permissions. In order to do so, the supporting application must have the capabilities to create native XACML constructs from the tree representation.

4.3.2 Visualization of Spatial Restrictions

Spatial access restrictions are naturally suitable to be represented as a map. The spatial conditions can be rendered according to the permission geometry, using different line styles for representing the different spatial relations (2.2.3, page 21). The matching criteria of the spatial condition and the effect of the superordinate rule can be printed on the outer borderline of the permission geometry.

Figure 4.36 shows the different line styles for the defined spatial relations. The spatial relation *Within* is represented by a line with vertical fringes to the inside; *Touches* is represented by a line with filled points; *Crosses* is represented by a line with vertical fringes that cross the line; *Equals* is represented by a thin line on a brighter bold line; *Overlaps* is represented by a line with crossing and solid diamonds; *Disjoint* is represented by a line with filled diamonds on the outside of the line; *Intersects* is represented by a zig-zag line. The rendering of the outer boundary and inner boundary/boundaries of a complex restriction area (*Surface* with holes) for the spatial relation *Within*, the figs must show in the appropriate direction: The figs of the inner boundaries must point to the outer boundary and the figs of the outer boundary must point toward the inner boundary/boundaries.

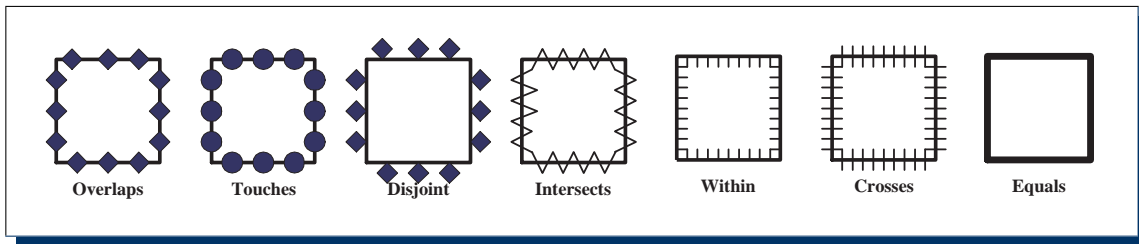


Figure 4.36: Line styles for rendering spatial restricted areas

The encoding of spatial permissions that use the negation of a defined spatial relation can be visualized, using the line styles as illustrated in figure 4.37. The spatial relation \neg Within is represented by a line with vertical negation symbols⁹ to the inside; \neg Touches is represented by a line with unfilled points; \neg Crosses is represented by a line with the logical negation symbols (\neg) crossing the line; \neg Equals is represented by a line that is comprised of negation (\neg) symbols; \neg Overlaps is represented by a line with crossing and unfilled diamonds; \neg Disjoint is represented by a line with unfilled diamonds on the outside of the line; \neg Intersects is represented by a zig-zag line on the outside of the borderline. The rendering of the outer boundary and inner boundary/boundaries of a complex restriction area (*Surface* with holes)

⁹In order to avoid confusion with the semantics of outside, this visualization is used.

for *Within* and \neg *Within*, the figs must show in the appropriate direction: The figs of the inner boundaries must point to the outside and the negation symbols of the outer boundary must point toward the inside.

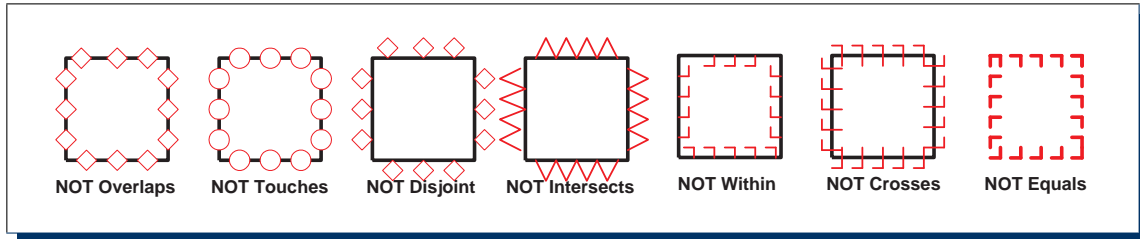


Figure 4.37: Line styles for rendering negative spatial restricted areas

Using these visualization conventions, the following spatial example restrictions can be displayed as a map, as shown in figure 4.38 and 4.39.

$$R1 = \{Bob, read, foo, //am:Building, C1\} \rightarrow Permit$$

$$C1 = \{./am:shape, Within, \{foo, 0\ 0,9\ 0,9\ 4,0\ 4,0\ 0\}\}$$

$$R2 = \{Bob, read, foo, //am:Building, C2\} \rightarrow Permit$$

$$C2 = \{./am:shape, \neg Within, \{foo, 0\ 0,9\ 0,9\ 4,0\ 4,0\ 0\}\}$$

For condition C1, the permission geometry is represented as a simple surface where the borderline is rendered by the line with vertical fringes to the inside, representing the spatial relation *Within* (figure 4.38). For condition C2, the permission geometry is represented as a simple surface where the boundary line is rendered by the line with vertical negation symbols to the inside, representing the spatial relation \neg *Within* (figure 4.39). For both permissions, the matching conditions of the associated rule, the outcome and the resource object matching is printed on the outer borderline.

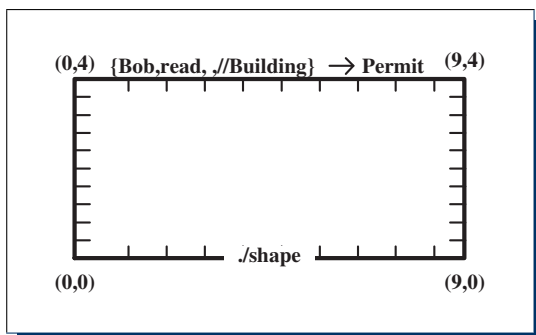


Figure 4.38: Visual representation of the spatial example permission R1 and C1, using the spatial relation *Within*

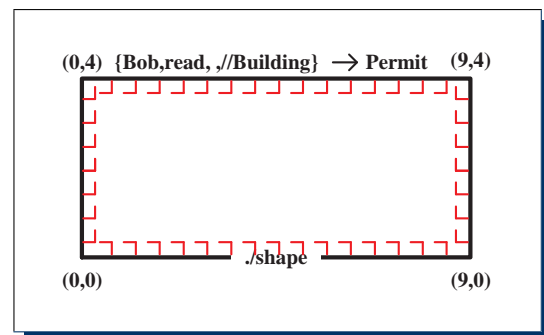


Figure 4.39: Visual representation of the spatial example permission R2 and C2, using the spatial relation \neg *Within*

4.3.3 Visualization of Combined Spatial Restrictions

The need for the visualization of combined spatial restrictions come from different spatial permissions, being comprised in a permission tree. As stated earlier, a permission tree can have OR- and/or AND-branches, depending on the used combining algorithms in a PolicySet or Policy.

The combined visualization of all spatial permissions of a permission tree can be achieved by visualizing the individual spatial restrictions as illustrated before. In order to reflect the different processing of AND- and OR-branches of a permission tree, a layered map can be used.

- The spatial conditions of an AND-branch potentially apply to the same request. The corresponding visual representations of the spatial permissions are therefore drawn on the same layer.
- The spatial conditions of an OR-branch do not all have effect on the deriving of an authorization decision. Just one of the rules is being used. Therefore, the spatial conditions of an OR-branch can be drawn in a multi-layer map. Each layer contains the spatial condition of one OR-branch.

In order to illustrate the visualization for an AND-branch, the following example shall be used:

$$\begin{aligned}
 P1 &= \{\dots, \dots, \dots, \dots, \text{deny-overrides}, R11, R12, R13\} \\
 R11 &= \{\dots, \dots, \dots, \dots, C11\} \rightarrow \text{Permit} \\
 C11 &= \{\dots, \text{Within}, \{\text{foo}, 3\ 1,11\ 2,15\ 5,15\ 10,4\ 15,1\ 2,3\ 1\}\} \\
 R12 &= \{\dots, \dots, \dots, \dots, C12\} \rightarrow \text{Permit} \\
 C12 &= \{\dots, \text{Within}, \{\text{foo}, 10\ 2,15\ 1,20\ 5,20\ 15,8\ 11,10\ 2\}\} \\
 R13 &= \{\dots, \dots, \dots, \dots, C13\} \rightarrow \text{Permit} \\
 C13 &= \{\dots, \text{Within}, \{\text{foo}, 12\ 8,17\ 17,7\ 20,12\ 8\}\}
 \end{aligned}$$

In figure 4.40, the permission hierarchy for the previous permission constructs is shown¹⁰. Figure 4.41 shows the visualization of the spatial conditions, declared in C11, C12 and C13¹¹. Because the spatial conditions belong to an AND-tree, all permission geometries are drawn in the same layer (layer 1).

In order to illustrate the visualization of an OR-branch, the following example shall be

¹⁰The matching conditions are omitted for simplicity.

¹¹The use of the introduced visualization is omitted for readability.

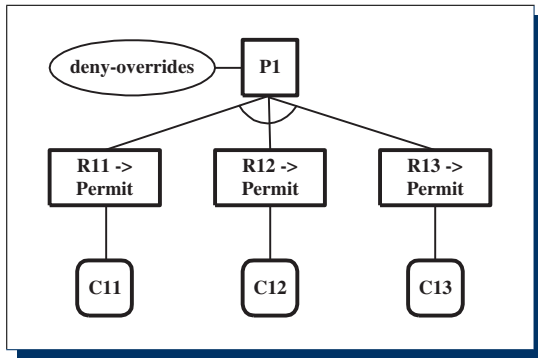


Figure 4.40: Illustrating a permission-tree for a Policy construct that uses the combining algorithm deny-overrides

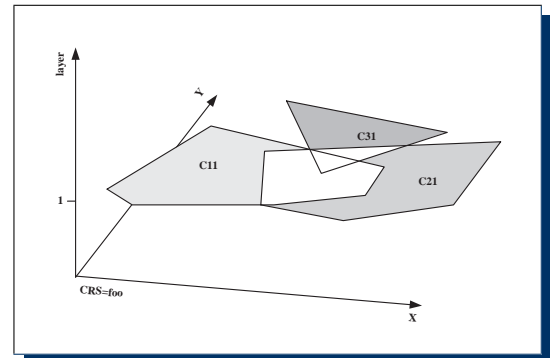


Figure 4.41: Visualization of the spatial conditions of the example AND-graph from the figure to the left

used:

$$\begin{aligned}
 P2 &= \{ \dots, \dots, \dots, \text{first-applicable}, R21, R22, R23 \} \\
 R21 &= \{ \dots, \dots, \dots, C21 \} \rightarrow \text{Permit} \\
 C21 &= \{ \dots, \text{Within}, \{ \text{foo}, 3, 1, 11, 2, 15, 5, 15, 10, 4, 15, 1, 2, 3, 1 \} \} \\
 R22 &= \{ \dots, \dots, \dots, C22 \} \rightarrow \text{Permit} \\
 C22 &= \{ \dots, \text{Within}, \{ \text{foo}, 10, 2, 15, 1, 20, 5, 20, 15, 8, 11, 10, 2 \} \} \\
 R23 &= \{ \dots, \dots, \dots, C23 \} \rightarrow \text{Permit} \\
 C23 &= \{ \dots, \text{Within}, \{ \text{foo}, 12, 8, 17, 17, 7, 20, 12, 8 \} \}
 \end{aligned}$$

In figure 4.42, the permission hierarchy for the following permission constructs is shown¹². Figure 4.43 shows the visualization of the spatial conditions, declared in C21, C22 and C23¹³. Because the spatial conditions belong to an OR-tree, each permission geometry is drawn on a separate layer (layers 1 to 3).

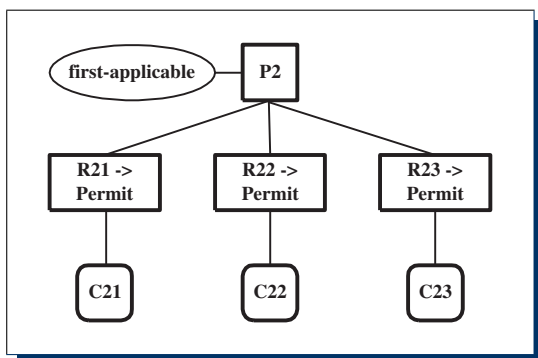


Figure 4.42: Illustrating a permission-tree for a Policy construct that uses the combining algorithm first-applicable

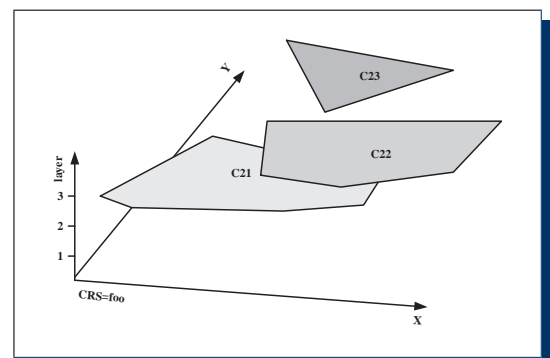


Figure 4.43: Visualization of the spatial conditions of the example OR-graph from the figure to the left

¹²The matching conditions are omitted for simplicity.

¹³The use of the introduced visualization is omitted for readability.

According to these conventions, the visualization of all spatial restrictions of a permission tree requires in general a multi-layered map. The number of required layers and which spatial conditions must be drawn together on the same layer, can be determined from the permission tree that potentially can contain one or multiple AND- and/or OR-subtrees: The permission tree can be represented as an equation, using the multiply (\bullet) and plus (+) symbols. Each AND-branch of two or more permissions can be presented as a multiplication of the branches. Each OR-branch of two or more permissions can be presented as an addition.

For example, a PolicySet PS is the root node and defines an AND-branch to the direct subordinate policies P1 and P2, then PS can be represented as $PS = P1 \bullet P2$. In the same manner, an OR-branch can be represented as an addition of the direct subordinate permissions. For example, the Policy P1 that has two direct subordinate rules R11 and R12, which can be represented as $P1 = R11 + R12$. After creating all equations, a recursive replacement can take place, starting at the root node of the permission tree and ending at the leaves that represent a spatial condition. In order to know the number of required layers, the final equation of the tree must be in a form, similar to the disjunctive form: The permission tree can be represented by the form $Term_1 + Term_2 + \dots + Term_n$. The number n represents the required number of layers. Each term consists of a spatial conditions that must be drawn on the same layer. For example, $Term_1 = C11 \bullet C12$ requires to draw the spatial restrictions C11 and C12 on the same layer: Layer 1 for the example.

In order to illustrate that approach, lets use the permission tree from figure 4.35. The root of the permission tree is PS. It is also the root node of an AND-branch with direct subordinate policies P1 and P2. This can be represented as $PS = P1 \bullet P2$. The policy constructs P1 and P2 define an OR-tree. This results in the following representations: $P1 = R11 + R12$ and $P2 = R21 + R22$. Because each rule has just one spatial condition, P1 and P2 can be represented as $P1 = C11 + C12$ and $P2 = C21 + C22$ by substituting the rules with the spatial conditions. Substituting the equations P1 and P2 into the equation for PS results in the following equation: $PS = (C11 + C12) \bullet (C21 + C22)$. Multiplying out the parenthesis results in the desired form: $PS = C11 \bullet C21 + C11 \bullet C22 + C12 \bullet C21 + C12 \bullet C22$. As the result denotes, the visualization requires four different layers: Layer 1 keeps the spatial conditions C11 and C21, layer 2 keeps C11 and C22, layer 3 keeps C12 and C21 and layer 4 keeps C12 and C22. The resulting graphical representation of this permission tree is shown in figure 4.44.

Even the approach is illustrated for just one example, it is possible to transform any permission tree into the required form by applying basic mathematics. For example, exchanging the combining algorithms PS and P1 would result in the following equations¹⁴: $PS = P1 + P2$, $P1 = R11 \bullet R12$, $P2 = R21 + R22$. Substituting P1 and P2 -and the spatial conditions- results in the equation $PS = C11 \bullet C12 + C21 + C22$. The corresponding visualization requires 3 layers: Layer 1 contains C11 and C12, layer 2 contains C21 and layer 3 contains C22.

4.3.4 Remarks to the Visualization

The introduced model does not allow the definition of complex rule conditions, using more than one permission geometry for encoding a spatial restriction and more than one spatial relation. However, the model supports the use of complex surfaces, which already represents

¹⁴The combining algorithm of PS is first-applicable and of P1 is deny-overrides now.

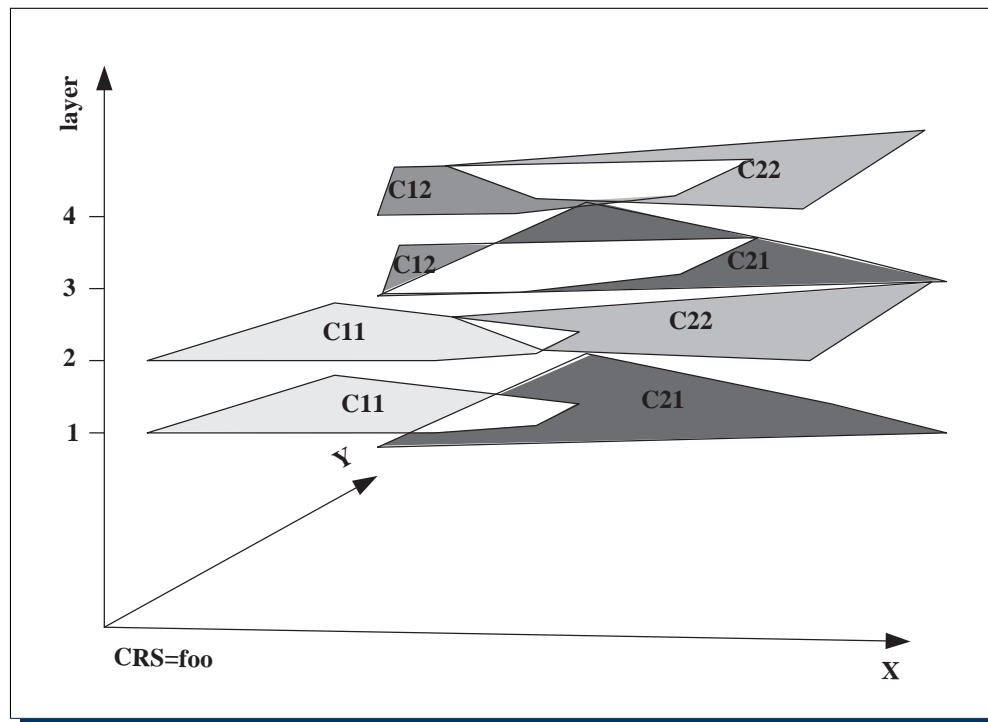


Figure 4.44: Visualization of spatial permissions, as a multi-layered map

a complex spatial condition. Due to this limitation, it is not possible that additional AND-and/or OR-trees result from the spatial conditions itself.

The visualization of the spatial restrictions does not consider the declared matching. The introduced visualization supports the policy writer to realize, which permissions are associated to which areas. Because the map of all spatial permissions of a permission tree can become quite complex and very confusing, a visualization tool may support the on/off-switching of layers. In such a way, the policy writer can select the branches of interest. In addition, the visualization may support the selection of specific areas of interest. The combination of selecting the area of interest and the layers of concern can improve the significance of the visualization.

4.4 Approximate Detection of Inconsistent Permissions

The approximation of inconsistent access permissions does not require the knowledge about existing subjects, operations and resources. The approximation evaluates the used matching of policy sets, policies and rules. The approximation results in a classification that indicates potential inconsistencies. The information can be used to annotate the visualization in order to direct the policy writer to the potential cause. The simplest cause of inconsistent permissions is the misspelling of matching expressions, the use of wrong combining algorithms or erroneous structuring of permission encodings. However, the information is not sufficient to instruct the policy writer how to fix a problem. The exact detection of inconsistent permissions, as illustrated in sections 4.5.1, 4.5.2 and 4.5.3 is based on the knowledge of possible

subjects, existing operations and available resource objects. Only this allows to determine concrete information for a policy writer to correct any existing problems.

Different inconsistencies can exist in a set of declared access permissions: unreachable, incomplete and incorrect permissions.

Unreachable permissions express the constellation that a superordinate permission declares a matching in such a way that the subordinated permission is never reached. This means that for a given permission tree not all nodes can be reached. The unreachable permissions do not have effect on the authorization decision and are therefore unnecessary. Their existence simply make the permission repository unnecessarily large and complicated. However, they can influence the deriving of an authorization decision if the permission structure or the matching changes.

Incomplete permissions express the constellation that not all possible requests are matched. For the all-explicit declaration strategy, the existence of incomplete permissions represent the error case, where an authorization decision results in N/A. For the all-deny or all-permit declaration strategy, the existence of incomplete permissions result in a wrong enforcement, because the N/A authorization decision is interpreted as additional (not declared) accept or block of the request. For an error-free enforcement, it is therefore essential to detect their existence.

Contrary permissions express the constellation that at least two permissions match for the same request, but with different effects: One declares the effect Deny and the other Permit. It is important to detect the incorrect permissions, because their existence bare potential error, which can not be accepted. For the introduced types of access restrictions, two different constellations can exist: At least two permissions match the same subject, operation and resources and their effect is different. This constellation is to be corrected.

The other constellation expresses a general permission and a specific permission. This constellation can be detected if one permission matches a subset of the resources from the other permission and the combining algorithm `specific-general` is being used. This constellation must not be corrected, because it is intended by the policy writer.

The introduced model for expressing class-based, object-based and spatial restrictions uses XACML for the encoding and processing. This allows the subject, operation and resource matching based on attribute value pairs. For the resources, an additional matching based on Xpath expressions is possible for the resource content. XACML defines a `Target` element for the permission constructs `PolicySet`, `Policy` and `Rule`. The `Target` element holds the `Subject`, `Operation` and `Resource` elements, which provide the matching expressions. In addition, the `Rule` can contain a `Condition` element that defines additional matching constraints for the subject, operation, resource or even environment attributes. This section restricts the capabilities of the `Condition` to restrict the matching effective for the resource content. This assumption still allows the declaration of object-based and spatial restrictions, because the condition is used to restrict the applicability of the rule based on object's spatial and non-spatial characteristics.

For the approximation of an inconsistent constellation, it is essential to declare the essential matching condition. This is based on the matching expression of the `Target` element. Without

a request, the matching criteria for subject, operation and resource of the **Target** elements from different permissions are evaluated against each other. If the essential condition is satisfied by two different permissions, the **Condition** must be evaluated in order to fetch additional information.

This chapter provides a systematic approximation for the class-based, object-based and spatial restrictions. But before, the matching for resource objects based on the Xpath expression is introduced.

4.4.1 The Essential Test Conditions

As mentioned earlier, the essential condition for inconsistent permissions is that at least two different permissions are applicable to the same request. This can be determined upon the matching conditions of the **Subject**, **Operation**, **Resource** and **Resource Content** elements. Because no information about the possible requests (subjects, operations, resource attributes and resource objects) is available, the test must evaluate the matching conditions of the declared permissions.

Sub/Superordinate Relation between Permissions

For the detection of inconsistent permissions, the relationship between permissions is important. In particular, which policies are comprised in a policy set and which rules are comprised in a policy. This can be defined, using the introduced tree representation (3.3, page 67) in the following way: A **PolicySet** is superordinate to a **Policy** if a path exists from the **PolicySet** to the **Policy** that has the distance one (1). In the same manner, a **Policy** is superordinate to a **rule** if a path exist with the distance one (1). This super/subordinate relationship between a **Rule** and a **Policy**, resp. a **Policy** and a **PolicySet** can be defined by the subordinate operator (\angle) :

$$\text{Rule} \angle \text{Policy} \angle \text{PolicySet} \quad (4.41)$$

Because the \angle operator is transitive, a **Rule** is also subordinate to a **PolicySet**. Then, the distance from the **PolicySet** to the **Rule** is two (2).

$$\text{Rule} \angle \text{PolicySet} \quad (4.42)$$

The relationship also expresses the relation that the **PolicySet** is superordinate to **Policy**, the **Policy** is superordinate to **Rule** and therefore the **PolicySet** is superordinate to **Rule**.

Subject, Operation and Resource Matching

The introduced model supports the use of attribute value pair matching in order to find an applicable permission. This matching actually takes place between the **SM**, **OM** and **RM** expressions of a permission and a request. Because no request is available, matching expressions of permissions must be checked for correlation. Such an evaluation is required on all levels of the permission tree: **PolicySet**, **Policy** and **Rule**.

For simplicity, it is assumed that the matching expressions are defined for the same attribute. For example, two permissions use the subject attribute `id` for matching a subject. Then, two permissions match to the same request based the subjects or operations if their matching expressions satisfy the conditions, enumerated in table 4.6.

Type SM_1	Type SM_2	Condition
String	String	The condition is satisfied if both matching expressions are identical: $SM_1 \equiv SM_2$.
String	Reg. expr.	The condition is satisfied if the string S_1 is a word of the language that the reg.-expr. defines.: $SM_1 \in \mathcal{L}(SM_2)$
Reg. expr.	Reg.expr.	The condition is satisfied if the language defined by SM_1 is a subset of the language, defined by SM_2 : $\mathcal{L}(SM_1) \subseteq \mathcal{L}(SM_2)$

Table 4.6: Evaluation of matching criteria for permissions

For explaining further proceedings, let `Match` be the function that evaluates the matching according to table 4.6 for the subject, operation and resource.

$$Match(string_1, string_2) \rightarrow \begin{cases} True & | string_1 \equiv string_2 \\ False & | string_1 \neq string_2 \end{cases} \quad (4.43)$$

$$Match(expr., string) \rightarrow \begin{cases} True & | string \in \mathcal{L}(expr.) \\ False & | string \notin \mathcal{L}(expr.) \end{cases} \quad (4.44)$$

$$Match(expr._1, expr._2) \rightarrow \begin{cases} True & | \mathcal{L}(expr._1) \subseteq \mathcal{L}(expr._2) \\ False & | \mathcal{L}(expr._1) \not\subseteq \mathcal{L}(expr._2) \end{cases} \quad (4.45)$$

Resource Content Object Matching based on Xpath Expressions

The introduced access control system protects the access to online resources. These resources are requested from services, which encode the response in GML. This response contains the resource content, for which the access restrictions are enforced. It is required that the resource content is valid GML, according to the definitions of the associated GML application schema. This schema defines the possible elements, their XML markup as well as the markup of the resource content itself. For the approximation, a concrete resource content is not available. However, the use of a resource content template can be used to evaluate the Xpath based resource matching. Therefore, it allows to test if the essential condition of resource content matching is satisfied. For the class-based restrictions, where no conditions on the resource content is necessary, the template provides sufficient resource information. For the object-based restrictions, the template does not provide sufficient information. For the spatial restrictions, the template provides useful information about the dimension of resource object geometries.

A resource template contains of elements that represent resource objects of the supported classes; hence feature types. But, it does not carry characterizing information of the objects. The listing 4.3 provides a resource content template for the City Model, according to the previously defined GML application schema from figure 2.8.


```

1 <CityModel xmlns="http://www.in.tum.de/am" xmlns:am="http://www.in.tum.de/am"
2   xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://http://www.in.tum.de/am CityModel.xsd"
5   fid="CityModel">
6   <gml:boundedBy>
7     <gml:Box><gml:coordinates/></gml:Box>
8   </gml:boundedBy>
9   <gml:featureMember>
10    <Street>
11      <name/>
12      <location><gml:coord><gml:X/><gml:Y/></gml:coord></location>
13    </Street>
14  </gml:featureMember>
15  <gml:featureMember>
16    <Building>
17      <address/>
18      <shape>
19        <gml:outerBoundaryIs>
20          <gml:LinearRing><gml:coordinates/></gml:LinearRing>
21        </gml:outerBoundaryIs>
22      </shape>
23    </Building>
24  </gml:featureMember>
25  <gml:featureMember>
26    <Street>
27      <name/>
28      <line><gml:coordinates/></line>
29    </Street>
30  </gml:featureMember>
31 </CityModel>

```

Listing 4.3: Resource content template for the City Model

Let \mathcal{RT} be the resource content template and xp be the value expression for the resource content matching XM_R , then Match_{XR} is the function that returns *True*, if at least one resource element could be fetched, based on the Xpath expression. Let the function $XQuery$ be the function that fetches XML nodes for a given Xpath and XML document.

$$\text{Match}_{XR}(xp, \mathcal{RT}) \rightarrow \begin{cases} \text{True} & | XQuery(xp, \mathcal{RT}) \neq \emptyset \\ \text{False} & | XQuery(xp, \mathcal{RT}) \equiv \emptyset \end{cases} \quad (4.46)$$

4.4.2 Approximate Detection of Unreachable Class-Based Permissions

The approximation of the existence of unreachable class-based permissions is possible without the knowledge about possible subjects, operations and resources. Assuming two permissions with a super/subordinated relation match according to the subject and operation expressions, the matching on the resource content must be determined. For XACML encoded permissions, unattainability can occur on two levels of the permission tree: Policy and Rule. The PolicySet as the top level node of the tree is always reachable but may not always match. The class-based permission does not require the use of a condition. Therefore, the unattainability for a Policy or Rule can be determined in the same way. It depends on the matching expression of the Target element only.

Reachable: A Policy or Rule is reachable if the essential condition is satisfied: The matching

for SM, OM and RM is satisfied. In addition it is required that the resource content matching of both constructs (XM) is satisfied in the following way: The matching of resource content objects for the subordinate construct must be a subset of the resource content objects, matched by the superordinate construct. This can be formalized as follows:

$$\begin{aligned} & Match(SM_P, SM_{PS}) \wedge Match(OM_P, OM_{PS}) \wedge Match(RM_P, RM_{PS}) \wedge \\ & (XQuery(XM_{PS}, \mathcal{RT}) \supseteq XQuery(XM_P, \mathcal{RT})), P \angle PS \\ \Leftrightarrow & \text{Policy reachable} \end{aligned} \quad (4.47)$$

$$\begin{aligned} & Match(SM_R, SM_P) \wedge Match(OM_R, OM_P) \wedge Match(RM_R, RM_P) \wedge \\ & (XQuery(XM_P, \mathcal{RT}) \supseteq XQuery(XM_R, \mathcal{RT})), R \angle P \\ \Leftrightarrow & \text{Rule reachable} \end{aligned} \quad (4.48)$$

An example for a reachable rule is the following permission hierarchy, which is based on the City Model resource template:

$$\begin{aligned} P1 &= \{Alice, r, \epsilon, //Street, \dots, R1\} \\ R1 &= \{Alice, *, \epsilon, //Street, \epsilon\} \rightarrow \dots \\ & Match(Alice, Alice) \rightarrow True \\ & Match(*, r) \rightarrow True \\ & Match(\epsilon, \epsilon) \rightarrow True \\ X1 &= XQuery(//Street, \{Building, Street, Intersection\}) = \{Street\} \\ X2 &= XQuery(//Street, \{Building, Street, Intersection\}) = \{Street\} \\ X1 &\supseteq X2 \rightarrow True \\ \Rightarrow & \text{Rule is reachable} \end{aligned}$$

Potentially unreachable: A Policy or Rule is potentially unreachable if the essential condition is satisfied -the Match for SM, OM and RM are satisfied- but the resource content matching does not refer to the same class. In such a case, a conclusion can only be made for a particular resource content. This is because the resource content template contains objects that represent all possible classes. For this content, a subordinated policy or rule can always be reached, even if the Xpath expressions do not refer to the same class. But, if a concrete resource content does not contain objects, which are required to make the superordinate PolicySet or Policy applicable, the subordinated Policy or Rule is not reachable. This can be formulated as follows:

$$\begin{aligned} & Match(SM_P, SM_{PS}) \wedge Match(OM_P, OM_{PS}) \wedge Match(RM_P, RM_{PS}) \wedge \\ & (XQuery(RM_{PS}, \mathcal{RT}) \not\supseteq XQuery(RM_P, \mathcal{RT})), P \angle PS \\ \Leftrightarrow & \text{Policy potentially unreachable} \end{aligned} \quad (4.49)$$

$$\begin{aligned} & Match(SM_R, SM_P) \wedge Match(OM_R, OM_P) \wedge Match(RM_R, RM_P) \wedge \\ & (XQuery(XM_P, \mathcal{RT}) \not\supseteq XQuery(XM_R, \mathcal{RT})), R \angle P \\ \Leftrightarrow & \text{Rule potentially unreachable} \end{aligned} \quad (4.50)$$

For example, the resource template from listing 4.3 contains of elements that represent the classes **Building**, **Street** and **Intersection**. Let the policy P1 use the resource matching expression *//am:Building* and the subordinated rule R1 uses the resource matching expression *//am:Street*.

$$\begin{aligned}
P1 &= \{Alice, r, \epsilon, //Building, \dots, R1\} \\
R1 &= \{Alice, *, \epsilon, //Street, \epsilon\} \rightarrow \dots \\
&\quad Match(Alice, Alice) \rightarrow True \\
&\quad Match(*, r) \rightarrow True \\
&\quad Match(\epsilon, \epsilon) \rightarrow True \\
X1 &= XQuery(//Building, \{Building, Street, Intersection\}) = \{Building\} \\
X2 &= XQuery(//Street, \{Building, Street, Intersection\}) = \{Street\} \\
&\Rightarrow \text{Rule R1 is potentially unreachable}
\end{aligned}$$

Unreachable: A Policy or Rule is unreachable if the essential condition is not satisfied; at least one of the matches for SM, OM and RM is not satisfied. In such a case, the matching for the resource content must not be determined.

$$\begin{aligned}
&\overline{Match(SM_P, SM_{PS})} \vee \overline{Match(OM_P, OM_{PS})} \vee \overline{Match(RM_P, RM_{PS})}, P \not\perp PS \\
&\Leftrightarrow \text{Policy unreachable} \tag{4.51}
\end{aligned}$$

$$\begin{aligned}
&\overline{Match(SM_R, SM_P)} \vee \overline{Match(SM_R, SM_P)} \vee \overline{Match(RM_R, RM_P)}, R \not\perp P \\
&\Leftrightarrow \text{Rule unreachable} \tag{4.52}
\end{aligned}$$

For example if a superordinate policy matches the subject *Alice* and the subordinate rule matches the subject **Bob**, this rule is never reached.

$$\begin{aligned}
P1 &= \{Alice, *, \epsilon, //Building, \dots, R1\} \\
R1 &= \{Bob, *, \epsilon, //Building, \epsilon\} \rightarrow \dots \\
&\quad Match(Bob, Alice) \rightarrow False \\
&\quad Match(*, *) \rightarrow True \\
&\quad Match(\epsilon, \epsilon) \rightarrow True \\
&\Rightarrow \text{Policy is unreachable}
\end{aligned}$$

4.4.3 An Illustrating Example of Unreachable Permissions

In order to illustrate the detection of unreachable permissions, the following declarations¹⁵, as illustrated in figure 4.45, are used:

$$\begin{aligned}
 PS &= \{*, *, *, \epsilon, \dots, P1, P2, P3\} \\
 P1 &= \{Alice, *, \epsilon, //Building, \dots, R11, R12\} \\
 P2 &= \{Alice, *, \epsilon, //*, \dots, R21, R22\} \\
 P3 &= \{Bob, *, \epsilon, //*, \dots, R3\} \\
 R11 &= \{*, r, \epsilon, //Street, \epsilon\} \rightarrow \dots \\
 R12 &= \{*, w, \epsilon, //Street, \epsilon\} \rightarrow \dots \\
 R21 &= \{*, r, \epsilon, //Building, \epsilon\} \rightarrow \dots \\
 R22 &= \{Bob, w, \epsilon, //Building, \epsilon\} \rightarrow \dots \\
 R3 &= \{Alice, *, \epsilon, //*, \epsilon\} \rightarrow \dots
 \end{aligned}$$

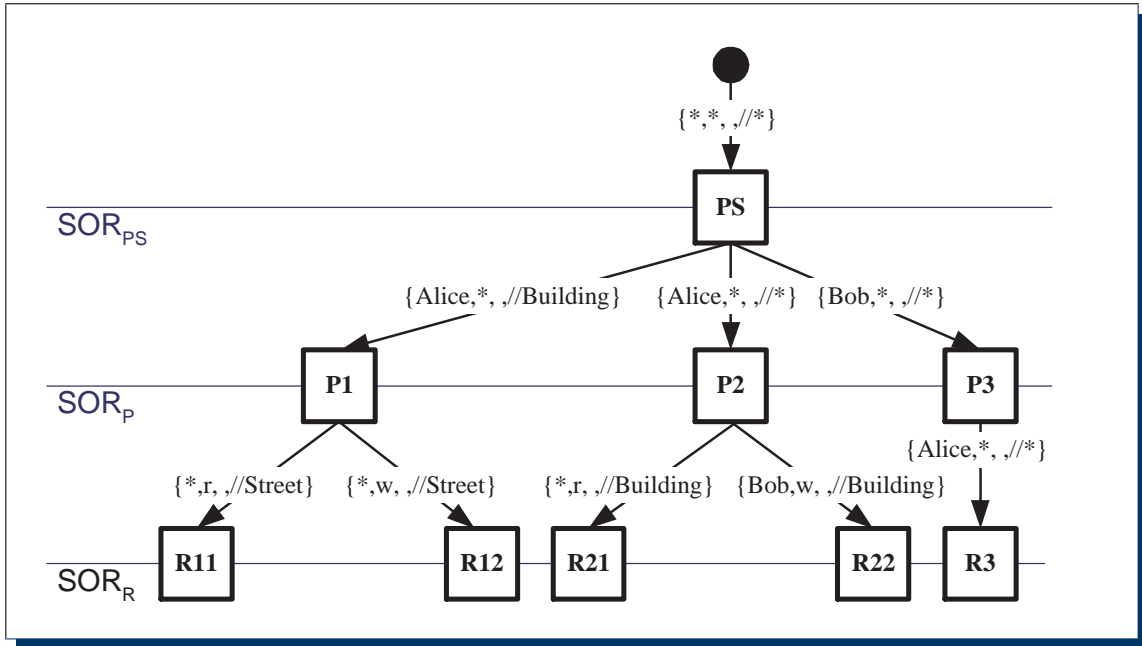


Figure 4.45: A permission tree with unreachable and potentially unreachable rules

This example is based on the resource content template \mathcal{RT} , as enumerated in listing 4.3.

$$\mathcal{RT} = \{Building, Street, Intersection\}$$

¹⁵The used combining algorithms and rule effects are irrelevant.

R11, R12: The rules R11 and R12 are potentially unreachable.

$$\begin{aligned}
Match_S &= Match(SM_{R11}, SM_{P1}) = Match(*, Alice) \rightarrow True \\
Match_O &= Match(OM_{R11}, OM_{P1}) = Match(r, *) \rightarrow True \\
Match_R &= Match(RM_{R11}, RM_{P1}) = Match(\epsilon, \epsilon) \rightarrow True \\
\mathcal{RO}_{P1} &= (XQuery(XM_{P1}, \mathcal{RT})) = \{Building\} \\
\mathcal{RO}_{R11} &= (XQuery(XM_{R11}, \mathcal{RT})) = \{Street\} \\
Match_S \wedge Match_O \wedge Match_R \wedge \mathcal{RO}_{R11} &\not\subseteq \mathcal{RO}_{P1} \rightarrow True \\
&\Rightarrow \text{potentially unreachable}
\end{aligned}$$

$$\begin{aligned}
Match_S &= Match(SM_{R12}, SM_{P1}) = Match(*, Alice) \rightarrow True \\
Match_O &= Match(OM_{R12}, OM_{P1}) = Match(w, *) \rightarrow True \\
Match_R &= Match(RM_{R12}, RM_{P1}) = Match(\epsilon, \epsilon) \rightarrow True \\
\mathcal{RO}_{P1} &= (XQuery(XM_{P1}, \mathcal{RT})) = \{Building\} \\
\mathcal{RO}_{R12} &= (XQuery(XM_{R12}, \mathcal{RT})) = \{Street\} \\
Match_S \wedge Match_O \wedge Match_R \wedge \mathcal{RO}_{R12} &\not\subseteq \mathcal{RO}_{P1} \rightarrow True \\
&\Rightarrow \text{potentially unreachable}
\end{aligned}$$

R21: The rule R21 is a reachable rule.

$$\begin{aligned}
Match_S &= Match(SM_{R21}, SM_{P2}) = Match(*, Alice) \rightarrow True \\
Match_O &= Match(OM_{R21}, OM_{P2}) = Match(r, *) \rightarrow True \\
Match_R &= Match(RM_{R21}, RM_{P1}) = Match(\epsilon, \epsilon) \rightarrow True \\
\mathcal{RO}_{P2} &= (XQuery(XM_{P2}, \mathcal{RT})) = \{Building, Street, Intersection\} \\
\mathcal{RO}_{R21} &= (XQuery(XM_{R21}, \mathcal{RT})) = \{Building\} \\
Match_S \wedge Match_O \wedge Match_R \wedge \mathcal{RO}_{R21} &\subseteq \mathcal{RO}_{P2} \Rightarrow \text{reachable}
\end{aligned}$$

R22: The rule R22 is unreachable, because it results from the matching of the subject and operation from P2 and R22.

$$\begin{aligned}
Match_S &= Match(SM_{R22}, SM_{P2}) = Match(Bob, Alice) \rightarrow False \\
Match_O &= Match(OM_{R22}, OM_{P2}) = Match(w, *) \rightarrow True \\
Match_R &= Match(RM_{R22}, RM_{P2}) = Match(\epsilon, \epsilon) \rightarrow True \\
Match_S \wedge Match_O \wedge Match_R &\Rightarrow \text{unreachable}
\end{aligned}$$

R3: The rule R3 is also unreachable, because it results from the matching of the subject and operation from P3 and R3.

$$\begin{aligned}
Match_S &= Match(SM_{R3}, SM_{P3}) = Match(Alice, Bob) \rightarrow False \\
Match_O &= Match(OM_{R3}, OM_{P3}) = Match(*, *) \rightarrow True \\
Match_R &= Match(RM_{R3}, RM_{P3}) = Match(\epsilon, \epsilon) \rightarrow True \\
Match_S \wedge Match_O \wedge Match_R &\Rightarrow \text{unreachable}
\end{aligned}$$

The resulting annotated permission tree is shown in figure 4.46. The reachable policy set, policies and rule R21 are not colored. The potentially unreachable rules R11 and R12 are colored yellow and the unreachable rules R22 and R3 are colored red.

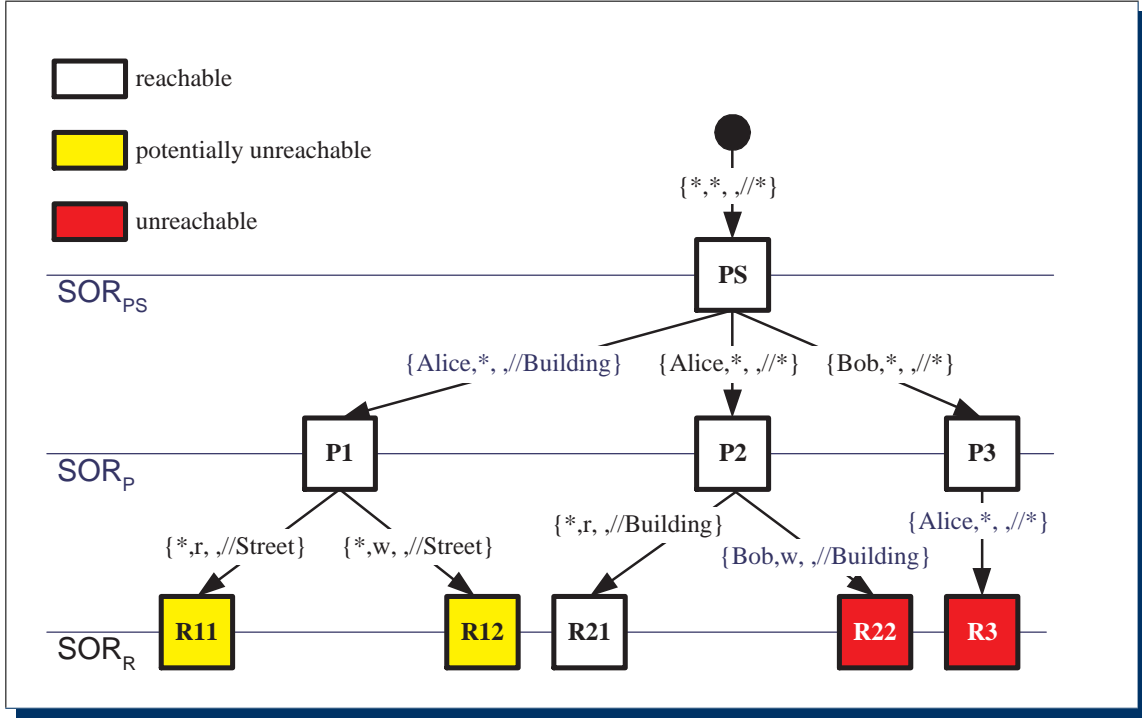


Figure 4.46: An annotated permission tree with unreachable rules

4.4.4 Approximate Detection of Complete Class-Based Permissions

The approximate detection of the existence of complete class-based permissions is possible without the knowledge about possible subjects, operations and resources. However, the only solution of completeness is that a permission sub-tree exists, which matches the all-quantifier $\{*,*,*\}$ for subject, operation and resource on all levels. For the top level, this is the PolicySet. In addition it must be satisfied that at least one subordinate permission of that policy set also provides the all-quantifier matching. For the XACML permission hierarchy, this requires that at least one subordinate Policy exists that provides the all-quantifier matching. The same applies to the subordinate rules of that policy. At least one Rule must exist that provides an all-quantifier matching.

Let \mathcal{PS} be the set of top-level PolicySet permissions and let PS_i be the i -th permission of the PolicySet. Let PS_{Ci} be the i th policy set that satisfies the condition to define the all-quantifier matching. Then, the top level of the permission tree provides a complete matching if at least one policy set exists, where $SM_{PS} = OM_{PS} = RM_{PS} = "*"$ and $XM_{PS} = "/*"$.

$$PS_{Ci} := (SM_{PS_i} = OM_{PS_i} = RM_{PS_i} = "*" \wedge XM_{PS_i} = "/*") \quad (4.53)$$

$$\text{PolicySet level complete} \Leftrightarrow \bigvee_i^{\|\mathcal{PS}\|} PS_{Ci} \rightarrow \text{True} \quad (4.54)$$

The Policy level of the sub-tree defines a complete matching, if at least one policy provides the all-quantifier matching. And, this policy must be subordinate to the PolicySet, which defines the all-quantifier matching in the superordinate level. Let \mathcal{P} be the set of policy permissions and let P_j be the j th policy. Let P_{Cj} be the j th policy that satisfies the condition by defining the all-quantifier matching and be subordinate to the i th permission of the PolicySet, which satisfies the condition PS_{Ci} . Then, the policy level provides a complete matching if at least one policy exists, where $SM_P = OM_P = RM_P = "*" \wedge XM_P = "/*"$ and the superordinate policy set PS_i satisfies PS_{Ci} .

$$P_{Cj} := (SM_{Pj} = OM_{Pj} = RM_{Pj} = "*" \wedge XM_{Pj} = "/*") \quad (4.55)$$

$$\text{Policy level complete} \Leftrightarrow \bigvee_j^{\|\mathcal{P}\|} (P_{Cj} \mid P_j \angle PS_i \wedge PS_{Ci}) \rightarrow True \quad (4.56)$$

The Rule level of the sub-tree defines a complete matching if at least one rule provides the all-quantifier matching. And, this rule must be subordinate to the policy, which defines the all-quantifier matching in the superordinate level. Let \mathcal{R} be the set of rule permissions and let R_k be the k th rule. Let R_{Ck} be the k th rule that satisfies the condition by defining the all-quantifier matching and be subordinate to the j th policy, which satisfies the condition P_{Cj} . Then the rule level provides a complete matching if at least one rule exists, where $SM_R = OM_R = RM_R = "*" \wedge XM_R = "/*"$ and the superordinate policy P_j satisfies P_{Cj} .

$$R_{Ck} := (SM_{Rk} = OM_{Rk} = RM_{Rk} = "*" \wedge XM_{Rk} = "/*") \quad (4.57)$$

$$\text{Rule level complete} \Leftrightarrow \bigvee_k^{\|\mathcal{R}\|} (R_{Ck} \mid R_k \angle P_j \wedge P_{Cj}) \rightarrow True \quad (4.58)$$

One example of a permission tree that provides complete matching on all levels is shown in figure 4.47

4.4.5 Approximate Detection of Contrary Class-Based Permissions

The approximate detection of the existence of contrary class-based permissions is possible without the knowledge about possible subjects, operations and resources. Under these pre-conditions, two reachable permissions are contrary if they define contrary effects. For the XACML encoding, only the Rule defines an effect that is either Deny or Permit. The essential condition is that the rule is reachable. Otherwise it does not influence the process of deriving an authorization decision. Also, the rules must be subordinate to the same policy. Let $R1$ and $R2$ be reachable rules that are subordinate to the policy P ($R1 \angle P$) and ($R2 \angle P$). Further let E_{R1} be the effect of rule $R1$ and E_{R2} be the effect of rule $R2$. Then, the contradiction of

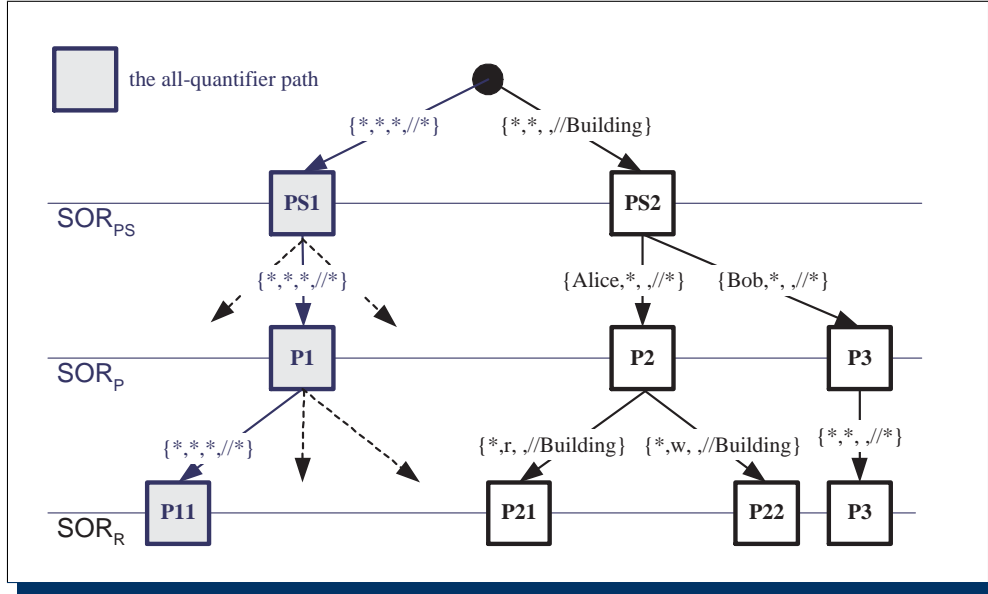


Figure 4.47: A permission tree with a sub-tree that provides complete (all-quantified) matching

R1 and R2 can be detected as follows:

$$\begin{aligned}
 I_{R1,R2}^C := & \mathcal{L}(SM_{R1}) \cap \mathcal{L}(SM_{R2}) \neq \emptyset \wedge \mathcal{L}(OM_{R1}) \cap \mathcal{L}(OM_{R2}) \neq \emptyset \wedge \\
 & \mathcal{L}(RM_{R1}) \cap \mathcal{L}(RM_{R2}) \neq \emptyset \wedge \\
 & (XQuery(XM_{R1}, \mathcal{RT}) \equiv XQuery(XM_{R2}, \mathcal{RT})) \quad (4.59)
 \end{aligned}$$

$$R1 \text{ is contrary to } R2 \Leftrightarrow I_{R1,R2}^C \wedge (E_{R1} \neq E_{R2}) \mid R1 \angle P, R2 \angle P, R1 \neq R2 \quad (4.60)$$

4.4.6 Approximate Detection of Unreachable Object-Based Permissions

The approximate detection of the existence of unreachable object-based permissions is possible without the knowledge about possible subjects, operations and resources. The proceeding is identical to the detection of unreachable class-based permissions. For the object-based permission, the condition of a rule defines additional matching on the non-spatial characteristics of resource objects. This additional constraint reduces the set of matching resource objects, which defines the distinctive difference to the class-based permission. Therefore, an object-based permission is reachable, if the essential conditions for detection of a reachable class-based permission can be satisfied. In the same manner, an object-based permission is potentially unreachable or unreachable, as defined for the class-based permission.

4.4.7 Approximate Detection of Complete Object-Based Permissions

The approximate detection of the existence of complete object-based permissions is not possible without the knowledge about available resources. The condition of a rule reduces the

set of matched resource content objects based on non-spatial characteristics. The detection of a complete resource matching can only be achieved by comparing the union of all matched resource objects from all rules and the superordinate policy. But, this is -per definition- not possible.

It is also questionable if this detection is necessary and meaningful, because the policy writer must not have the intention to declare object-based restrictions in such a way that they result in a complete matching. Typically, an object-oriented permission is declared to express an exception to a general permission, encoded as a class-based restriction.

4.4.8 Approximate Detection of Contrary Object-Based Permissions

The approximate detection of the existence of contrary object-based permissions is possible without the knowledge about possible subjects, operations and resources. Assuming that two rules are reachable and match according to the subject and operation, the resource matching and the condition constraint must be evaluated.

It is the essential condition that both rules match the same class, defined through the resource match expression. The refinement matching, defined by the condition of the rule can apply to any (non-spatial) attribute of the resource object. If the conditional matching of two rules do not refer to the same attribute, no approximation is possible. This relies on the limitation that it cannot be verified if the conditional matching refers to the same resource object. Only if both conditions match for the same attribute, a further evaluation is possible. In this case it can be tested if both matching patterns select the same object. This can be evaluated, using the Match_{XR} function.

Let C_{R1} and C_{R2} be the conditions of rules R1 and R2. And, let XM_{C1}^O be the resource content matching expression of condition C_{R1} and XM_{C2}^O be the resource content matching expression of condition C_{R2} . Then, two rules are contrary if they are reachable, declare different effects and match the same resource content objects, defined by XM_{C1}^O and XM_{C2}^O .

$$\begin{aligned} X_1^O &:= XQuery(XM_{R1}, \mathcal{RT}) \cap XQuery(XM_{C1}^O, \mathcal{RT}) \\ X_2^O &:= XQuery(XM_{R2}, \mathcal{RT}) \cap XQuery(XM_{C2}^O, \mathcal{RT}) \\ I_{R1,R2}^O &:= \mathcal{L}(SM_{R1}) \cap \mathcal{L}(SM_{R2}) \neq \emptyset \wedge \mathcal{L}(OM_{R1}) \cap \mathcal{L}(OM_{R2}) \neq \emptyset \wedge \\ &\quad \mathcal{L}(RM_{R1}) \cap \mathcal{L}(RM_{R2}) \neq \emptyset \wedge X_1^O \cap X_2^O \neq \emptyset \end{aligned} \quad (4.61)$$

$$R1 \text{ is contrary to } R2 \Leftrightarrow I_{R1,R2}^O \wedge (E_{R1} \neq E_{R2}) \mid R1 \angle P, R2 \angle P, R1 \neq R2 \quad (4.62)$$

4.4.9 Approximate Detection of Unreachable Spatial Permissions

The reaching of spatial permissions depend on the subject, operation and resource matching in the first place. Therefore, the reaching of a spatial permission can be defined as for a class-based permission. Because the spatial characteristics is only evaluated in the condition of the rule, it has not effect on the matching for the superordinate policy and policy set.

4.4.10 Approximate Detection of Incomplete Spatial Permissions

The approximate detection of spatial permissions define a complete set cannot be verified without the knowledge of possible resource object geometry. The spatial permissions redefine the matching for resource objects, based on their spatial characteristics. If the possible geometries are unknown, it cannot be verified if all resource object geometries are handled by declared spatial permissions.

It is also questionable if this detection is necessary and meaningful, because the policy writer must not have the intention to declare spatial restrictions in such a way that they result in a complete set. Typically, spatial permissions declare permissions for restricted areas only.

4.4.11 Approximate Detection of Contrary Spatial Permissions

The approximate detection if contrary spatial permissions is possible without the knowledge of concrete subjects, operations and resources. In particular, the approximation is possible without having specific information about the geometries of the resource objects.

As for the class-based permissions, the contrary spatial permission must be reachable in order to influence the authorization decision process. The XACML encoding of a spatial permission is possible with a `Rule` construct that uses a `Condition` construct that expresses a spatial condition.

The essential condition for two rules to be contrary is that they are reachable and both match the same subject, operation and resource, as defined for reachable class-based restrictions. Two rules, satisfying this condition must declare different effects in order to be contrary.

The sufficient condition for the two rules to be actually contrary, depend on the spatial matching constraint defined by the condition. For the spatial permission, the condition defines a topological relation between a fixed permission geometry and a resource object geometry. The rule applies if the topological relation can be satisfied. The topological relation is expressed by the already defined spatial methods `Disjoint`, `Touches`, `Crosses`, `Within`, `Overlaps`, `Intersects` and `Equals`.

According to the essential condition, both rules apply to resource objects of the same class. The spatial condition therefore applies to the same resource object. For the following detection, it is essential to differentiate between the geometry of the resource object from rule R1 and the geometry of the resource object from rule R2. This is because it is possible that different spatial properties are matched by the spatial condition of rule R1 and rule R2. As an example, let's use one geospatial information object representing an area of oil resources in the Atlantic. This area might have different spatial properties of 0D geometry, defining the locations of oil platforms: `locationOfPlatform1` defines the location of platform one, `locationOfPlatform2` defines the location of platform two, etc. Then, it is possible that rule R1 matches the property `locationOfPlatform1` and rule R2 matches the property `locationOfPlatform2`. From that matching, G_{R1} represents the geometry, represented by the property `locationOfPlatform1` and G_{R2} represents the geometry of `locationOfPlatform2` respectively.

Let R1 and R2 be the rules to be tested, G_{P1} and G_{P2} the permission geometries of R1

and R2 and let G_{R1} and G_{R2} be the resource content object geometries that are matched by the spatial conditions of rule R1 and R2. Then, two rules are contrary if the following conditions can be satisfied at the same time:

$$SC_1 := SpatialRelation_1(G_{R1}, G_{P1}) \quad (4.63)$$

$$SC_2 := SpatialRelation_2(G_{R2}, G_{P2}) \quad (4.64)$$

$$SC_3 := SpatialRelation_1(G_{R2}, G_{P1}) \quad (4.65)$$

$$SC_4 := SpatialRelation_2(G_{R1}, G_{P2}) \quad (4.66)$$

SC₁: The spatial condition SC_1 defines the condition that the resource object geometry of the spatial relation of rule R1 must satisfy the topological relation with the permission geometry of rule R1. This essential condition must be satisfied in order to make rule R1 become applicable.

SC₂: The spatial condition SC_2 defines the condition that the resource object geometry of the spatial relation of rule R2 must satisfy the topological relation with the permission geometry of rule R2. This essential condition must be satisfied in order to make rule R2 become applicable.

SC₃: This spatial condition is the first test condition that must be satisfied, in order to have rule R2 apply to the resource object geometry, matched by rule R1. Therefore, the resource geometry of rule R2 (G_{R2}) must satisfy the spatial relation of rule R1 ($SpatialRelation_1$) with the permission geometry of rule R1 (G_{P1}).

SC₄: This spatial condition is the second test condition that must be satisfied, in order to have rule R1 apply to the resource object geometry, matched by rule R2. Therefore, the resource geometry of rule R1 (G_{R1}) must satisfy the spatial relation of rule R2 ($SpatialRelation_2$) with the permission geometry of rule R2 (G_{P2}).

For the evaluation, under which circumstances the spatial conditions SC_3 and SC_4 can be satisfied, it is assumed that the dimension of the permission geometry is 2D ($\dim(G_{P1}) = \dim(G_{P2})=2$). Thus, the permission geometry defines a 2D surface, encoded as a **Polygon**.

The satisfiability of SC_3 and SC_4 depend on two factors: the dimension of the resource geometries and the topological relation between the permission geometries. The likelihood that SC_3 and/or SC_4 can be satisfied at the same time can be classified into **Impossible**, **Likely**, **Assured**. The classification **Impossible** describes the constellation that can never be satisfied at the same time. In such sense, **Assured** represents the constellation that the conditions SC_3 and SC_4 are always satisfied. **Likely** describes a constellation, where SC_3 and SC_4 may get satisfied and no condition can be defined that either result in the classification **Impossible** or **Assured**.

Let $\mathcal{CL} := \{\text{Impossible (I), Likely (L), Assured (A)}\}$ be the set of possible classifications, $I \prec L \prec A$ be the superordinate relation and $\mathcal{CL}_s \subseteq \mathcal{CL}$. Then, $Max_{\prec}(\mathcal{CL}_s) \rightarrow cl, cl \in \mathcal{CL}$ is

the function that selects the maximum superordinate classification.

$$\mathcal{CL} := \{I, L, A\}, I \angle L \angle A, cl \angle cl \mid cl \in \mathcal{CL} \quad (4.67)$$

$$Max_{\angle}(\mathcal{CL}_s) \rightarrow \begin{cases} I & \mid \exists I \in \mathcal{CL}_s, \forall cl_i \in \mathcal{CL}_s : cl_i \angle I \\ L & \mid \exists L \in \mathcal{CL}_s, \forall cl_i \in \mathcal{CL}_s : cl_i \angle L \\ A & \mid \exists A \in \mathcal{CL}_s, \forall cl_i \in \mathcal{CL}_s : cl_i \angle A \end{cases} \quad (4.68)$$

Let $CL_1(SC_3)$ be a relation that returns the classification for the given SpatialRelation of the constellation SC_3 and let $CL_2(SC_4)$ be a relation that returns the classification for the given SpatialRelation of the constellation SC_4 . Then, the classification cl of the satisfiability that rule R1 and rule R2 are contrary, is the most superordinate classification.

$$cl_1(G_{R2}, G_{P1}) \rightarrow cl_1, cl_1 \in \mathcal{CL} \quad (4.69)$$

$$cl_2(G_{R1}, G_{P2}) \rightarrow cl_2, cl_2 \in \mathcal{CL} \quad (4.70)$$

$$cl := Max_{\angle}(cl_1 \cup cl_2) \quad (4.71)$$

Before starting the classification of possible constellations, the worst case number of different constellations has the following outcome: There are seven (7) different spatial methods defined that can be used to represent the spatial conditions SC_3 and SC_4 . This results in twenty-eight (28) possible permutations¹⁶ for the spatial conditions (Disjoint-Disjoint, Disjoint-Touches, ... Intersects-Equals, Equals-Equals). For each permutation, it is required to test all possible topological relations between the permission geometries. Because the spatial method Crosses does not apply to 2D geometries, only six (6) spatial methods must be evaluated. However, the spatial relation Within is not a symmetrical relation. Therefore the topological relations $Within(G_{P1}, G_{P2})$ is different from $Within(G_{P2}, G_{P1})$. This results in seven (7) different topological relations between the permission geometries. For the worst case, different evaluations are required for each dimension of the resource object geometry from rule R1 and rule R2. The possible dimensions are 0D, 1D and 2D, which results in nine (9) permutations for the geometry dimension. This results in $28 * 7 * 9 = 1764$ testable constellations for the worst case.

In order to reduce the possible constellations to a reasonable number, the topological relations can be limited. This can be done according to the informal poll, where the two most important topological relations have been named as Within and Touches. Within allows the declaration of permissions for a restricted area such as a sovereign, county or municipal territory or a military site. Touches allows to express spatial permissions for geospatial information objects that are adjacent with another geometry. The limitation of the possible spatial conditions results in three (3) different permutations. This results in $3 * 7 * 9 = 189$ constellations for the worst case. Fortunately, not all permutations of resource object geometry dimensions must be tested for each topological relation between the resource geometries of rules R1 and R2.

For the illustration of the different test constellations throughout the following subsections, the legend of figure 4.48 is being used. The resource geometries, represented by blue squares satisfy the spatial conditions SC_1 and SC_4 , resp. SC_2 and SC_3 . The resource geometries, represented with red circles do satisfy the conditions SC_1 , resp. SC_2 but do not satisfy the

¹⁶ $\sum_{i=1}^7 i = 28$

test condition SC_3 , resp. SC_4 . For example, a resource geometry G_{R1} represented as a red circle satisfies SC_1 but not SC_4 . In the same fashion, G_{R2} represented as a red circle satisfies SC_2 but not SC_3 .

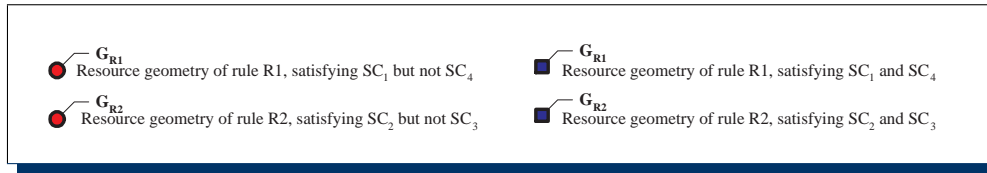


Figure 4.48: Classification legend for illustration of satisfying and not satisfying geometries

Classification of the Spatial Methods Within and Within

This situation appears if two reachable rules are tested that use the spatial condition *Within*. Two example rules that belong into this category are R1 and R2:

$$R1 = \{ \dots, //aClass, C1 \} \rightarrow Permit$$

$$R2 = \{ \dots, //aClass, C2 \} \rightarrow Deny$$

$$C1 = \{ Within(G_{R1}, G_{P1}) \}$$

$$C2 = \{ Within(G_{R2}, G_{P2}) \}$$

According to the previous considerations, it is essential that SC_1 and SC_2 are satisfied:

$$SC_1 = Within(G_{R1}, G_{P1}) \Leftrightarrow True$$

$$SC_2 = Within(G_{R2}, G_{P2}) \Leftrightarrow True$$

The spatial conditions that determine the classification are

$$SC_3 = Within(G_{R2}, G_{P1})$$

$$SC_4 = Within(G_{R1}, G_{P2})$$

For this test situation, the dimension of the resource geometries must not be considered. The pre-conditions $Within(G_{R1}, G_{P1})$ and $Within(G_{R2}, G_{P2})$ can only be satisfied if the resource geometries do not extend the interior of G_{P1} , resp. G_{P2} : $G_{R1} \cap I(G_{P1}) \neq \emptyset \wedge G_{R1} \cap B(G_{P1}) = G_{R1} \cap E(G_{P1}) \equiv \emptyset$. The classification depends on the satisfiability of SC_3 and SC_4 , which depend on the topological relation between G_{P1} and G_{P2} .

Disjoint(G_{P1}, G_{P2}) results in the classification I.

For this topological relation between G_{P1} and G_{P2} , as illustrated in figure 4.49, SC_3 and SC_4 can never be satisfied at the same time. A resource geometry G_{R1} that satisfies the essential spatial condition $Within(G_{R1}, G_{P1})$ can never satisfy the test condition $Within(G_{R1}, G_{P2})$. This is because G_{P1} and G_{P2} are disjoint. The same argument is

true for a resource geometry G_{R2} .

$$\begin{aligned} SC_3 &:= \text{Within}(G_{R2}, G_{P1}) \\ SC_4 &:= \text{Within}(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow I \\ cl_2 &:= CL_2(SC_4) \rightarrow I \\ cl &:= \text{Max}_\angle(\{I, I\}) \rightarrow I \end{aligned}$$

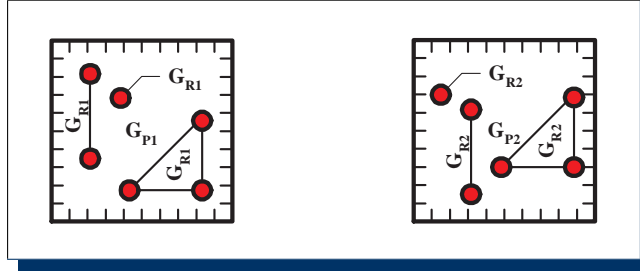


Figure 4.49: Illustrating the test constraint $\text{Disjoint}(G_{P1}, G_{P2})$

Touches(G_{P1}, G_{P2}) results in the classification I.

For this topological relation between G_{P1} and G_{P2} , SC_3 and SC_4 can never be satisfied at the same time¹⁷.

$$\begin{aligned} SC_3 &:= \text{Within}(G_{R2}, G_{P1}) \\ SC_4 &:= \text{Within}(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow I \\ cl_2 &:= CL_2(SC_4) \rightarrow I \\ cl &:= \text{Max}_\angle(\{I, I\}) \rightarrow I \end{aligned}$$

Overlaps(G_{P1}, G_{P2}) results in the classification L.

For this topological relation between G_{P1} and G_{P2} , as illustrated in figure 4.50, SC_3 and SC_4 can be satisfied at the same time. Let G_{P12} be the intersection of G_{P1} and G_{P2} ($G_{P12} = G_{P1} \cap G_{P2}$). Then, SC_3 and SC_4 are satisfied if the topological relations $\text{Within}(G_{R2}, G_{P12})$ and $\text{Within}(G_{R1}, G_{P12})$ are satisfied. This is possible, because G_{P12} is within G_{P1} and G_{P2} ($\text{Within}(G_{P12}, G_{P1}) \wedge \text{Within}(G_{P12}, G_{P2})$). The classification is Likely and not Assured, because the exterior of each resource object cannot be restricted to be within G_{P12} .

The spatial condition SC_3 cannot be satisfied if G_{R2} is inside of G_{P2} but outside G_{P1} . Let G_{P2-1} be the geometry area of G_{P2} minus the area of G_{P1} ($G_{P2-1} = G_{P2} \setminus G_{P1}$). Then, SC_3 cannot be satisfied if G_{R2} is within G_{P2-1} $\text{Within}(G_{R2}, G_{P2-1})$.

The spatial condition SC_4 cannot be satisfied if G_{R1} is inside of G_{P1} but outside G_{P2} . Let G_{P1-2} be the geometry area of G_{P1} minus the area of G_{P2} ($G_{P1-2} = G_{P1} \setminus G_{P2}$).

¹⁷See argument for $\text{Disjoint}(G_{P1}, G_{P2})$.

Then, SC_4 cannot be satisfied if G_{R1} within G_{P1-2} ($Within(G_{R1}, G_{P1-2})$).

$$G_{P12} := G_{P1} \cap G_{P2}$$

$$G_{P1-2} := G_{P1} \setminus G_{P2}$$

$$G_{P2-1} := G_{P2} \setminus G_{P1}$$

$$SC_3 := Within(G_{R2}, G_{P1})$$

$$SC_4 := Within(G_{R1}, G_{P2})$$

$$cl_1 := CL_1(SC_3) \rightarrow \begin{cases} L & | Within(G_{R2}, G_{P12}) \\ I & | Within(G_{R2}, G_{P2-1}) \end{cases}$$

$$cl_2 := CL_2(SC_4) \rightarrow \begin{cases} L & | Within(G_{R1}, G_{P12}) \\ I & | Within(G_{R1}, G_{P1-2}) \end{cases}$$

$$cl := Max_{\angle}(\{L, I\} \cup \{L, I\}) \rightarrow L$$

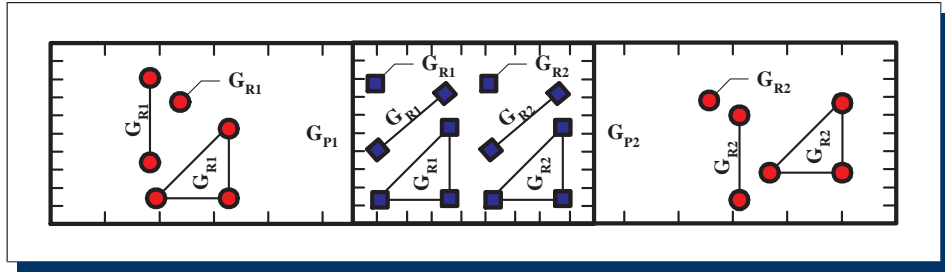


Figure 4.50: Illustrating the test constraint $Overlaps(G_{P1}, G_{P2})$

$Within(G_{P2}, G_{P1})$ results in the classification A.

For this topological relation between G_{P1} and G_{P2} , as illustrated in figure 4.51, SC_3 and SC_4 can be satisfied at the same time. The resource object geometry G_{R2} satisfies the essential spatial condition $Within(G_{R2}, G_{P2})$. This satisfies the test condition $SC_3 = Within(G_{R2}, G_{P1})$ at the same time, because G_{P2} is within G_{P1} ($Within(G_{P1}, G_{P2})$).

The resource geometry object does not necessarily satisfy the spatial condition SC_4 . Let G_{P1-2} be the geometry area G_{P1} minus G_{P2} ($G_{P1-2} = G_{P1} \setminus G_{P2}$) and let G_{P12} be the intersection of the geometries G_{P1} and G_{P2} ($G_{P12} = G_{P1} \cap G_{P2}$). Then, G_{R1} cannot satisfy the spatial condition SC_4 if within G_{P1-2} . But, G_{R1} can satisfy the spatial condition SC_4 if within G_{P12} .

$$G_{P12} := G_{P1} \cap G_{P2}$$

$$G_{P1-2} := G_{P1} \setminus G_{P2}$$

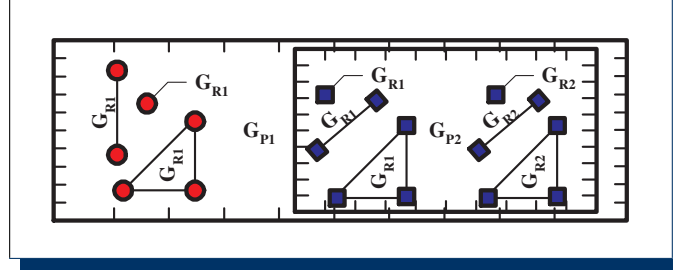
$$SC_3 := Within(G_{R2}, G_{P1})$$

$$SC_4 := Within(G_{R1}, G_{P2})$$

$$cl_1 := CL_1(SC_3) \rightarrow A$$

$$cl_2 := CL_2(SC_4) \rightarrow \begin{cases} L & | Within(G_{R1}, G_{P12}) \\ I & | Within(G_{R1}, G_{P1-2}) \end{cases}$$

$$cl := Max_{\angle}(\{A\} \cup \{L, I\}) \rightarrow A$$

Figure 4.51: Illustrating the test constraint $\text{Within}(G_{P2}, G_{P1})$

$\text{Within}(G_{P1}, G_{P2})$ results in the classification A.

For this topological relation, the same logical argument as for $\text{Within}(G_{P2}, G_{P1})$ is valid.

$\text{Equals}(G_{P1}, G_{P2})$ results in the classification A.

For this topological relation between G_{P1} and G_{P2} , as illustrated in figure 4.52, SC_3 and SC_4 are always satisfied at the same time. The resource object geometry G_{R2} satisfies the essential spatial condition $\text{Within}(G_{R2}, G_{P2})$. This satisfies the test condition $SC_3 = \text{Within}(G_{R2}, G_{P1})$ at the same time, because G_{P2} is equal to G_{P1} ($G_{P2} \equiv G_{P1}$).

The similar argument is valid for G_{R1} . The resource object geometry G_{R1} satisfies the essential spatial condition $\text{Within}(G_{R1}, G_{P1})$. This satisfies the test condition $SC_4 = \text{Within}(G_{R1}, G_{P2})$ at the same time, because G_{P1} is equal G_{P2} ($G_{P1} \equiv G_{P2}$).

$$\begin{aligned} SC_3 &:= \text{Within}(G_{R2}, G_{P1}) \\ SC_4 &:= \text{Within}(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow A \\ cl_2 &:= CL_2(SC_4) \rightarrow A \\ cl &:= \text{Max}_{\angle}(\{A\} \cup \{A\}) \rightarrow A \end{aligned}$$

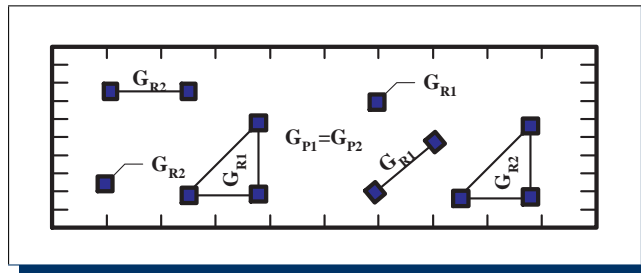
Figure 4.52: Illustrating the test constraint $\text{Equals}(G_{P1}, G_{P2})$

Table 4.7 combines the results from the different test cases.

Classification of the Spatial Methods Within and Touches

This situation appears if two rules are tested, where the spatial conditions from rule R1 is Within and from rule R2 is Touches. Two example rules, which fall into this category are R1

$\text{Within}(G_{R1}, G_{P1}) \wedge \text{Within}(G_{R2}, G_{P2}) \wedge \text{Within}(G_{R2}, G_{P1}) \wedge \text{Within}(G_{R1}, G_{P2})$			
SpatialRelation	dim(G_{R1})	dim(G_{R2})	classification
Disjoint(G_{P1}, G_{P2})	No restriction	No restriction	Impossible
Touches(G_{P1}, G_{P2})	No restriction	No restriction	Impossible
Equals(G_{P1}, G_{P2})	No restriction	No restriction	Assured
Overlaps(G_{P1}, G_{P2})	No restriction	No restriction	Likely
Within(G_{P1}, G_{P2})	No restriction	No restriction	Assured
Within(G_{P2}, G_{P1})	No restriction	No restriction	Assured

Table 4.7: Worst case classification for the occurrence of incorrect spatial restrictions, based on the spatial relations *Within* and *Within*

and R2:

$$\begin{aligned}
R1 &= \{ \dots, //aClass, C1 \} \rightarrow \text{Permit} \\
R2 &= \{ \dots, //aClass, C2 \} \rightarrow \text{Deny} \\
C1 &= \{ \text{Within}(G_{R1}, G_{P1}) \} \\
C2 &= \{ \text{Touches}(G_{R2}, G_{P2}) \}
\end{aligned}$$

According to the previous considerations, it is essential that SC_1 and SC_2 are satisfied:

$$\begin{aligned}
SC_1 &= \text{Within}(G_{R1}, G_{P1}) \Leftrightarrow \text{True} \\
SC_2 &= \text{Touches}(G_{R2}, G_{P2}) \Leftrightarrow \text{True}
\end{aligned}$$

The spatial conditions that determine the classification are

$$\begin{aligned}
SC_3 &= \text{Within}(G_{R2}, G_{P1}) \\
SC_4 &= \text{Touches}(G_{R1}, G_{P2})
\end{aligned}$$

For some topological relations between G_{P1} and G_{P2} , the classification must be differentiated based on the dimension of the resource object geometry G_{R2} . The essential spatial relation for G_{R2} is that is touching with G_{P2} . For all 0D resource geometries G_{R2} ($\text{dim}(G_{R2})=0$), the exterior is the empty set. This reduces the extension of such a geometry to its location. Therefore, all resource geometries that satisfy $\text{Touches}(G_{R2}, G_{P2})$ are located on the boundary of G_{P2} ($G_{R1} \in B(G_{P2})$). For resource geometries G_{R2} with the dimension 1D or 2D, the exterior cannot be restricted. The restriction of the possible exterior of the geometry G_{R2} – as for 0D – enables to approximate the satisfiability of the spatial constraint SC_3 . The exterior of the resource geometry G_{R1} is restricted by its topological relation with G_{P1} , independent from its dimension ($G_{R1} \cup I(G_{P1}) \neq \emptyset$).

Disjoint(G_{P1}, G_{P2}) results in the classification I.

For this topological relation between G_{P1} and G_{P2} , as illustrated in figure 4.53, it is not necessary to make assumptions on the dimension of G_{R2} . This is because SC_3 and SC_4 can never be satisfied at the same time.

For G_{R1} it is essential to satisfy the spatial condition SC_1 . This restricts the extension of G_{R1} to the interior of G_{P1} . This prevents that G_{R1} can satisfy the test condition SC_4 ($\text{Touches}(G_{R1}, G_{P2})$) at the same time, because G_{P1} and G_{P2} are topologically disjoint ($\text{Disjoint}(G_{P1}, G_{P2})$). In the same fashion, the resource geometry G_{R2} that satisfies the essential spatial condition C_2 cannot satisfy the spatial condition SC_3 at the same time.

$$\begin{aligned} SC_3 &:= \text{Within}(G_{R2}, G_{P1}) \\ SC_4 &:= \text{Touches}(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow I \\ cl_2 &:= CL_2(SC_4) \rightarrow I \\ cl &:= \text{Max}_\angle(\{I\} \cup \{I\}) \rightarrow I \end{aligned}$$

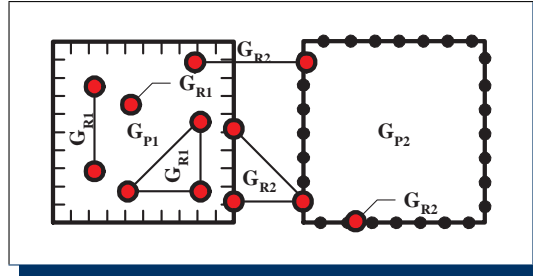


Figure 4.53: Illustrating the test constraint $\text{Disjoint}(G_{P1}, G_{P2})$

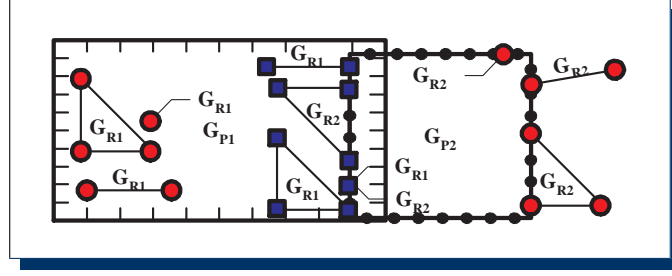
$\text{Touches}(G_{P1}, G_{P2})$ results in the classification I.

For this topological relation between G_{P1} and G_{P2} , it is not necessary to make assumptions on the dimension of G_{R2} . This is because SC_3 and SC_4 can never be satisfied at the same time. The argument is logically equivalent to the previous argument for the topological relation $\text{Disjoint}(G_{P1}, G_{P2})$.

$\text{Overlaps}(G_{P1}, G_{P2})$ results in the classification L.

For this topological relation between G_{P1} and G_{P2} , it is not necessary to make assumptions on the dimension of G_{R2} . This is because the satisfaction of SC_3 is independent from the dimension of G_{R2} . The spatial condition Touches does not allow to restrict the extension of the resource geometry G_{R2} in such a way that the satisfaction of SC_3 can be assured. It is also impossible to make assumptions that allows to classify the spatial condition SC_4 as impossible. Therefore, this topological relation is classified as Likely. The same is true for G_{R1} : No assumption can be made that qualifies G_{R1} to assure a satisfaction of SC_4 or make it impossible.

$$\begin{aligned} SC_3 &:= \text{Within}(G_{R2}, G_{P1}) \\ SC_4 &:= \text{Touches}(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow L \\ cl_2 &:= CL_2(SC_4) \rightarrow L \\ cl &:= \text{Max}_\angle(\{L\} \cup \{L\}) \rightarrow L \end{aligned}$$

Figure 4.54: Illustrating the test constraint $\text{Overlaps}(G_{P1}, G_{P2})$

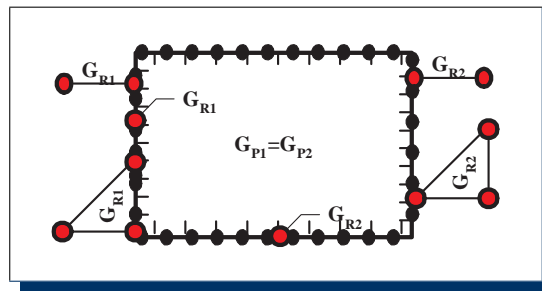
$\text{Equals}(G_{P1}, G_{P2})$ results in the classification I.

For this topological relation between G_{P1} and G_{P2} , as illustrated in figure 4.55, it is not necessary to make assumptions on the dimension of G_{R2} . This is because SC_3 and SC_4 can never be satisfied at the same time.

For G_{R1} it is essential to satisfy the spatial condition SC_1 . This restricts the exterior of G_{R1} to the interior of G_{P1} . This prevents that G_{R1} can satisfy the test condition SC_4 ($\text{Touches}(G_{R1}, G_{P2})$) at the same time. This is because G_{P1} and G_{P2} are topologically equal and therefore $\text{Within}(G_{R1}, G_{P1})$ and $\text{Touches}(G_{R1}, G_{P1})$ can never be satisfied at the same time.

In order to satisfy SC_4 , G_{R1} must share geometry points on the boundary of G_{P2} , which is equal to G_{P1} and is therefore impossible.

$$\begin{aligned} SC_3 &:= \text{Within}(G_{R2}, G_{P1}) \\ SC_4 &:= \text{Touches}(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow I \\ cl_2 &:= CL_2(SC_4) \rightarrow I \\ cl &:= \text{Max}_\perp(\{I\} \cup \{I\}) \rightarrow I \end{aligned}$$

Figure 4.55: Illustrating the test constraint $\text{Equals}(G_{P1}, G_{P2})$

Within(G_{P2}, G_{P1}) For this situation, the classification must be differentiated for $\text{dim}(G_{R2})=0$ and $\text{dim}(G_{R2})>0$.

dim(G_{R2})=0 results in the classification A.

The type of geometry G_{R2} is limited to a Point, which has per definition no exterior ($E(G_{R2}) = \emptyset$). Therefore, each resource geometry G_{R2} that satisfies the essential

spatial condition SC_2 satisfies the test condition SC_3 at the same time. This is because the valid locations of G_{R2} are limited to the boundary of G_{P2} , which is per definition inside G_{P1} : $G_{R2} \cap I(G_{P2}) \equiv \emptyset$, $G_{R2} \cap E(G_{P2}) \equiv \emptyset$ and $G_{R2} \cap B(G_{P2}) \neq \emptyset$. The satisfaction of the spatial test condition SC_4 is possible, because each resource geometry G_{R1} that satisfies SC_1 has that potential. However, it is also possible that G_{R1} does not satisfy SC_4 : All resource geometries G_{R1} that are within G_{P2} do not satisfy the spatial condition $Touches(G_{R1}, G_{P2})$.

$$\begin{aligned}
 SC_3 &:= Within(G_{R2}, G_{P1}) \\
 SC_4 &:= Touches(G_{R1}, G_{P2}) \\
 cl_1 &:= CL_1(SC_3) \rightarrow A \\
 cl_2 &:= CL_2(SC_4) \rightarrow L \\
 cl &:= Max_{\angle}(\{A\} \cup \{L\}) \rightarrow A
 \end{aligned}$$

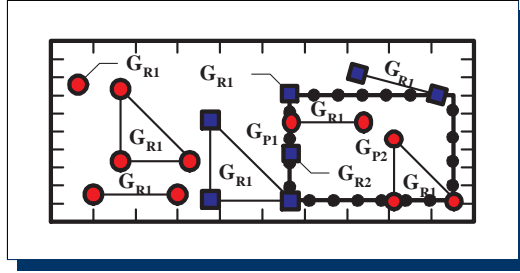


Figure 4.56: Illustrating the test constraint $Within(G_{P2}, G_{P1})$ for $\dim(G_{R2})=0$

$\dim(G_{R2}) > 0$ results in the classification L.

The relaxing of the dimension restriction for G_{R2} prevents that concrete constellations can be defined, which result in the classification Assured as illustrated in figure 4.57.

A resource geometry G_{R2} that satisfies the condition $Within(G_{R2}, G_{P1})$ can also satisfy $Touches(G_{R2}, G_{P2})$. But, a resource geometry G_{R2} that satisfies the condition $Touches(G_{R2}, G_{P2})$ must not necessarily satisfy the condition $Within(G_{R2}, G_{P1})$. This is because its exterior is not limited to the interior of G_{P1} . This results in the classification L.

$$\begin{aligned}
 SC_3 &:= Within(G_{R2}, G_{P1}) \\
 SC_4 &:= Touches(G_{R1}, G_{P2}) \\
 cl_1 &:= CL_1(SC_3) \rightarrow L \\
 cl_2 &:= CL_2(SC_4) \rightarrow L \\
 cl &:= Max_{\angle}(\{L\} \cup \{L\}) \rightarrow L
 \end{aligned}$$

$Within(G_{P1}, G_{P2})$ results in the classification I.

For this topological relation between G_{P1} and G_{P2} , as illustrated in figure 4.58, it is not necessary to make assumptions on the dimension of G_{R2} . This is because SC_3 and

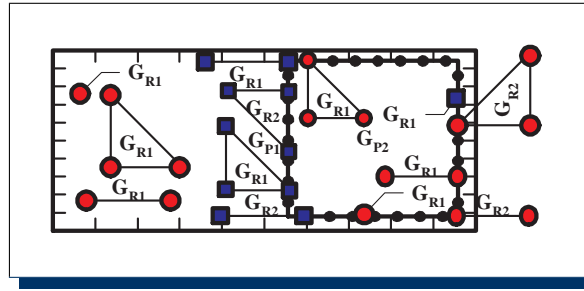


Figure 4.57: Illustrating the test constraint $Within(G_{P2}, G_{P1})$ for $\dim(G_{R2}) > 0$

SC_4 can never be satisfied at the same time: $Touches(G_{R2}, G_{P2})$ limits G_{R2} to share no interior points of G_{P2} which is essential to satisfy the condition $Within(G_{R2}, G_{P1})$.

For G_{R1} it is essential to satisfy the spatial condition SC_1 . This restricts the extension of G_{R1} to the interior of G_{P1} . This prevents that G_{R1} can satisfy the test condition SC_4 – $Touches(G_{R1}, G_{P2})$ – at the same time. This is because G_{P1} and G_{P2} do not share any geometry point, for which the condition can become true.

$$\begin{aligned}
 SC_3 &:= Within(G_{R2}, G_{P1}) \\
 SC_4 &:= Touches(G_{R1}, G_{P2}) \\
 cl_1 &:= CL_1(SC_3) \rightarrow I \\
 cl_2 &:= CL_2(SC_4) \rightarrow I \\
 cl &:= Max_{\angle}(\{I\} \cup \{I\}) \rightarrow I
 \end{aligned}$$

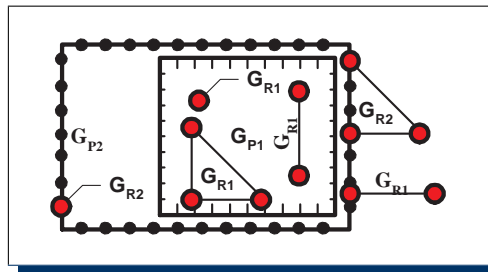


Figure 4.58: Illustrating the test constraint $Within(G_{P1}, G_{P2})$

Table 4.8 combines the results from the different spatial situations.

Classification of the Spatial Methods Touches and Touches

This situation appears if two rules are tested, where both spatial methods are Touches. Two example rules, which fall into this category are rules R1 and R2.

$$\begin{aligned}
 R1 &= \{ \dots, //aClass, C1 \} \rightarrow Permit \\
 R2 &= \{ \dots, //aClass, C2 \} \rightarrow Deny \\
 C1 &= \{ Touches(G_{R1}, G_{P1}) \} \\
 C2 &= \{ Touches(G_{R2}, G_{P2}) \}
 \end{aligned}$$

$\text{Within}(\mathbf{G}_{R1}, \mathbf{G}_{P1}) \wedge \text{Touches}(\mathbf{G}_{R2}, \mathbf{G}_{P2}) \wedge \text{Within}(\mathbf{G}_{R2}, \mathbf{G}_{P1}) \wedge \text{Touches}(\mathbf{G}_{R1}, \mathbf{G}_{P2})$			
SpatialRelation	$\text{dim}(\mathbf{G}_{R1})$	$\text{dim}(\mathbf{G}_{R2})$	classification
$\text{Disjoint}(\mathbf{G}_{P1}, \mathbf{G}_{P2})$	No restriction	No restriction	Impossible
$\text{Touches}(\mathbf{G}_{P1}, \mathbf{G}_{P2})$	No restriction	No restriction	Impossible
$\text{Equals}(\mathbf{G}_{P1}, \mathbf{G}_{P2})$	No restriction	No restriction	Impossible
$\text{Overlaps}(\mathbf{G}_{P1}, \mathbf{G}_{P2})$	No restriction	No restriction	Likely
$\text{Within}(\mathbf{G}_{P1}, \mathbf{G}_{P2})$	No restriction	No restriction	Impossible
$\text{Within}(\mathbf{G}_{P2}, \mathbf{G}_{P1})$	No restriction	0	Assured
	No restriction	1,2	Likely

Table 4.8: Worst case classification for the topological relations Within and Touches

According to the previous considerations, it is essential that SC_1 and SC_2 are satisfied:

$$SC_1 = \text{Touches}(G_{R1}, G_{P1}) \Leftrightarrow \text{True}$$

$$SC_2 = \text{Touches}(G_{R2}, G_{P2}) \Leftrightarrow \text{True}$$

The spatial conditions that determine the classification are

$$SC_3 = \text{Touches}(G_{R2}, G_{P1})$$

$$SC_4 = \text{Touches}(G_{R1}, G_{P2})$$

For some topological relations between G_{P1} and G_{P2} , the classification must be differentiated based on the dimension of the resource object geometry for G_{R1} and G_{R2} .

The essential spatial relation for G_{R2} is touching with G_{P2} . For all 0D resource geometries G_{R2} ($\text{dim}(G_{R2})=0$), the exterior is the empty set. This reduces the extension of such a geometry to its location. Therefore, all 0D resource geometries that satisfy $\text{Touches}(G_{R2}, G_{P2})$ are located on the boundary of G_{P2} . For resource geometries G_{R2} with the dimension 1D or 2D, restrictions on the exterior are impossible. The restriction of the possible extension of the geometry G_{R2} enables to approximate the satisfiability of the spatial constraint SC_3 . The extension of the resource geometry G_{R1} is restricted by its topological relation with G_{P1} , independent from its dimension. This is due to the spatial relation Touches that does not restrict the exterior of the resource geometries G_{R1} and G_{R2} a priori, as Within does it. Only, if the dimension of G_{R1} and G_{R2} is zero, the geometry is limited to a Point and the location of G_{R1} , resp. G_{R2} is restricted to the boundary of the permission geometry G_{P1} or G_{P2} .

Disjoint(G_{P1}, G_{P2}) For this topological relation between G_{P1} and G_{P2} , it is necessary to distinguish the dimension of the resource geometries.

$(\text{dim}(G_{R1})=0) \wedge (\text{dim}(G_{R2})=0)$ results in the classification I.

In this situation a resource geometry G_{R1} , which satisfies the topological relation $\text{Touches}(G_{R1}, G_{P1})$ can never satisfy the spatial method $\text{Touches}(G_{R1}, G_{P2})$ at the same time. This is because G_{R1} is restricted to locations on the boundary of G_{P1} and G_{R2} is restricted to locations on the boundary of G_{P2} , but G_{P1} does not share

any points with G_{P2} .

$$\begin{aligned}
 SC_3 &:= Touches(G_{R2}, G_{P1}) \\
 SC_4 &:= Touches(G_{R1}, G_{P2}) \\
 cl_1 &:= CL_1(SC_3) \rightarrow I \\
 cl_2 &:= CL_2(SC_4) \rightarrow I \\
 cl &:= Max_{\angle}(\{I\} \cup \{I\}) \rightarrow I
 \end{aligned}$$

$(\dim(G_{R1})=0) \vee (\dim(G_{R2})=0)$ results in the classification L.

In this situation, a resource geometry G_{R1} , which satisfies the topological relation $Touches(G_{R1}, G_{P1})$ can never satisfy the spatial method $Touches(G_{R1}, G_{P2})$ at the same time. This is because G_{R1} is restricted to locations on the boundary of G_{P1} , but G_{P1} does not share any points with G_{P2} ($Disjoint(G_{P1}, G_{P2})$).

The same logical argument is valid for G_{R2} . A resource geometry G_{R2} , which satisfies the topological relation $Touches(G_{R2}, G_{P2})$ can never satisfy the spatial method $Touches(G_{R1}, G_{P1})$ at the same time. This is because G_{R2} is restricted to locations on the boundary of G_{P2} , but G_{P2} does not share any points with G_{P1} ($Disjoint(G_{P1}, G_{P2})$).

$$\begin{aligned}
 SC_3 &:= Touches(G_{R2}, G_{P1}) \\
 SC_4 &:= Touches(G_{R1}, G_{P2}) \\
 cl_1 &:= CL_1(SC_3) \rightarrow \begin{cases} L \\ I \end{cases} \mid \dim(G_{R2}) = 0 \\
 cl_2 &:= CL_2(SC_4) \rightarrow \begin{cases} L \\ I \end{cases} \mid \dim(G_{R1}) = 0 \\
 cl &:= Max_{\angle}(\{L, I\} \cup \{L, I\}) \rightarrow L
 \end{aligned}$$

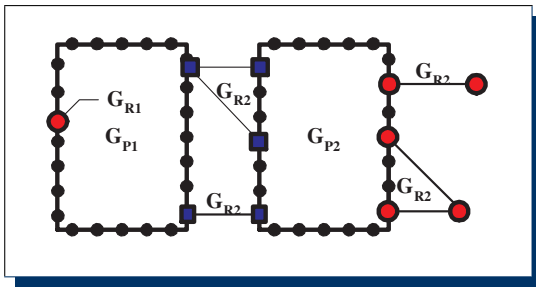


Figure 4.59: Illustrating the test constraint $Disjoint(G_{P1}, G_{P2})$ for $\dim(G_{R1})=0$ and $\dim(G_{R2})>0$

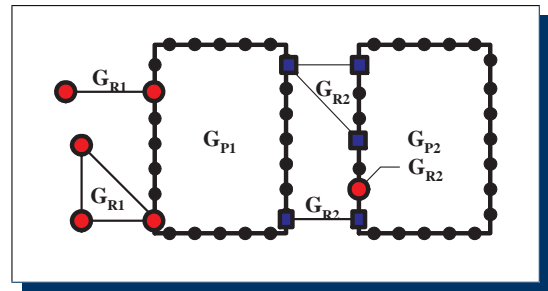


Figure 4.60: Illustrating the test constraint $Disjoint(G_{P1}, G_{P2})$ for $\dim(G_{R1})>0$ and $\dim(G_{R2})=0$

$(\dim(G_{R1})>0) \wedge (\dim(G_{R2})>0)$ results in the classification L.

The distinguishing characteristic of this situation is that neither the extension of G_{R1} nor G_{R2} has any restrictions that allow the concrete classification of SC_3 and SC_4 . The illustration of this situation is shown in figure 4.61.

Each resource geometry G_{R1} that satisfies SC_1 has the potential to satisfy SC_3 . The same is true for the resource geometry G_{R2} . However, no assumptions can be made to identify situations that can be classified as Assured or Impossible.

$$\begin{aligned} SC_3 &:= Touches(G_{R2}, G_{P1}) \\ SC_4 &:= Touches(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow L \\ cl_2 &:= CL_2(SC_4) \rightarrow L \\ cl &:= Max_{\angle}(\{L\} \cup \{L\}) \rightarrow L \end{aligned}$$

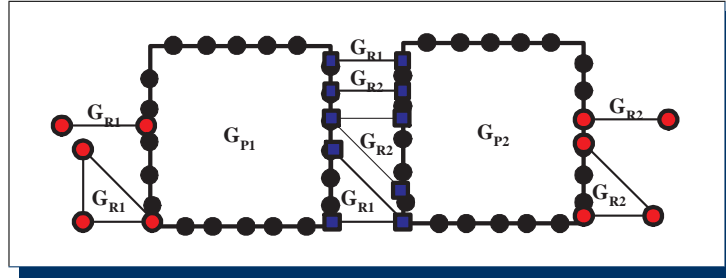


Figure 4.61: Illustrating the test constraint $Disjoint(G_{P1}, G_{P2})$ for $\dim(G_{R1}) > 0$ and $\dim(G_{R2}) > 0$

Touches(G_{P1} , G_{P2}) results in the classification L.

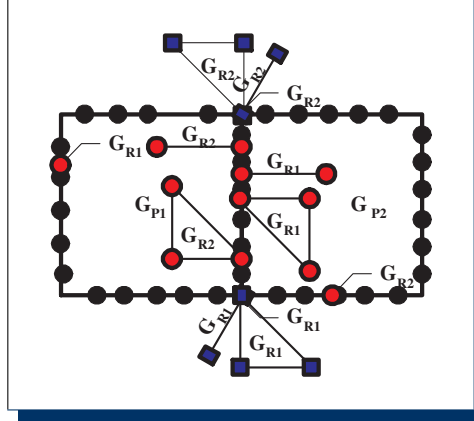
For this topological relation between G_{P1} and G_{P2} , it is not necessary to differentiate the classification for the resource geometry dimensions. No situation can be identified that results in the classification Impossible or Assured. Each resource geometry that satisfies the essential condition SC_1 , resp. SC_2 has the potential to satisfy SC_3 , resp. SC_4 . This situation is illustrated in figure 4.62.

$$\begin{aligned} SC_3 &:= Touches(G_{R2}, G_{P1}) \\ SC_4 &:= Touches(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow L \\ cl_2 &:= CL_2(SC_4) \rightarrow L \\ cl &:= Max_{\angle}(\{L\} \cup \{L\}) \rightarrow L \end{aligned}$$

Overlaps(G_{P1} , G_{P2}) results in the classification L.

The argument is logically identical to that for the topological relation $Touches(G_{P1}, G_{P2})$. Common points of G_{P1} and G_{P2} exist that allow G_{R1} and G_{R2} to satisfy the test condition. However, the satisfaction can not be assured.

$$\begin{aligned} SC_3 &:= Touches(G_{R2}, G_{P1}) \\ SC_4 &:= Touches(G_{R1}, G_{P2}) \\ cl_1 &:= CL_1(SC_3) \rightarrow L \\ cl_2 &:= CL_2(SC_4) \rightarrow L \\ cl &:= Max_{\angle}(\{L\} \cup \{L\}) \rightarrow L \end{aligned}$$

Figure 4.62: Illustrating the test constraint $\text{Touches}(G_{P1}, G_{P2})$

$\text{Within}(G_{P1}, G_{P2})$ results in the classification I.

For this topological relation between G_{P1} and G_{P2} , as illustrated in figure 4.63, it is not necessary to make assumptions on the dimension of G_{R2} . This is because SC_3 and SC_4 can never be satisfied at the same time.

A resource geometry G_{R1} that satisfies the essential condition SC_1 can never satisfy the condition SC_3 at the same time. If the dimension of G_{R1} is 0D, the location is limited to be on the boundary of G_{P1} . Because G_{P1} is within G_{P2} , G_{R1} can never be touching with G_{P2} . For a 1D resource geometry G_{R1} that touches G_{P1} , it can only satisfy the topological relations $\text{Within}(G_{R1}, G_{P2})$ or $\text{Crosses}(G_{R1}, G_{P2})$. But it can never satisfy the topological condition $\text{Touches}(G_{R1}, G_{P2})$.

A resource geometry G_{R2} that satisfies the essential condition SC_2 can never satisfy the condition SC_4 at the same time. The essential condition is $\text{Touches}(G_{R2}, G_{P2})$. Because the touches relationship prohibits that any geometry points of G_{R2} fall in the interior of G_{P2} , it is impossible that G_{R2} can satisfy the condition $\text{Touches}(G_{R2}, G_{P1})$. This is because G_{P1} is within G_{P2} .

$$SC_3 := \text{Touches}(G_{R2}, G_{P1})$$

$$SC_4 := \text{Touches}(G_{R1}, G_{P2})$$

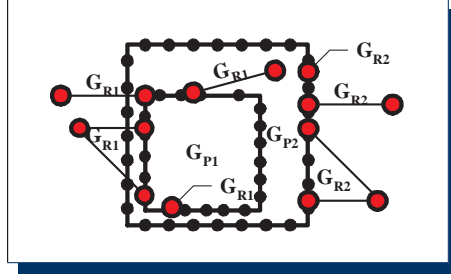
$$cl_1 := CL_1(SC_3) \rightarrow I$$

$$cl_2 := CL_2(SC_4) \rightarrow I$$

$$cl := \text{Max}_\angle(\{I\} \cup \{I\}) \rightarrow I$$

$\text{Equals}(G_{P1}, G_{P2})$ results in the classification A.

A resource geometry G_{R1} that satisfies the essential condition $\text{Touches}(G_{R1}, G_{P1})$ satisfies the condition $\text{Touches}(G_{R1}, G_{P2})$ always at the same time, because G_{P1} is equal to

Figure 4.63: Illustrating the test constraint $\text{Within}(G_{P1}, G_{P2})$

G_{P2} ($G_{P1} \equiv G_{P2}$. The same logical argument applies to a resource geometry G_{R2} .

$$\begin{aligned}
 SC_3 &:= \text{Touches}(G_{R2}, G_{P1}) \\
 SC_4 &:= \text{Touches}(G_{R1}, G_{P2}) \\
 cl_1 &:= CL_1(SC_3) \rightarrow A \\
 cl_2 &:= CL_2(SC_4) \rightarrow A \\
 cl &:= \text{Max}_{\angle}(\{A\} \cup \{A\}) \rightarrow A
 \end{aligned}$$

Table 4.9 combines the results from the different spatial situations.

$\text{Touches}(G_{R1}, G_{P1}) \wedge \text{Touches}(G_{R2}, G_{P2}) \wedge \text{Touches}(G_{R2}, G_{P1}) \wedge \text{Touches}(G_{R1}, G_{P2})$			
SpatialRelation	$\text{dim}(G_{R1})$	$\text{dim}(G_{R2})$	classification
Disjoint(G_{P1}, G_{P2})	0	0	Impossible
	$\neq 0$	$\neq 0$	Likely
Touches(G_{P1}, G_{P2})	No restriction	No restriction	Likely
Equals(G_{P1}, G_{P2})	No restriction	No restriction	Assured
Overlaps(G_{P1}, G_{P2})	No restriction	No restriction	Likely
Within(G_{P1}, G_{P2})	No restriction	No restriction	Impossible
Within(G_{P2}, G_{P1})	No restriction	No restriction	Impossible

Table 4.9: Worst case classification for spatial relations Touches and Touches

4.5 Exact Detection of Inconsistent Permissions

The duty of an access control system is to ensure an error-free enforcement of existing access restrictions to protected resources. This requires that all restrictions are encoded as permissions. For the exact detection of incorrect permissions, such as unreachable, incomplete or contrary permissions, it is essential to know all possible requests. Only if all possible subjects, operations and resources are known¹⁸, an exact detection is possible. This is the distinguishing difference to the approximate detection, as introduced earlier.

¹⁸This implies a read-only access to the resource objects.

For a resource-sided enforcement, all possible resources and operations are known. In case that only class-based restrictions are enforced, the possible resource objects are identical to the existing classes. In such a case, the resource content template, created from the GML application schema is sufficient for the detection. For the object-based and spatial restrictions, the permissions can be declared for individual resource objects. It is therefore essential to have a resource content that contains all possible permutations of resource objects. This is a valid assumption, even for a large content: The number of permutations is always finite, because the number of resource objects is finite.

For the resource-sided enforcement, it is difficult to determine the possible subjects if the permissions use arbitrary subject attribute assertions. But, if a Role Based Access Control model is used, the possible subjects are limited to the defined roles. Because RBAC is a widespread access control model, it is a good assumption for further considerations.

In such a case, where all possible requests are known, the simplest approach to detect incorrect permissions is based on requesting authorization decisions for each possible request. This approach has two important drawbacks: Complexity and statelessness.

Complexity: The approach has a high complexity, because all permutations of possible requests must be created and processed by the authentication process. Let \mathcal{S} be the set of possible subjects (hence roles), \mathcal{O} be the set of possible operations, \mathcal{RA} be the set of possible resource attributes and let \mathcal{RO} be the set of possible resource objects. Let $s \in \mathcal{S}$ be one possible subject, $o \in \mathcal{O}$ be one possible operation, $r \in \mathcal{RA}$ be one possible resource and $\mathcal{RO}_S \subseteq 2^{\mathcal{RO}}$ be a subset of the powerset¹⁹ of resource objects. Then, one possible request SOR and the set of all possible request SOR is defined as follows:

$$SOR := \{s, o, r, \mathcal{RO}_S\}, \quad s \in \mathcal{S}, \quad o \in \mathcal{O}, \quad r \in \mathcal{RA}, \quad \mathcal{RO}_S \subseteq 2^{\mathcal{RO}} \setminus \emptyset \quad (4.72)$$

$$SOR := \mathcal{S} \times \mathcal{O} \times \mathcal{RA} \times \{2^{\mathcal{RO}} \setminus \emptyset\} \quad (4.73)$$

This results in $\|\mathcal{S}\| * \|\mathcal{O}\| * \|\mathcal{RA}\| * (\|2^{\mathcal{RO}}\| - 1)$ possible requests²⁰. As an example, assuming twenty (20) subjects (or roles), five (5) operations, two (2) resource attributes and ten-thousand (10,000) resource objects, the total number of requests is $20 * 5 * 2 * \sum_{i=1}^{10,000} 1 = 10,001,000,000$.

Statelessness: The approach is stateless, because each processed request has no correlation to the current structure of permissions. Therefore, the detection as it is based on the simple processing requires to process all request each time a permission in the repository has been changed, added or deleted.

The advantage of the introduced approaches is that they are state-full and independent from the combining algorithms. They evaluate the matching expressions for the subject, operation and resource in order to determine incorrectness. Because the approach uses the permission tree to process the existing permissions and save results, it enables incremental processing, in case that permissions change or get added. This can be achieved if a policy writing tool exists that incorporates with the detection algorithm by marking the changed, added or deleted nodes.

¹⁹ $\mathcal{RO} = \{A, B, C\} \Rightarrow 2^{\mathcal{RO}} = \{\emptyset, \{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}, \{B, C\}, \{A, B, C\}\}$

²⁰One (1) is subtracted from the order of $2^{\mathcal{RO}}$, because the empty set does not represent a valid request.

4.5.1 Exact Detection of Unreachable Permissions

The exact detection of unreachable permissions has no distinguishing differences to the approximate detection. If a permission is reachable, depends on its matching and the matching of the superordinate permission. The exact detection is therefore not different from the approximate approach. Based on the criteria of the approximate detection, the reachability of permissions can be classified as reachable, potentially unreachable and unreachable.

4.5.2 Exact Detection of Contrary Permissions

The exact detection of contrary permissions is not limited to class-based permissions, as the approximate detection is. This is, because the refined matching of the rule's condition is no longer limited to the same property. The exact detection of contrary permissions relies on the full set of resource objects, which allows to select resource objects on any matching criteria.

As for the approximate detection, it is essential that at least two rules, subordinate to the same policy are reachable or potentially reachable. Also, the combining algorithm (CA_P) of the policy may not be specific-general, which explicitly defines contrary permissions.

Assuming that two rules satisfy the sufficient condition that their effects are different, then potentially reachable rules are considered to be potentially contrary and reachable rules are considered to be contrary.

Let $R1$ and $R2$ be reachable rules that are subordinate to the policy P ($R1 \angle P$) and ($R2 \angle P$). Further let E_{R1} be the effect of rule $R1$ and E_{R2} be the effect of rule $R2$. Then, the contradiction of $R1$ and $R2$ can be detected as follows:

$$R1 \text{ is contrary to } R2 \Leftrightarrow I_{R1,R2} \wedge (E_{R1} \neq E_{R2}) \wedge \\ (CA_P \neq \text{specific-general}) \mid R1 \angle P, R2 \angle P, R1 \neq R2 \quad (4.74)$$

$$\mathcal{RO}_{R1} := (XQuery(XM_{R1}, \mathcal{RO}) \cap XQuery(XM_{C1}, \mathcal{RO})) \quad (4.75)$$

$$\mathcal{RO}_{R2} := (XQuery(XM_{R2}, \mathcal{RO}) \cap XQuery(XM_{C2}, \mathcal{RO})) \quad (4.76)$$

$$I_{R1,R2} := L(SM_{R1}) \cap L(SM_{R2}) \neq \emptyset \wedge L(OM_{R1}) \cap L(OM_{R2}) \neq \emptyset \wedge \\ L(RM_{R1}) \cap L(RM_{R2}) \neq \emptyset \wedge \mathcal{RO}_{R1} \cap \mathcal{RO}_{R2} \neq \emptyset \quad (4.77)$$

For illustrating the detection of contrary permissions, the following set of one policy set, two policies and four rules is given. Rule $R11$ uses a condition to refine the matching, according to the object-based restriction.

$$PS = \{*, *, \epsilon, //*, \dots, P1, P2\}$$

$$P1 = \{Role_A, *, \epsilon, //Street, \text{specific-general}, R11, R12\}$$

$$P2 = \{*, *, \epsilon, //*, \text{deny-overrides}, R21, R22\}$$

$$R11 = \{*, r, \epsilon, //Street, C11\} \rightarrow Deny$$

$$C11 = \{\text{boolean}, ./name, \{ "3 Street A" \}\}$$

$$R12 = \{*, *, \epsilon, //Street, \epsilon\} \rightarrow Permit$$

$$R21 = \{*, r, \epsilon, //Building, \epsilon\} \rightarrow Permit$$

$$R22 = \{Role_B, *, \epsilon, //Building, \epsilon\} \rightarrow Deny$$

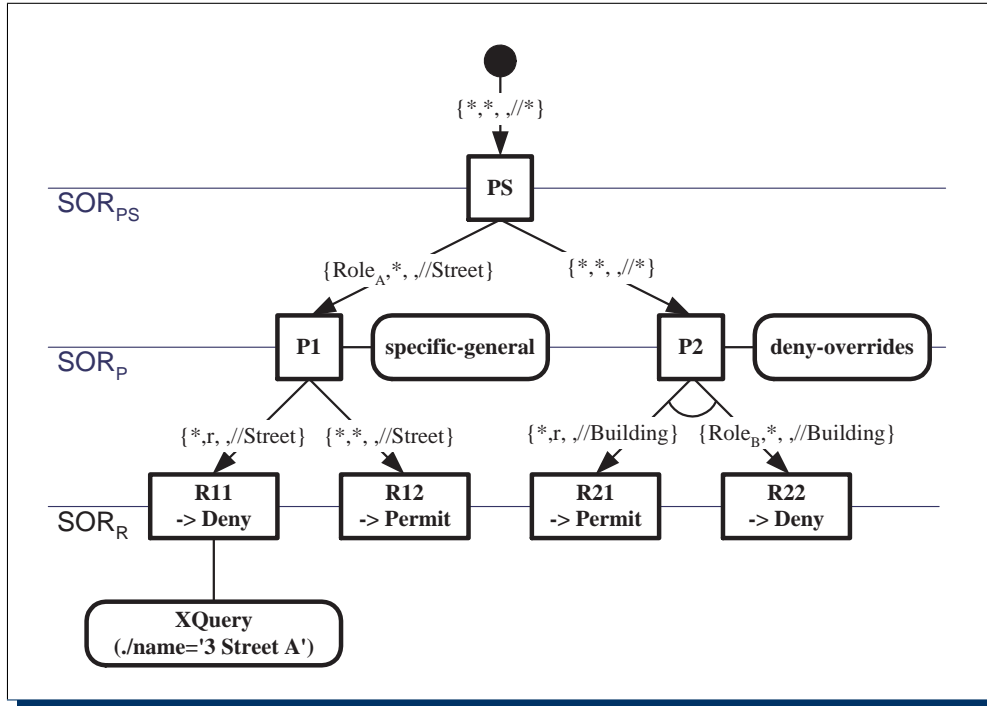


Figure 4.64: A permission tree with contrary rules

The following subjects, operations, resource attributes and resource objects are available:

$$\begin{aligned}
 \mathcal{S} &:= \{Role_A, Role_B\} \\
 \mathcal{O} &:= \{w, r\} \\
 \mathcal{RA} &:= \{\epsilon\} \\
 \mathcal{RO} &:= \{Street("3 Street A"), Street("5 Street D"), Building\}
 \end{aligned}$$

For the detection of contrary permissions, each possible request must contain all resource objects, because this represents the maximum likelihood for finding contrary permissions. This results in the following requests, represented by \mathcal{SOR} :

$$\begin{aligned}
 \mathcal{SOR} &:= \{ \{Role_A, w, \epsilon, \{Street("3 Street A"), Street("5 Street D"), Building\}\}, \\
 &\quad \{Role_A, r, \epsilon, \{Street("3 Street A"), Street("5 Street D"), Building\}\}, \\
 &\quad \{Role_B, w, \epsilon, \{Street("3 Street A"), Street("5 Street D"), Building\}\}, \\
 &\quad \{Role_B, r, \epsilon, \{Street("3 Street A"), Street("5 Street D"), Building\}\} \}
 \end{aligned}$$

Policy P1 uses the *specific-general* combining algorithm. Therefore, the subordinate rules must not be processed. Because the rules of policy P2 are reachable, the detection of contrary

rules result in the following:

$$\begin{aligned}
 E_{R21} &= \textit{Permit} \\
 E_{R22} &= \textit{Deny} \\
 CAP_2 &= \textit{deny-overrides} \neq \textit{specific-general} \\
 L(SM_{R21}) \cap L(SM_{R22}) &= \{\textit{Role}_B\} \neq \emptyset \\
 L(OM_{R21}) \cap L(OM_{R22}) &= \{r\} \neq \emptyset \\
 L(RM_{R21}) \cap L(RM_{R22}) &= \{\epsilon\} \neq \emptyset \\
 XQuery(XM_{R12}, \mathcal{RO}) \cap XQuery(XM_{C22}, \mathcal{RO}) &= \{\textit{Building}\} \neq \emptyset \\
 I_{R1,R2} &\rightarrow \textit{True} \\
 &\Rightarrow \text{Rule R21 and R22 are contrary}
 \end{aligned}$$

The rules R21 and R22 both match to the request tuple $\{\textit{Role}_B, r, \epsilon, \textit{Building}\}$, define different effects and the superordinate policy does not use the combining algorithm **specific-general**. This information can be used to annotate the tree representation, as illustrated in figure 4.65.

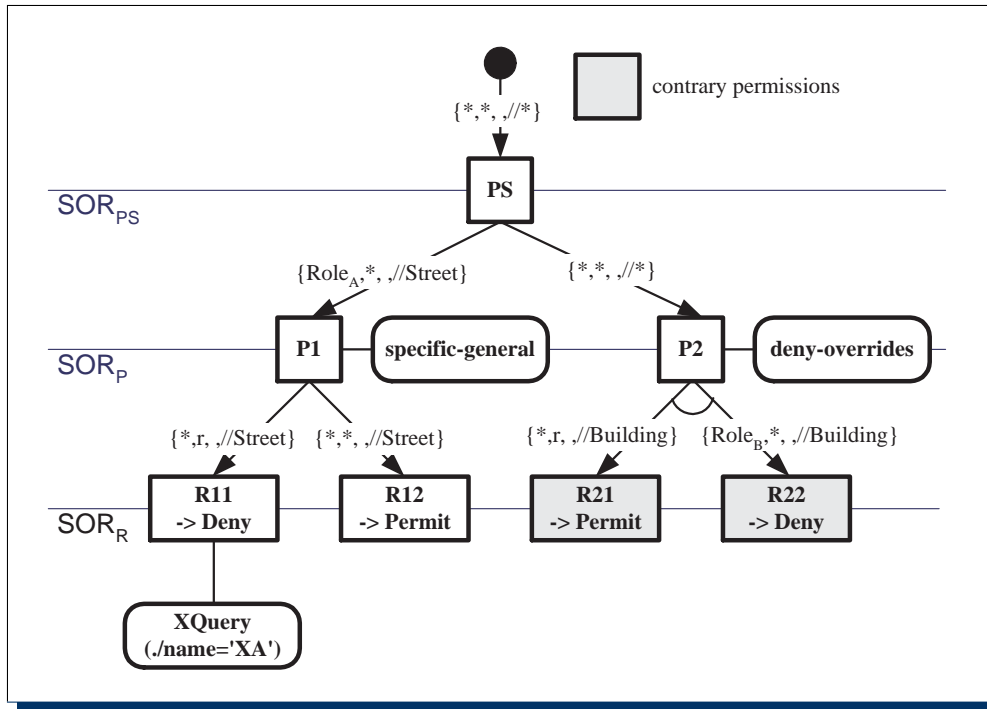


Figure 4.65: An annotated permission policy tree with contrary rules

4.5.3 Exact Detection for Complete Permissions

Using XACML, permissions are encoded by using the constructs **PolicySet**, **Policy** and **Rule** and organize them according to the XACML hierarchy (figure 3.3), which can be represented as a permission tree (listing 3.3). In addition, XACML enforces to declare the permissions,

using the all-explicit strategy. Therefore, the detection of incomplete permissions is vital for an error-free permission repository.

The detection of incomplete restrictions can be based on the matching criteria of the permission constructs and their tree hierarchy. Each PolicySet, Policy and Rule construct uses a Target element, which contains the Subject, Action²¹, Resource. The Target element therefore defines the matching of a permission.

For this work, the capabilities of the XACML Condition is limited in such respect that it can only refine the matching of a rule according to the resource objects, as required for the declaration of the object-based and spatial restrictions.

The basic idea behind this detection is that at the policy set, policy and rule level of the tree, the possible request must be processed. It is essential that this is ensured for all levels of the tree, because the driver of the authorization decision is a rule. At the top most level, a set of PolicySet constructs SOR_{PS} must provide the complete processing by correlated matching expressions. The set of subordinate policies of a policy set must also provide the complete matching. And finally, all subordinate rules of a policy must provide complete matching.

Because each PolicySet and Policy defines a matching criteria for subject, operation and resource objects that can be seen as a filter, the subordinate policies or rules must only process a subset of possible requests. This is true for the subject and operation matching expression, because only one single element can exist in the request. It is different for the resource, because in the introduced model, resource matching takes place for a resource content, encoded as GML. This requires to declare the resource matching as Xpath expressions. Because the XACML processing of a request does not modify the request, the resource content of the request remains unchanged if processed by a PolicySet, Policy, Rule or Condition. Therefore, for the exact detection of incomplete permissions, subordinate matching must only consider a subset of possible subjects and operations, but not resource objects.

However, each level of the tree defines a complete matching, if the union of the matched request tuples is identical to the possible request tuples.

An Illustrating Example

Before formalizing the exact detection for incomplete permissions, the following example may illustrate the approach. For visualization purposes, a simplified tree with one policy and two rules from figure 2.10, page 49 is used. The use of one PolicySet and multiple Policy constructs can be omitted, because their processing is identical to the processing of rules. If the procedure provides the correct result for the rule level, it can be applied to the Policy and PolicySet level in the same fashion. The simplified tree is shown in figure 4.66.

In the example, one policy P exists that has two subordinate rules, R1 and R2. Because it is also essential to know the possible roles, operations and resource objects, this can be

²¹The Action element represents the operation.

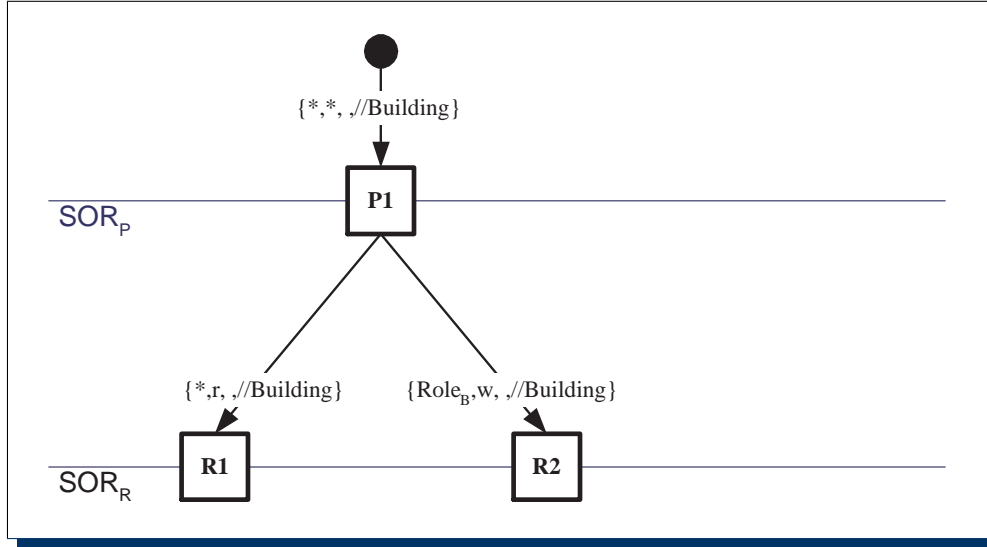


Figure 4.66: Policy tree for the detection of incompleteness

declared as follows:

$$P1 = \{*, *, \epsilon, //Building, \dots, R1, R2\}$$

$$R1 = \{*, r, \epsilon, //Building\} \rightarrow \dots$$

$$R2 = \{Role_B, w, \epsilon, //Building\} \rightarrow \dots$$

$$\mathcal{S} = \{Role_A, Role_B\}$$

$$\mathcal{O} = \{r, w\}$$

$$\mathcal{RA} = \{\epsilon\}$$

$$\mathcal{RO} = \{Building, Street\}$$

For the exact detection of complete matching for declared permissions, all available resource content objects are used. From the previous declarations and assumptions about the subjects, operations and resource objects, the following requests must be considered:

$$\mathcal{SOR} = \{\{Role_A, r, \epsilon, \{Building, Street\}\}, \{Role_A, w, \epsilon, \{Building, Street\}\}, \\ \{Role_B, r, \epsilon, \{Building, Street\}\}, \{Role_B, w, \epsilon, \{Building, Street\}\}\}$$

Policy P matches all subjects, all operations and resource attributes but only the resource object of class **Building**. Therefore, another policy is required that matches the remaining requests. This can be achieved with the rule $\{*, *, \epsilon, //Street, \epsilon\}$. According to the matching of policy P, the subordinate rules R1 and R2 must process the requests that remain. According to the matching of the rules R1 and R2, another rule is required that defines the following matching: $\{Role_A, w, \epsilon, //Building, \epsilon\}$.

Formalizing the Approach

For the formalization of the approach, two different types of Subject-Operation-Resource tuples can be identified: SOR^+ defines all tuples that are matched by a permission and SOR^- defines all tuples that are not processed by the permission. Because matching takes place on three levels of the tree, the SOR^+ tuple exists in three variations: For the PolicySet level defines SOR_{PS}^+ , for the Policy level defines SOR_P^+ and for the Rule level defines SOR_R^+ the matched tuples. In the same manner, the different variations of SOR^- can be defined: For the PolicySet level, SOR_{PS}^- defines the remaining tuples for the subordinate policies and SOR_P^- defines the remaining tuples for the subordinate rules. According to the assumptions, the Condition is correlated to the rule in such respect that it reduces the matched resource objects, as defined by the Resource matching expression. Therefore, no remaining tuples for the rule level must be determined.

PolicySet level: Let PS be one PolicySet and \mathcal{PS} be the set of available policy sets for an authorization process and $N_{PS} = \|\mathcal{PS}\|$ be the number of policy sets. Also let SM_{PS} , OM_{PS} , RM_{PS} and XM_{PS} be the matching elements of a policy set PS, then a single policy set reduces the possible requests SOR to the processed requests SOR_{PS}^+ and the remaining requests for additional required policy sets SOR_{PS}^- in the following fashion:

$$SOR := \mathcal{S} \times \mathcal{O} \times \mathcal{R} \times 2^{\mathcal{RO}} \quad (4.78)$$

$$PS := \{SM_{PS}, OM_{PS}, RM_{PS}, XM_{PS}\} \quad (4.79)$$

$$SOR_{PS}^+ := \mathcal{S}_{PS} \times \mathcal{O}_{PS} \times \mathcal{RA}_{PS} \times \mathcal{RO}_{PS} \quad (4.80)$$

$$\mathcal{S}_{PS} := \mathcal{L}(SM_{PS}) \cap \mathcal{S} \quad (4.81)$$

$$\mathcal{O}_{PS} := \mathcal{L}(OM_{PS}) \cap \mathcal{O} \quad (4.82)$$

$$\mathcal{RA}_{PS} := \mathcal{L}(RM_{PS}) \cap \mathcal{RA} \quad (4.83)$$

$$\mathcal{RO}_{PS} := XQuery(XM_{PS}, \mathcal{RO}) \quad (4.84)$$

The union of all SOR_{PS}^+ defines all request tuples that are processed by the set of policy sets. If that union is equivalent to the possible requests SOR than the set of policies declare a complete set of permissions. Let $SOR_{PS,i}^+$ be the SOR^+ tuple of the i th policy set, then the completeness can be defined as follows:

$$SOR \equiv \bigcup_i^{N_{PS}} SOR_{PS,i}^+ \Leftrightarrow \text{Completeness on PolicySet level} \quad (4.85)$$

The unmatched requests that must be processed by additional PolicySet constructs can be determined by subtracting the matched tuples from the possible tuples:

$$SOR_{PS}^- := SOR \setminus SOR_{PS}^+ \quad (4.86)$$

Policy level: Let P be one Policy and \mathcal{P} be the set of policies, subordinate to the PolicySet PS and $N_P = \|\mathcal{P}\|$ be the number of the policies. Also let SM_P , OM_P , RM_P and XM_P be the matching elements of a policy P, then a single policy reduces the possible requests

SOR_{PS}^+ to the processed requests SOR_P^+ and the remaining requests for additional required policies SOR_P^- in the following fashion:

$$P := \{SM_P, OM_P, RM_P, XM_P\} \quad (4.87)$$

$$SOR_P^+ := \mathcal{S}_P \times \mathcal{O}_P \times \mathcal{RA}_P \times \mathcal{RO}_P \quad (4.88)$$

$$\mathcal{S}_P := \mathcal{L}(SM_P) \cap \mathcal{S}_{PS}, P \angle PS \quad (4.89)$$

$$\mathcal{O}_P := \mathcal{L}(OM_P) \cap \mathcal{O}_{PS}, P \angle PS \quad (4.90)$$

$$\mathcal{RA}_P := \mathcal{L}(RM_P) \cap \mathcal{RA}_{PS}, P \angle PS \quad (4.91)$$

$$\mathcal{RO}_P := XQuery(XM_P, \mathcal{RO}) \quad (4.92)$$

The union of all SOR_P^+ defines all request tuples that are processed by the set of policies. If that union is equivalent to the possible requests SOR_{PS}^+ , then the set of policies declare a complete set of permissions. Let $SOR_{P,i}^+$ be the SOR^+ tuple of the i th policy, then the completeness can be defined as follows:

$$SOR_{PS}^+ \equiv \bigcup_i^{N_P} SOR_{P,i}^+ \Leftrightarrow \text{Completeness on Policy level} \quad (4.93)$$

The unmatched requests that must be processed by additional policies can be determined by subtracting the matched tuples from the possible tuples:

$$SOR_P^- := SOR_{PS}^+ \setminus SOR_P^+ \mid P \angle PS \quad (4.94)$$

Rule level: Let R be one Rule and \mathcal{RO} be the set of subordinate rules to a Policy P and $N_R = \|\mathcal{R}\|$ be the number of rules. Also let SM_R, OM_R, RM_R, XM_R be the matching elements of a rule R , then a single rule reduces the possible requests SOR_P^+ to the processed requests SOR_R^+ and the remaining requests for additional required rules SOR_R^- in the following fashion:

$$R := \{SM_R, OM_R, RM_R, XM_R, XM_C\} \quad (4.95)$$

$$SOR_R^+ := \mathcal{S}_R \times \mathcal{O}_R \times \mathcal{RA}_R \times \mathcal{RO}_R \quad (4.96)$$

$$\mathcal{S}_R := \mathcal{L}(SM_R) \cap \mathcal{S}_P, R \angle P \quad (4.97)$$

$$\mathcal{O}_R := \mathcal{L}(OM_R) \cap \mathcal{O}_P, R \angle P \quad (4.98)$$

$$\mathcal{RA}_R := \mathcal{L}(RM_R) \cap \mathcal{RA}_P, R \angle P \quad (4.99)$$

$$\mathcal{RO}_R := XQuery(XM_R, \mathcal{RO}) \cap XQuery(XM_C, \mathcal{RO}) \quad (4.100)$$

The union of all SOR_R^+ defines all request tuples that are processed by the set of rules. If that union is equivalent to the possible requests SOR_P^+ , then the set of rules declare a complete set of permissions. Let $SOR_{R,i}^+$ be the SOR^+ tuple of the i th rule, then the completeness can be defined as follows:

$$SOR_P^+ \equiv \bigcup_i^{N_R} SOR_{R,i}^+ \Leftrightarrow \text{Completeness on Rule level} \quad (4.101)$$

Applying the Formalism to the Illustrating Example

The example does not declare any PolicySet constructs. Therefore, all request tuples are processed by the policy P. This can be formalized by assigning the possible requests SOR to the requests, matched by the not existing PolicySet constructs SOR_{PS}^+ .

$$\begin{aligned} SOR &= \{\{Role_A, r, \epsilon, \{Building, Street\}\}, \{Role_A, w, \epsilon, \{Building, Street\}\}, \\ &\quad \{Role_B, r, \epsilon, \{Building, Street\}\}, \{Role_B, w, \epsilon, \{Building, Street\}\}\} \\ SOR_{PS} &= SOR \end{aligned}$$

According to the declaration of policy P1= $\{*, *, \epsilon, //Building, \dots, R1, R2\}$, the calculation of SOR_{PS}^+ takes place as follows:

$$\begin{aligned} P &= \{*, *, \epsilon, //Building\} \\ SOR_P^+ &= \{\{Role_A, r, \epsilon, Building\}, \{Role_A, w, \epsilon, Building\}, \\ &\quad \{Role_B, r, \epsilon, Building\}, \{Role_B, w, \epsilon, Building\}\} \\ SOR_P^- &= \{\{Role_A, r, \epsilon, \{Building, Street\}\}, \{Role_A, w, \epsilon, \{Building, Street\}\}, \\ &\quad \{Role_B, r, \epsilon, \{Building, Street\}\}, \{Role_B, w, \epsilon, \{Building, Street\}\}\} \setminus \\ &\quad \{\{Role_A, r, \epsilon, Building\}, \{Role_A, w, \epsilon, Building\}, \\ &\quad \{Role_B, r, \epsilon, Building\}, \{Role_B, w, \epsilon, Building\}\} \\ &= \{\{Role_A, r, \epsilon, \{Street\}\}, \{Role_A, w, \epsilon, \{Street\}\}, \\ &\quad \{Role_B, r, \epsilon, \{Street\}\}, \{Role_B, w, \epsilon, \{Street\}\}\} \end{aligned}$$

According to the matching for the rules R1 and R2, the detection results in the following expressions:

$$\begin{aligned} SOR_P^+ &= \{\{Role_A, r, \epsilon, Building\}, \{Role_A, w, \epsilon, Building\}, \\ &\quad \{Role_B, r, \epsilon, Building\}, \{Role_B, w, \epsilon, Building\}\} \\ R1 &= \{*, r, \epsilon, //Building, \epsilon\} \\ SOR_{R1}^+ &= \{\{Role_A, r, \epsilon, Building\}, \{Role_B, r, \epsilon, Building\}\} \\ R2 &= \{Role_B, w, \epsilon, //Building, \epsilon\} \\ SOR_{R2}^+ &= \{Role_B, w, \epsilon, Building\} \\ SOR_R^+ &= \{\{Role_A, r, \epsilon, Building\}, \{Role_B, r, \epsilon, Building\}, \{Role_B, w, \epsilon, Building\}\} \\ SOR_R^- &= \{Role_A, w, \epsilon, Building\} \end{aligned}$$

The information of unmatched requests on the policy and rule level can be used to annotate the tree representation, as illustrated in figure 4.67.

Identical Matching at the same Level: PolicySet, Policy or Rule

XACML defines a particular hierarchy of the permission constructs PolicySet, Policy and Rule. However, it does not restrict how the matching is associated for a given structure. It is therefore possible that two or more permissions (PolicySet or Policy constructs) use

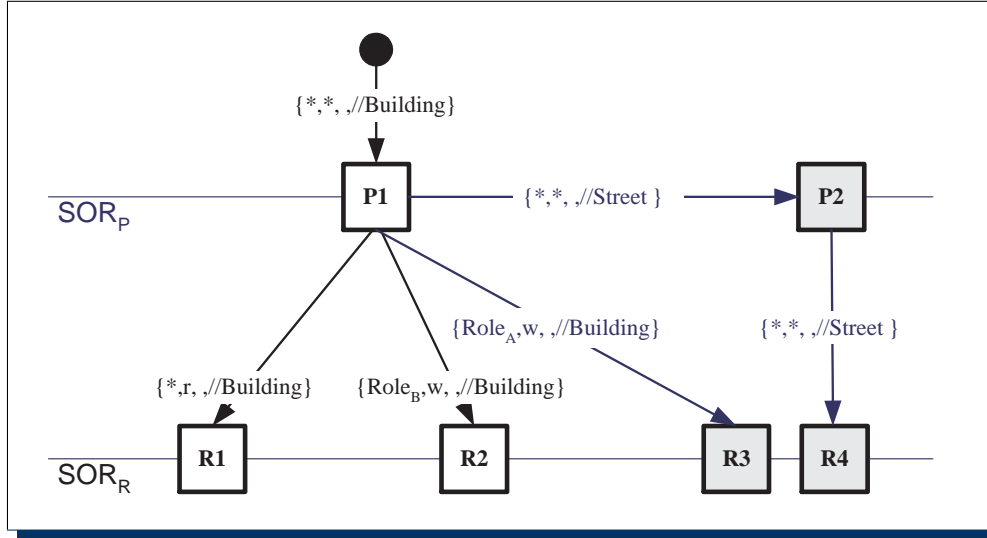


Figure 4.67: Policy tree for the detection of completeness, annotated with the request tuples that are not being matched

identical matching for the subject, operation, resource and resource content (SM, OM, RM and XM). This requires that the exact detection for complete permissions combines these permissions first, before starting the detection process for subordinate permissions (Policy or Rule constructs). This is because the introduced detection processes each permission and the subordinates separately. But in this case, the results must be merged in order to determine a complete matching.

An example permission tree, where two policies define the identical matching is shown in figure 4.68. The visualized permissions can be declared as follows²²:

$$\begin{aligned}
 PS &= \{*, *, \epsilon, //*, \dots, P1, P2, P3\} \\
 P1 &= \{Role_A, *, \epsilon //*, \dots, R11, R12\} \\
 P2 &= \{Role_A, *, \epsilon, //*, \dots, R21, R22\} \\
 P3 &= \{Role_B, *, \epsilon, //*, \dots, R3\} \\
 R11 &= \{*, r, \epsilon, //Street, \epsilon\} \rightarrow \dots \\
 R12 &= \{*, w, \epsilon, //Street, \epsilon\} \rightarrow \dots \\
 R21 &= \{*, r, \epsilon, //Building, \epsilon\} \rightarrow \dots \\
 R22 &= \{*, w, \epsilon, //Building, \epsilon\} \rightarrow \dots \\
 R3 &= \{*, *, \epsilon, //*, \epsilon\} \rightarrow \dots
 \end{aligned}$$

This set of permissions declares a complete matching, because all possible requests are processed. But, if the policies P1 and P2 and their subordinate rules are processed without combining, the result is that the rules of P1 and P2 do not define complete matching, which is wrong. For this example, the subordinate rules of P1 (R11 and R12) do not allow processing of the requests, as they are processed by the subordinate rules of P2 (R21 and R22). But

²²For the detection of completeness, the used combining algorithms and the effect of the rules is irrelevant.

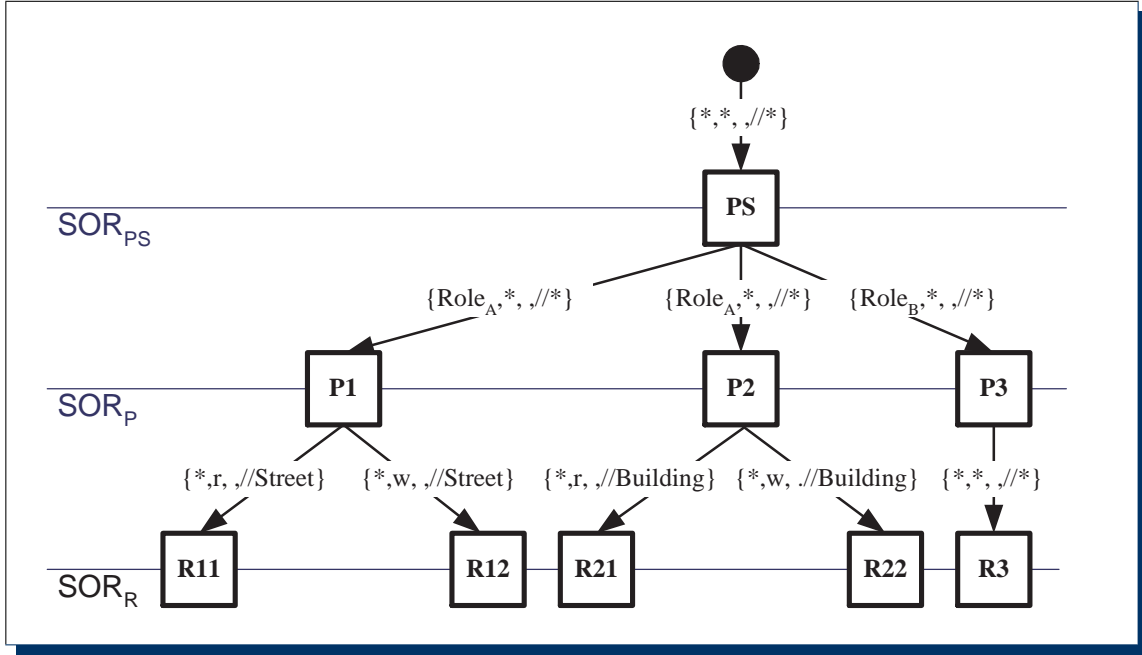


Figure 4.68: A permission tree with double matching

after merging the sub-trees of policy P1 and P2, which results in the combining of rules R12, R12, R21 and R22 to one superordinate policy, result in the correct result.

Assuming the subjects $\mathcal{S}=\{\text{Role}_A, \text{Role}_B\}$, operations $\mathcal{O}=\{r, w\}$, resource attributes $\mathcal{RA}=\{\epsilon\}$ and resource content $\mathcal{RO}=\{\text{Building}, \text{Street}\}$, the following matching can be identified: P1 and P2 matches the request tuples $\{\text{Role}_A, *, \epsilon, //*\}$ and P3 matches the request tuple $\{\text{Role}_B, *, \epsilon, //*\}$. This results to a complete matching on the policy level.

$$\begin{aligned} \text{SOR} = \{ & \{\text{Role}_A, r, \epsilon, \text{Building}\}, \{\text{Role}_A, w, \epsilon, \text{Building}\}, \\ & \{\text{Role}_A, r, \epsilon, \text{Street}\}, \{\text{Role}_A, w, \epsilon, \text{Street}\}, \\ & \{\text{Role}_B, r, \epsilon, \text{Building}\}, \{\text{Role}_B, w, \epsilon, \text{Building}\}, \\ & \{\text{Role}_B, r, \epsilon, \text{Street}\}, \{\text{Role}_B, w, \epsilon, \text{Street}\} \} \end{aligned}$$

The matching for the three policies result in the following SORs:

$$\begin{aligned} \text{SOR}_{P1}^+ &= \{ \{\text{Role}_A, r, \epsilon, \text{Building}\}, \{\text{Role}_A, w, \epsilon, \text{Building}\}, \\ & \quad \{\text{Role}_A, r, \epsilon, \text{Street}\}, \{\text{Role}_A, w, \epsilon, \text{Street}\} \} \\ \text{SOR}_{P2}^+ &= \{ \{\text{Role}_A, r, \epsilon, \text{Building}\}, \{\text{Role}_A, w, \epsilon, \text{Building}\}, \\ & \quad \{\text{Role}_A, r, \epsilon, \text{Street}\}, \{\text{Role}_A, w, \epsilon, \text{Street}\} \} \\ \text{SOR}_{P3}^+ &= \{ \{\text{Role}_B, r, \epsilon, \text{Building}\}, \{\text{Role}_B, w, \epsilon, \text{Building}\}, \\ & \quad \{\text{Role}_B, r, \epsilon, \text{Street}\}, \{\text{Role}_B, w, \epsilon, \text{Street}\} \} \end{aligned}$$

The completeness on the policy level can be verified with the union of $\text{SOR}_{P_i}^+$ tuples:

$$\text{SOR} \stackrel{?}{=} \bigcup_i^3 \text{SOR}_{P_i}^+ \rightarrow \text{True}$$

The SORs for the rules of the policies:

$$\begin{aligned}
SOR_{R11}^+ &= \{Role_A, r, \epsilon, Street\} \\
SOR_{R12}^+ &= \{Role_A, w, \epsilon, Street\} \\
SOR_{R1}^- &= \{\{Role_A, r, \epsilon, Building\}, \{Role_A, w, \epsilon, Building\}\} \\
SOR_{R21}^+ &= \{Role_A, r, \epsilon, Building\} \\
SOR_{R22}^+ &= \{Role_A, w, \epsilon, Building\} \\
SOR_{R2}^- &= \{\{Role_A, r, \epsilon, Street\}, \{Role_A, w, \epsilon, Street\}\} \\
SOR_{R3}^+ &= \{\{Role_B, r, \epsilon, Building\}, \{Role_B, w, \epsilon, Building\}, \\
&\quad \{Role_B, r, \epsilon, Street\}, \{Role_B, w, \epsilon, Street\}\}
\end{aligned}$$

This results in the following completeness checks for the permission rules, which reflect that the subordinate rules of policy P1 and P2 do not declare complete matching:

$$\begin{aligned}
SOR_{P1}^+ &\stackrel{?}{=} (SOR_{R11}^+ \cup SOR_{R12}^+) \rightarrow False \\
SOR_{P2}^+ &\stackrel{?}{=} (SOR_{R21}^+ \cup SOR_{R22}^+) \rightarrow False \\
SOR_{P3}^+ &\stackrel{?}{=} SOR_{R3}^+ \rightarrow True
\end{aligned}$$

After joining the matching from P1 and P2 to permission P12 with four rules P11, P12, P21, P22, the following SOR tuples emerge for P12:

$$\begin{aligned}
SOR_{P12}^+ &= \{SOR_{P2}^+ \cup SOR_{P2}^+\} \\
&= \{\{Role_A, r, \epsilon, Building\}, \{Role_A, w, \epsilon, Building\}, \\
&\quad \{Role_A, r, \epsilon, Street\}, \{Role_A, w, \epsilon, Street\}\} \\
SOR_{R12}^+ &= \{Role_A, r, \epsilon, Street\} \\
SOR_{R12}^+ &= \{Role_A, w, \epsilon, Street\} \\
SOR_{R21}^+ &= \{Role_A, r, \epsilon, Building\} \\
SOR_{R22}^+ &= \{Role_A, w, \epsilon, Building\}
\end{aligned}$$

The tests for incompleteness return the following results:

$$\begin{aligned}
SOR_{P12}^+ &\stackrel{?}{=} (SOR_{R11}^+ \cup SOR_{R12}^+ \cup SOR_{R21}^+ \cup SOR_{R22}^+) \rightarrow True \\
SOR_{P3}^+ &\stackrel{?}{=} SOR_{R3}^+ \rightarrow True
\end{aligned}$$

Therefore, the detection returns the correct result, after matching the policies with identical matching.

4.6 Recommending a Structured Declaration of Permissions

XACML foresees the encoding of a permission as a Rule construct, because it allows to declare an effect (*Permit* or *Deny*). The PolicySet and Policy constructs are used to structure

the existing rules in an appropriate way. This requires that each `PolicySet` and `Policy` defines a particular matching and combining algorithm that controls the processing of an outcome of subordinated constructs. According to the previous sections, it is important that the structure defines a complete set of reachable and non-contrary permissions. For the declaration and enforcement of the introduced permission kinds (class-based, object-based and spatial permissions), this section defines a recommendation about the structuring of `PolicySet`, `Policy` and `Rule` constructs. This can be seen as a guideline for policy writing that allows one or multiple policy writers to declare permissions in the same fashion.

4.6.1 Implications, using One Authorization Service

XACML foresees an infrastructure, where one Authorization Service (Policy Decision Point in XACML terminology) serves multiple Enforcement Services (Policy Enforcement Point in XACML terminology). For such an infrastructure, it is essential that the Decision Service can correlate the resources, about which restrictions are to be enforced with the appropriate set of permissions. This is essential, because each permission uses an Xpath expression to match for resource objects. If the actual structure of the resource objects (the resource content) does not fit to the Xpath expression, wrong enforcement is the result.

One obvious approach to achieve this association is to give each resource object a unique ID, which is being used by the encoded permission for matching. However, this approach has two main drawback: Performance and matching limitations. This can be explained best, when thinking about the organization of encoded permissions. This organization can be illustrated by the tree, introduced in listing 3.3.

Performance: Let's recapitulate: The process of creating an authorization decision is driven by the outcome of a rule. Enforceable rules are reached after matching took place on the superordinate policy and policy set constructs. The correlation of permissions with a particular set of resource objects takes place at the third level of matching. This approach does not empower the available hierarchy in an appropriate way. For this approach, the hierarchical structure is useless.

Matching limitations: Correlating permissions to resource objects through the unique identity has the limitation that a sole matching on other characteristics of the objects cannot take place. Each matching must use the identity AND the additional matching. This results in complex matching terms and has a drawback on performance.

Knowledge about identity: The approach to correlate permissions and resource objects by their unique identity requires, that the policy writer must know all identities of resource objects, to be protected at the time of policy writing. This is unfeasible and practically impossible.

From the previous thoughts, the correlation between resource objects and permissions cannot be achieved by a unique identity. For the introduced service infrastructure, metadata about the service can be used to direct the authorization process into a distinct path of the XACML permission hierarchy. This results in a recommendation, how to use the XACML permission hierarchy, as explained in the the next section.

4.6.2 The Coordinate Reference System

The enforcement of spatial restrictions requires specific processing, which diverge from the processing defined in XACML. In particular, the correlation between the Coordinate Reference System of the permission geometry and the resource geometry is essential for the enforcement of spatial restrictions. This is because the geometry changes if the CRS changes. This correlation can be verified by evaluation of the `srsName` attribute of GML geometries. This represents matching ‘at the last occasion’, within the `<Condition>` element. For an earlier matching at the PolicySet, Policy or Rule level, using the `<Resource>` tag of the `<Target>` element, an additional attribute can be defined:

`http://www.andreas-matheus.de/geoxacml/1.0/resource#crs-id`. One appropriate encoding of the CRS is shown in listing 4.4.

```

1 <Attribute AttributeID="http://www.andreas-matheus.de/geoxacml/1.0/resource#crs-id"
2   DataType="http://www.w3c.org/2001/XMLSchema#anyURI" >
3   <AttributeValue>EPSG:4326</AttributeValue>
4 </Attribute>
```

Listing 4.4: Encoding of an environment attribute value that defines the used CRS

For a service that supports access to geodata, encoded in different Coordinate Reference Systems, the correlation between the resource geometry CRS and the permission CRS can be achieved in two different approaches:

1. One master rule encoding exists, where the permission geometry is encoded in one supported CRS. All requests for another supported CRS require the transformation of the permission geometry, according to the resource geometry CRS.

The advantage with this approach is that only one master rule exists, which is used for enforcement. It therefore keeps the permission repository simple, because it requires just the master rule. The approach also provides the flexibility to enforce spatial permissions for GML feature collections, where the contained feature geometries are encoded in different CRSs. The drawback is that a transformation of the permission geometry is required for each authorization decision request, if the resource (request) CRS is not identical to the permission geometry’s CRS.

2. Multiple rules exist, each declaring the same spatial restriction for another supported CRS.

This approach has the advantage that no CRS transformation is required. However, it requires a more complex structuring of the permission repository, as matching for the `.../resource#crs-id` attribute must be supported. Also, it does not support the enforcement of spatial restrictions for a GML feature collection that contains geometry encodings in different CRSs.

In order to match permissions, based on the value of the `.../resource#crs-id` attribute, the Enforcement Service must add this attribute to the authorization decision request and associate the correct value. Depending on the service request parameters, the Enforcement Service may not always be able to obtain the information.

For the WMS, the request does always contain the CRS information and the service response does not represent a GML feature collection. This allows the handling of the CRS by using approach one or two.

For the WFS, not all requests must contain the CRS information. The enforcement is then based on the GML feature collection – the resource content – as it is the response of the WFS. Because the contained features can potentially be encoded in different CRSs, only the handling according to the first approach is possible.

4.6.3 Recommended Matching of PolicySet, Policy and Rule

For the introduced service infrastructure, resource objects can only be accessed by using available service operations. Each service operation is a carrier for operations, which are to be invoked on resource objects. The implication to access control is that resource objects and their structuring is correlated with service operations. This correlation can be used to organize the permissions, encoded in XACML using the PolicySet, Policy and Rule constructs.

PolicySet matching:

The XACML standard does not instruct a particular organization of the permissions. In order to empower the full capabilities of the permission hierarchy, this work recommends a particular use that fits to the service oriented infrastructure: On the top level, a set of PolicySet constructs can exist. Each PolicySet organizes the matching in such a way that it is applicable to one service operation. The assumption behind it is that each service operation provides an output that has a particular structure of resource objects. All subordinate permissions can rely on that structure and refine matching. This work uses the service identification named `service-id` and the service operation identification named `operation-id` for switching between policy sets, because it ensures natural uniqueness. The full names of the GeoXACML identifiers are

`http://www.andreas-matheus.de/geoxacml/1.0/resource#service-id` (4.102)

`http://www.andreas-matheus.de/geoxacml/1.0/resource#operation-id` (4.103)

For the declaration of permissions to protect resources on Service A and B from figure 3.1, additional information for the identifiers `#service-id` and `#operation-id` is listed in table 4.10. The appropriate combination of the identifiers allows the correlation of a particular set of permissions to a service operation as it is important for the association of the possible resource context structure and the rule's Xpath matching expression.

In addition to these resource identification attributes, XACML enforces the use of the `resource-id` attribute (`urn:oasis:names:tc:xacml:1.0:resource:resource-id`). It can be used to control matching based on the target namespace of the GML application schema. This ensures the appropriate matching if one operation of a service supports different resource content structures.

An example XACML encoding that defines a PolicySet matching for the `GetFeature` operation of the service `http://serviceA.xyz` and the GML application schema target namespace `www.in.tum.de/am` is illustrated in listing 4.5. A formalization of this PolicySet that matches to the subject `Bob` and the operation `read`, using the combining

Service	.../resource#service-id	.../resource#operation-id
ServiceA	http://serviceA.xyz	GetFeatureWithLock GetFeature Transaction Lock
ServiceB	http://serviceB.xyz	GetMap GetFeatureInfo

Table 4.10: Metadata for the Services A and B (figure 3.1)

algorithm `first-applicable` is as follows:

$$PS = \{Bob, read, \{.../resource\#service-id == http://serviceA.xyz, \\ .../resource\#operation-id == GetFeature, \\ .../resource\#resource-id == www.in.tum.de/am\}, \\ /*, first-applicable, \dots\}$$

```

1 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal" >
2   <AttributeValue
3     DataType="http://www.w3.org/2001/XMLSchema#string" >
4     www.in.tum.de/am
5   </AttributeValue>
6   <ResourceAttributeDesignator
7     AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
8     DataType="http://www.w3.org/2001/XMLSchema#string" />
9 </ResourceMatch>
10 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal" >
11   <AttributeValue
12     DataType="http://www.w3.org/2001/XMLSchema#string" >
13     http://serviceA.xyz
14   </AttributeValue>
15   <ResourceAttributeDesignator
16     AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#service-id"
17     DataType="http://www.w3.org/2001/XMLSchema#string" />
18 </ResourceMatch>
19 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal" >
20   <AttributeValue
21     DataType="http://www.w3.org/2001/XMLSchema#string" >
22     GetFeature
23   </AttributeValue>
24   <ResourceAttributeDesignator
25     AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#operation-id"
26     DataType="http://www.w3.org/2001/XMLSchema#string" />
27 </ResourceMatch>

```

Listing 4.5: Service operation selector for associating policy sets to Service A's operation `DescribeFeatureType`

Policy matching:

Using the recommended matching for a `PolicySet` organizes the permission tree in such a way that all policies, subordinate to a `PolicySet` match for the same service operation.

Therefore, the subordinate Policy constructs correspond to the same resource content structure, because the structure is – per definition – related to a service operation. For the Policy matching it is recommended to use only the subject and operation matching. This is similar to an ACL of a resource content object. Each subordinate Rule and Condition construct can then match particular resource objects. This enables to declare the class-based, object-based and spatial restrictions as rule and condition matching that must declare particular Xpath expressions for the matching.

Rule matching:

According to the matching of the superordinate Policy and PolicySet constructs, it is ensured that a Rule is correlated to a particular resource content structure as it is important for the Xpath matching of the Rule and Condition. This allows the focusing of the encoding of class-based, object-based and spatial restrictions. It is the duty of the rule matching that encodes a spatial restriction to correlate the used CRS of the permission geometry with the resource content object geometry. This can be achieved by using a rule matching as illustrated in listing 4.6. The matching ensures matching for the CRS, identified with the code *EPSG:4326*.

```

1 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
2   <AttributeValue
3     DataType="http://www.w3.org/2001/XMLSchema#string">
4     EPSG:4326
5   </AttributeValue>
6   <ResourceAttributeDesignator
7     AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#crs-id"
8     DataType="http://www.w3.org/2001/XMLSchema#nayURL"/>
9 </ResourceMatch>

```

Listing 4.6: Rule matching to ensure CRS correlation between the permission geometry and the possible resource content geometry CRS EPSG:4326

The introduced organization for the declaration of GeoXACML permissions provides a clear and manageable structure.

4.6.4 An Illustrating Example, obeying the Recommend Structure

The following example shall illustrate the recommendation for the structured encoding of permissions. For this example, the service infrastructure consists of two services (WFS and WMS) as introduced in table 4.10. For simplicity, only the GetFeature and Transaction operation of the service `http://serviceA.xyz` are considered.

Let's assume two subjects exist: Bob and Alice. The possible operations on the resource content objects for service A are read, write, insert, delete and lock as they are provided by the service. For service B, the possible operations are map and read. Then, the following restrictions are to be encoded, assuming the City Model resource content structure and objects for the WFS (`http://serviceA.xyz`) and the resource content structure from listing 5.17 for the WMS (`http://serviceB.xyz`):

PS1: `http://serviceA.xyz`, GetFeature, Operation: read

R111: Alice can read all resource content objects of class **Building**.

R121: Bob cannot read resource content objects of class **Street** according to the geometry, expressed by the spatial property `gml:LineString`, inside the permission geometry $G_P = \{\text{foo}, 0, 0, 10, 0, 10, 10, 0, 0\}$.

R122: Bob can read all resource content objects of class **Street**.

PS2: `http://serviceA.xyz`, Transaction, Operations: write, create, delete and lock

R211: Alice can write all resource content objects of class **Building**.

R212: Alice cannot write the **Building** object, identified by the address '1600 Pennsylvania Avenue NW, Washington, DC 20500'.

PS3: `http://serviceB.xyz`, GetMap, Operation: map

R311: Alice can map resource content objects of layer **Building** if the area of interest, encoded from the request parameter `BBOX` is inside the permission geometry $G_P = \{\text{foo}, -2, 0, 2, 0, 2, 6, -2, 6, -2, 0\}$

R321: Bob can map resource content objects of layer **Building** if the area of interest, encoded from the request parameter `BBOX` is inside the permission geometry $G_P = \{\text{foo}, 2, 0, 7, 0, 7, 6, 2, 6, 2, 0\}$

PS4: `http://serviceB.xyz`, GetFeatureInfo, Operation: read

R411: Alice can read resource content objects of class **Building** if the point of interest, encoded from the request attributes `X` and `Y` are inside the permission geometry $G_P = \{\text{foo}, -2, 0, 2, 0, 2, 6, -2, 6, -2, 0\}$

For the top level matching, four `PolicySet` constructs are recommended. Each `PolicySet` uses the first-applicable combining algorithm that allows the declaration of general/specific permissions. This is used on the `Policy` level for ensuring complete matching.

```
PS1 = {*, *, {#service-id == http://serviceA.xyz, #operation-id == GetFeature},
      /*, first-applicable, P11, P12, P13}
```

```
PS2 = {*, *, {#service-id == http://serviceA.xyz, #operation-id == Transaction},
      /*, first-applicable, P21, P22}
```

```
PS3 = {*, *, {#service-id == http://serviceB.xyz, #operation-id == GetMap},
      /*, first-applicable, P31, P32}
```

```
PS4 = {*, *, {#service-id == http://serviceB.xyz, #operation-id == GetFeatureInfo},
      /*, first-applicable, P41, P42}
```

According to these declarations, the following policies and rules can be defined:

$$P11 = \{Alice, r, \epsilon, //*, \text{first-applicable}, R111, R112\}$$

$$R111 = \{*, *, \epsilon, //Building, \epsilon\} \rightarrow Permit$$

$$R112 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

$$P12 = \{Bob, r, \epsilon, //*, \text{first-applicable}, R121, R122, R123\}$$

$$R121 = \{*, *, foo, //Street, C121\} \rightarrow Deny$$

$$C121 = \{./gml:LineString, Within, \{foo, 0 0, 10 0, 10 0, 0\}\}$$

$$R122 = \{*, *, \epsilon, //Street, \epsilon\} \rightarrow Permit$$

$$R123 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

$$P13 = \{*, *, \epsilon, //*, \text{first-applicable}, R131\}$$

$$R131 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

$$P21 = \{Alice, w, \epsilon, //*, \text{first-applicable}, R211, R212, R213\}$$

$$R211 = \{*, *, \epsilon, //Building, C211\} \rightarrow Deny$$

$$C211 = \{boolean, ./address, \{"1600 Pennsylvania Avenue NW, Washington, DC 20500"\}\}$$

$$R212 = \{*, *, \epsilon, //Building, \epsilon\} \rightarrow Permit$$

$$R213 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

$$P22 = \{*, *, \epsilon, //*, \text{first-applicable}, R231\}$$

$$R221 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

$$P31 = \{Alice, m, \epsilon, //*, \text{first-applicable}, R311, R312\}$$

$$P311 = \{*, *, foo, //Building, C311\} \rightarrow Permit$$

$$C311 = \{//am:WMSResourceContent/gml:boundedBy, Within, \{foo, -2 0, 2 0, 2 6, -2 6, -2 0\}\}$$

$$R312 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

$$P32 = \{Bob, m, \epsilon, //*, \text{first-applicable}, R321, R322\}$$

$$R321 = \{*, *, foo, //Building, C321\} \rightarrow Permit$$

$$C321 = \{//am:WMSResourceContent/gml:boundedBy, Within, \{foo, 2 0, 7 0, 7 6, 2 6, 2 0\}\}$$

$$R322 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

$$P41 = \{Alice, r, foo, , \text{first-applicable}, R411, 412\}$$

$$R411 = \{*, *, \epsilon, //Building, C411\} \rightarrow Permit$$

$$C411 = \{./gml : Point, Within, \{foo, -2 0, 2 0, 2 6, -2 6, -2 0\}\}$$

$$R412 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

$$P42 = \{*, *, \epsilon, //*, \text{first-applicable}, R421\}$$

$$R421 = \{*, *, \epsilon, //*, \epsilon\} \rightarrow Deny$$

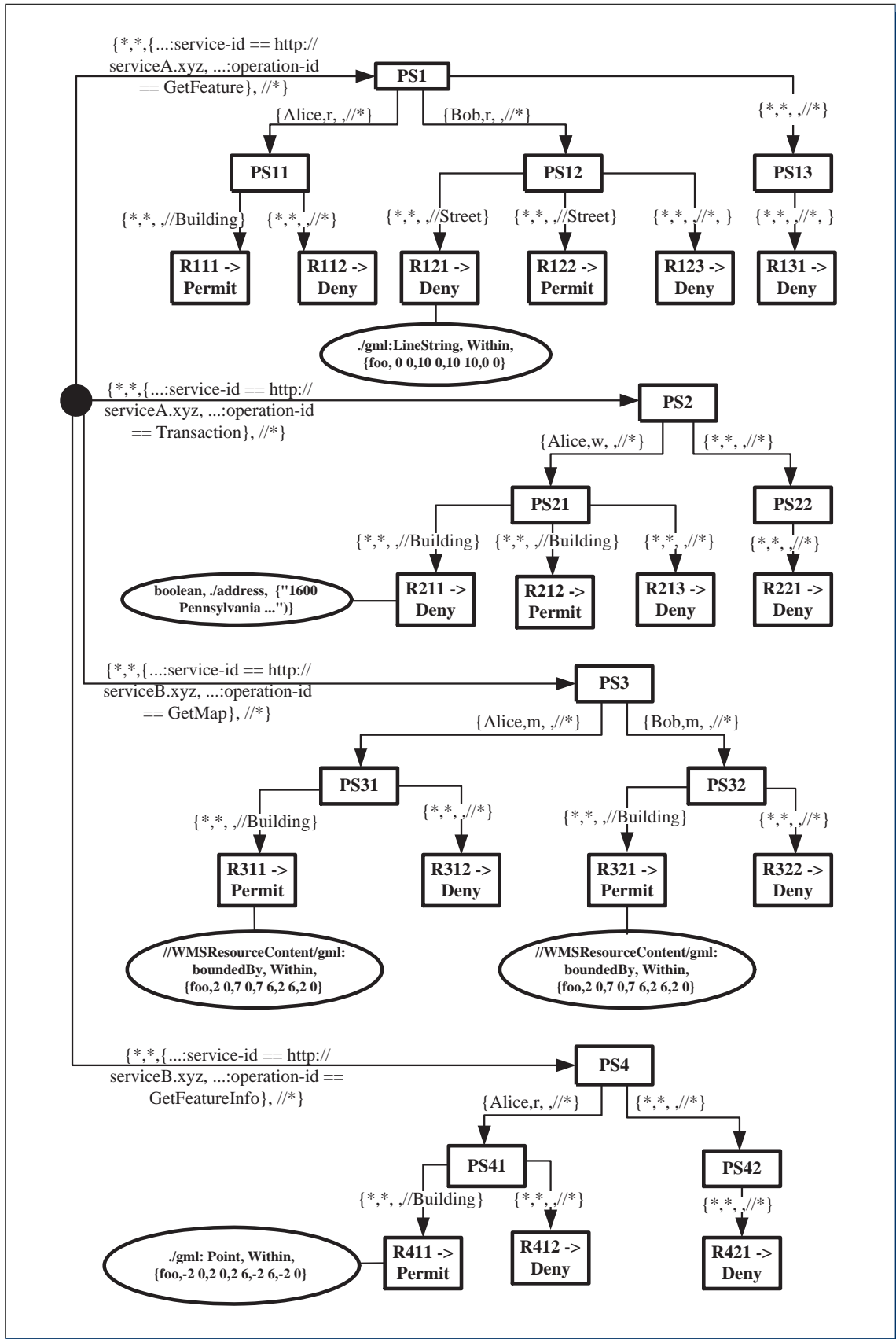


Figure 4.69: Recommended permission tree for the example policies and rules

Chapter 5

Evaluation and System Design

This chapter describes the implementation of an Authorization Service that derives authorization decision based on declared class-based, object-based and spatial restrictions, encoded in GeoXACML. Further, the implementation of two different Enforcement Services for the OGC Web Map Service and the Web Feature Service are introduced. This chapter closes with providing test cases that allow the evaluation of the implementation. This chapter starts with a discussion about the appropriate implementation infrastructure.

5.1 Architecture

This work introduces a solution for an access control system that allows the protection of geospatial information objects, accessible through a distributed services infrastructure. The question discussed in this section is how an access control system can be integrated in the existing services infrastructure.

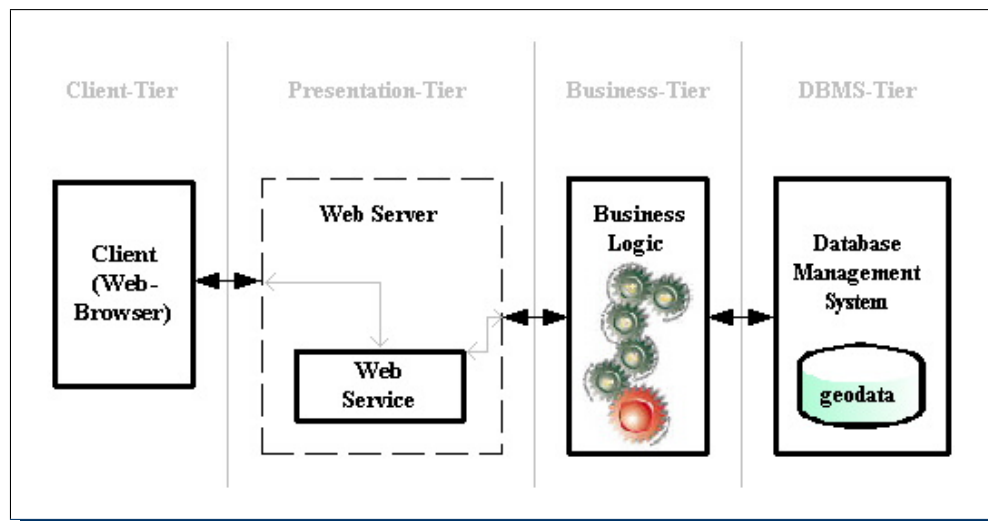


Figure 5.1: 4-Tier architecture for a service oriented architecture

The existing infrastructure of distributed services can be described as a 4-Tier architecture,

as illustrated in figure 5.1. The tiers have the following responsibility:

Tier 1: The Client-Tier provides the user interface that allows the system to sent access requests to the service and display the received response.

Tier 2: The Presentation-Tier provides the environment to host the representation logic; the Web Service. The client communicates with the Web Service by sending the request. The service communicates with the business logic that is responsible to create the response for the request.

Tier 3: The Business-Tier accepts requests, fetched by the Web Service and contacts the DBMS in order to create the response. The response is forwarded to the Web Service.

Tier 4: The DBMS-Tier provides the means for the actual data storage and allows the business logic to access the geodata by using designated operations.

In order to apply access control to such an architecture, where does it fit? Extending the functionality at Tier 1 results in a subject-sided enforcement of access restrictions, which is not intended. Extending Tier 4 does not provide transparency for the enforcement of access restrictions. Because the access restrictions apply to the service response, access control on the DBMS Tier is not transparent to the output of the business logic. This leaves Tiers 2 and 3 for the integration of the access control system.

Software developer who are in charge of the development of the business logic would probably tend to integrate the access control system into the business logic itself. This is not recommended, as outlined in [SecureWeb 2001]: *‘This practice is a major cause of applications vulnerabilities (including backdoor access). Holes and vulnerabilities in commercial and in-house web software constantly crop up and need to be fixed or plugged with upgrades or patches.’* And: *‘Web applications consist of bugs, backdoors, poor input validation, and weak state control. Businesses can’t afford to wait passively for security glitches to be discovered and fixed manually. Continually fixing code written in-house by hand is an expensive, time-consuming and never-ending task. In fact, most sites add so much new code every day that they could never hope to keep up by patching or fixing holes manually. This makes a majority of sites essentially insecure. Protecting the applications running at the heart of your online business by manually patching or upgrading is a strategy that will fail you.’* The conclusion is that a separate piece of processing logic is required that is independent from the business logic. This results in extending the capabilities of Tier 2. Here, an isolated Enforcement Service implements the interpretation of the fetched service requests, does exhaustive parameter validation, specific for the protected service. This extension to the 4-Tier infrastructure shows figure 5.2.

The bidirectional communication between the client and the service is intercepted by the Enforcement Service. Because the Enforcement Service requires access control specific metadata with the request, the client side must also be extended. Here, a specific software component called the client handler, adds required metadata to the actual request. As stated previously, this access control metadata can be encoded using SAML¹. The implementation of two different Enforcement Services for the OGC Web Map Service and the Web Feature Service analyses the request parameter, resp. uses the service response in order to create an authorization decision request.

¹A SAML encoding is not considered in this work.

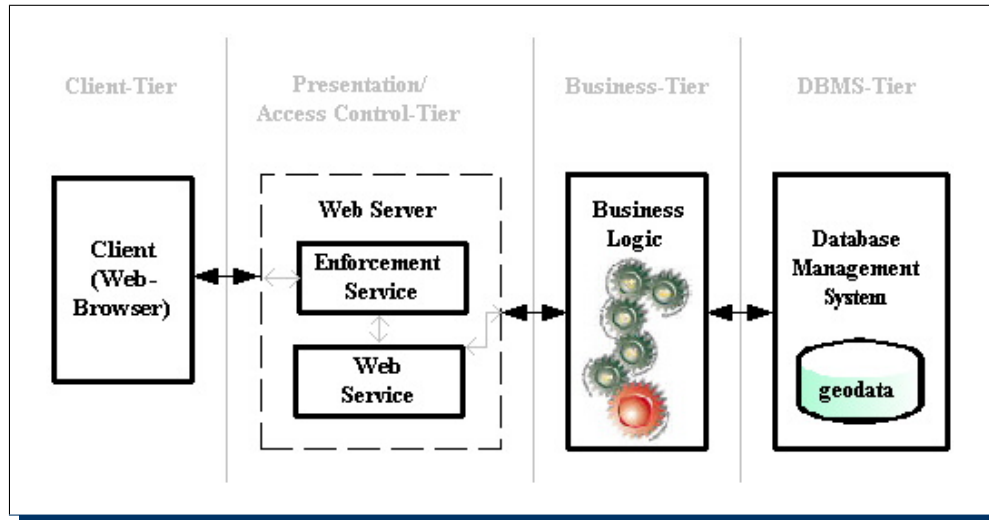


Figure 5.2: Extended 4-Tier architecture of a service oriented architecture with an Enforcement Service

5.2 GeoXACML, the Geospatial Extension to XACML

The declaration of spatial restrictions is not supported by the XACML standard. In particular, it does not support the encoding of geometry types and spatial relations. However, XACML defines extension points for the definition of the required spatial language constructs ([OASIS 2003, chapter 8]). This geospatial extension to XACML, as introduced in this work, is called GeoXACML.

5.2.1 Extending the XACML Data Types

XACML supports primitive data types, which define the type of an `AttributeValue` element. The primitive data type `String` is defined by the XACML specific URN `http://www.w3.org/2001/XMLSchema#string`. Structured XML data types can either be flattened to a string or the `CDATA` section must be used. For the flattening, the special characters are transformed by using the URL representation like `<` for `'<'` and `>` for `'>'`. For example, a GML `Point` encoding is flattened to the `String`: `"<Point srsName="foo"><coord><X>0</X><Y>0</Y></coord></Point>"`. The use of the `CDATA` section enables to represent XML encoded text inside the string value of a XML tag. It requires to mark the begin of the section with `'<![CDATA['` and the end with `']>'`. The presentation of the GML `Point` encoding, using the `CDATA` section is represented by the following: `'<![CDATA[<Point srsName="foo"><coord><X>0</X><Y>0</Y></coord></Point>]]>'`.

However, XACML supports the declaration of new structured data-types. The introduced model supports the declaration of spatial access restrictions for the defined simple geometries of section 2.2.2, page 20. For these geometries, GeoXACML declares structured data types using a GML 2.1 encoding (see 2.3.1, page 25). Table 5.1 lists the structured data types².

²This list can be extended as necessary in order to represent more complex geometry constructs.

Geometry type	XACML URN	GML 2.1 element
Point	http://www.opengis.net/gml#point	gml:Point
LineString	http://www.opengis.net/gml#linestring	gml:LineString
LinearRing	http://www.opengis.net/gml#linearring	gml:LinearRing
Polygon	http://www.opengis.net/gml#polygon	gml:Polygon
Box	http://www.opengis.net/gml#box	gml:Box

Table 5.1: GeoXACML structured data types for a subset of simple geometries using a GML 2.1 encoding

According to the defined structured data types, the GeoXACML representation of an `AttributeValue` of a 2D Point, holding the geometry information for the main entrance of the Technische Universität München, is shown in listing 5.1.

```

1 <AttributeValue DataType="http://www.opengis.net/gml#Point" >
2   <Point srsName="EPSG:4326">
3     <coord>
4       <X>40.1490</X>
5       <Y>11.5699</Y>
6     </coord>
7   </Point>
8 </AttributeValue>

```

Listing 5.1: Encoding of a 2-D Point geometry as a GeoXACML `<AttributeValue>`

5.2.2 Extending the XACML Functions

The declaration of spatial restrictions requires – among spatial data types – spatial functions. They are used in association with the spatial data types and define the spatial methods for testing a specific topological constellation between two geometries. Each function takes two parameters, which can be of any type of the spatial data types and returns `True` or `False`, according to the definition in 2.2.3, page 21. Each of the defined spatial methods is defined by a unique GeoXACML URN, as presented in table 5.2. All function URIs are identified by the same prefix: `http://www.andreas-matheus.de/geoxacml/1.0/function`.

The functions can only be used with the `<Condition>` element to refine matching of resource objects, according to their spatial properties as required by the spatial restriction. The `<Condition>` tag can reference a function by using the attribute `FunctionId` and/or an optional sub-element, named `<Function>`. This depends on the intended semantics. For example, if access is to be restricted to objects of the class `Intersection`, where the location is within a given restricted area, encoded as the permission geometry. Assuming resource objects with the structure and GML encoding from the City Model, the following permission can be expressed: Bob is not entitled to write the geospatial information objects of class `Intersection` if the location is inside the following restricted area: {foo,3 0,6 1,6 5,1 5,0 2,3

Spatial relation	URI
Disjoint	http://www.andreas-matheus.de/geoxacml/1.0/function#disjoint
Touches	http://www.andreas-matheus.de/geoxacml/1.0/function#touches
Crosses	http://www.andreas-matheus.de/geoxacml/1.0/function#crosses
Within	http://www.andreas-matheus.de/geoxacml/1.0/function#within
Overlaps	http://www.andreas-matheus.de/geoxacml/1.0/function#overlaps
Intersects	http://www.andreas-matheus.de/geoxacml/1.0/function#intersects
Equals	http://www.andreas-matheus.de/geoxacml/1.0/function#equals

Table 5.2: GeoXACML spatial functions

0}. This can be declared by using the rule and condition constructs R1 and C1:

$$R1 = \{Bob, write, //Intersection, C1\} \rightarrow Deny$$

$$C1 = \{./location, Within, \{foo, 3\ 0, 6\ 1, 6\ 5, 1\ 5, 0\ 2, 3\ 0\}\}$$

Listing 5.2 shows the equivalent GeoXACML encoding.

```

1 <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
2   <Function FunctionId="http://www.andreas-matheus.de/geoxacml/1.0/function#within"/>
3   <AttributeValue DataType="http://www.opengis.net/gml#polygon">
4     <Polygon gid="P1" srsName="foo">
5       <outerBoundaryIs>
6         <LinearRing>
7           <coord><X>3</X><Y>0</Y></coord>
8           <coord><X>6</X><Y>1</Y></coord>
9           <coord><X>6</X><Y>5</Y></coord>
10          <coord><X>1</X><Y>5</Y></coord>
11          <coord><X>0</X><Y>2</Y></coord>
12          <coord><X>3</X><Y>0</Y></coord>
13        </LinearRing>
14      </outerBoundaryIs>
15    </Polygon>
16  </AttributeValue>
17  <AttributeSelector
18    RequestContextPath="//am:CityModel/gml:featureMember/am:Intersection/am:location"
19    DataType="http://www.opengis.net/gml#point"/>
20 </Condition>

```

Listing 5.2: GeoXACML encoding of a spatial condition

The `<Condition>` tag in line 1 uses the Boolean function `any-of`, which returns `True` if at least one of the comprising expressions can be satisfied. Here, the comprising expressions are spatial functions that check the geometry value of the `Intersection` class, encoded by the element named `location`. The spatial method is referenced by the attribute `FunctionId` of the `<Function>` element in line 2. The spatial function `Within` requires two input parameters: the geometry of the permission geometry and the geometry of the resource object. The geometry of the permission geometry is encoded as an `<AttributeValue>` in lines 3-16. The type of the attribute value is encoded, using the attribute `DataType`. The geometry of the resource object, represented by the property `location` is fetched by the `<AttributeSelector>` element,

encoded from lines 17-19. The data type of the <AttributeSelector> element is declared by the attribute <DataType>. The used Xpath expression for fetching the resource object's geometry fits into the GML encoding for the City Model.

5.3 Declaration of Permissions, using GeoXACML Encoding

Section 3.1, page 55 enumerates access control requirements from an informal poll at the Intergeo 2002. Even all of these requirements are important, this work covers the class-based, object-based and spatial requirements. Therefore, this section focuses on the encoding of these restrictions only.

5.3.1 The Target Element

Common to the declaration of the different restrictions in XACML is the use of the <Target> element. It defines the matching of a PolicySet, Policy or Rule to a given Request. The <Target> tag holds the elements <Subject>, <Action> and <Resource>. The <Subject> element defines the matching for the permission Subject, the <Action> element defines the matching for the permission Operation and the <Resource> tag defines the matching for the permission Resource, resp. the resource content contained by the <ResourceContent> tag.

XACML foresees the identification of subjects through attribute value pairs. This work simplifies the identification of subjects through their identities as it is similar to the role based approach. Therefore, the <Subject> tag holds the identification for the subject, using the XACML attribute urn:oasis:names:tc:xacml:1.0:subject:subject-id. A valid XACML encoding of the Subject Bob, as used for the illustrating examples in previous sections, is shown in listing 5.3.

```

1 <Subject>
2   <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
3     <AttributeValue
4       DataType="http://www.w3.org/2001/XMLSchema#string">Bob</AttributeValue>
5     <SubjectAttributeDesignator
6       SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
7       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
8       DataType="http://www.w3.org/2001/XMLSchema#string"/>
9   </SubjectMatch>
10 </Subject>

```

Listing 5.3: XACML encoding of the subject, identified as Bob

The operation, which can be invoked on the resource is encoded by the <Action> tag. Even this work does not restrict the different operations, the examples deal with the common read and write operations. An XACML encoding of the read operation is shown in listing 5.4.

The resources, for which a Rule defines permissions, is addressed using the <Resource> tag. It holds an Xpath expression that allows the fetching of a particular set of XML elements from the resource content. Depending on the kind of the restriction, the Xpath expression is different. The next two sections present different examples for class-based and object-based restrictions.

```
1 <Action>
2   <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
3     <AttributeValue
4       DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
5     <ActionAttributeDesignator
6       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
7       DataType="http://www.w3.org/2001/XMLSchema#string"/>
8   </ActionMatch>
9 </Action>
```

Listing 5.4: XACML encoding of the operation read

5.3.2 The Declaration of Class-Based Restrictions

A restriction, which is declared for a class in the geodata's object-oriented data model must be enforced for all instances of that class, resp. for all objects. This requires an object-oriented data model, which is used for encoding of the service input and output. Based on the GML encoding, this can be achieved using a particular Xpath expression and comparison function.

The convention in this model is that a class from the underlying object-oriented data model is represented as a globally declared XML element, substitutable to `gml:Feature`. Due to the limited capabilities of the Xpath matching capabilities, a permission can be linked to a feature type in the following way: Xpath allows to query the tag's name by using the function `name` and a parameter, representing the Xpath expression to the tag. Then, a simple string compare between the returned tag name and the name of the class relates the permission to the class; hence the feature type.

In order to illustrate this in more detail, let's take an example class-based restriction, based on a part of the City Model resource objects. A class-based restriction shall protect the class `Building`. According to the structure of the GML encoded resource content for the City Model, the Xpath expression that fetches the name of the building tags is `name(//am:CityModel/gml:featureMember/am:Building)`. Because the restriction is already applicable if just one single tag exists in the XML document, it is sufficient to check for the first occurrence. This is encoded by adding the selector "[1]" to the previous Xpath expression. A simple string compare with the name of the class completes the `<Resource>` construct. Listing 5.5 shows the corresponding XACML encoding.

```
1 <Resource>
2   <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
3     <AttributeValue
4       DataType="http://www.w3.org/2001/XMLSchema#string">Building</AttributeValue>
5     <AttributeSelector
6       RequestContextPath="name(//am:CityModel/gml:featureMember/am:Building[1])"
7       DataType="http://www.w3.org/2001/XMLSchema#string"/>
8   </ResourceMatch>
9 </Resource>
```

Listing 5.5: Example XACML encoding for the class-based restriction on the class `Building` using the XACML function `string-equal`

XACML supports another solution for encoding class-based restrictions. In this case, the

<Resource> element keeps an Xpath expression that counts the number of occurrences of the class representing XML tag. For the example above, the corresponding Xpath expression is `count(//am:CityModel/gml:featureMember/am:Building)`. If the resulting number is greater than zero (0), the restriction must be enforced. Listing 5.6 shows the XACML encoding for that option.

```

1 <Resource>
2 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than" >
3 <AttributeValue
4   DataType="http://www.w3.org/2001/XMLSchema#integer">0</AttributeValue>
5 <AttributeSelector
6   RequestContextPath="count(//am:CityModel/gml:featureMember/am:Building)"
7   DataType="http://www.w3.org/2001/XMLSchema#integer" />
8 </ResourceMatch>
9 </Resource>

```

Listing 5.6: Example XACML encoding for the class-based restriction on the class `Building` using the XACML function `integer-less-than`

The <ResourceMatch> uses the XACML function `integer-less-than` and not `integer-greater-than`, because the order of the arguments corresponds to the order of the <AttributeValue> and <AttributeSelector> tags, which is predefined in XACML. Therefore, the `integer-less-than` function actually checks if at least one of the named tags exist in the resource content (0 `integer-less-than` `count(...)`).

5.3.3 The Declaration of Object-Based Restrictions

The declaration of object-based restrictions can be achieved in a similar fashion to the class-based restrictions. The difference is that designated elements are addressed by an Xpath expression that refines the fetching of elements to specific instances. These constraints are encoded as a condition, using the <Condition> element. Per convention, object-based restrictions refine the matching of resource objects based on their non-spatial characteristics and the spatial restriction according to the spatial characteristics.

For example, let's declare a condition that matches just the single information object of class `Building`, with the address "3 Street A". Using the resource content from the City Model example, the value of the address can be fetched by the following Xpath expression: `//am:CityModel/gml:featureMember/am:Building/am:address`. The corresponding XACML condition encoding is shown in listing 5.7.

```

1 <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of" >
2 <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
3 <AttributeValue
4   DataType="http://www.w3.org/2001/XMLSchema#string">3 Street A</AttributeValue>
5 <AttributeSelector
6   RequestContextPath="//am:CityModel/gml:featureMember/am:Building/am:address"
7   DataType="http://www.w3.org/2001/XMLSchema#string" />
8 </Condition>

```

Listing 5.7: Example XACML condition for selecting one specific information object

The Condition construct is comprised of a <AttributeSelector> element, which keeps the Xpath expression. The attribute selector fetches all elements from the resource content. For this result set, each entry is compared to the value of the <AttributeValue> element, using the function `string-equal`. If at least one of the comparisons result in a positive match, the condition evaluates to `True`. This is controlled by the condition function `any-of`.

In order to declare a condition that expresses the negative matching – the resource content does not contain of the `Building` object with the address “3 Street A” – the <Apply> element must be used in addition. This encoding is shown in listing 5.8.

```

1 <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:not">
2   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
3     <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
4     <AttributeValue
5       DataType="http://www.w3.org/2001/XMLSchema#string">3 Street A</AttributeValue>
6     <AttributeSelector
7       RequestContextPath="//am:CityModel/gml:featureMember/am:Building/am:address"
8       DataType="http://www.w3.org/2001/XMLSchema#string"/>
9   </Apply>
10 </Condition>

```

Listing 5.8: Example XACML condition that matches all but one specific information object

The <Apply> element takes the construct that is held by the <Condition> element in the previous example. In listing 5.8, the <Condition> element keeps the function `not` that inverses the result of the function from the <Apply> element. Based on the previous considerations, a complete object-based constraint can be declared that denies `Bob` to write the information object of class `Building`, characterized by the address `3 Street A`. Listing 5.9 shows the XACML encoding of that restriction.

5.3.4 The Declaration of Spatial Restrictions

The declaration of spatial restrictions can be achieved in a similar manner as for the declaration of object-based restrictions: The <Resource> element of the <Target> element keeps the Xpath expression to an XML tag, representing a class of the object-oriented data model. The difference to the object-based restrictions is the use of a spatial function inside the <Condition> element. For example, the frequently used spatial restriction that `Bob` can read all geospatial information objects, being instances of the class `Building` if within the boundary `{foo,3 0,6 1,6 5,1 5,0 2,3 0}`, can be encoded as a GeoXACML <Condition>, as illustrated in listing 5.10.

5.4 Enforcement of Declared Permissions

The XACML standard defines the processing of permissions, encoded as a `PolicySet`, `Policy` or `Rule` construct. However, the association of permissions to a particular `ResourceContent` structure and hence a service/operation correlation is not defined.

```

1 <Rule RuleId="diss:kapitel5:rule1" Effect="Deny">
2   <Description>Only Bob cannot write the Building object with address '3 Street A'</Description>
3   <Target>
4     <Subjects>
5       <Subject>
6         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
7           <AttributeValue
8             DataType="http://www.w3.org/2001/XMLSchema#string">Bob</AttributeValue>
9           <SubjectAttributeDesignator
10            SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
11            AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
12            DataType="http://www.w3.org/2001/XMLSchema#string"/>
13          </SubjectMatch>
14        </Subject>
15      </Subjects>
16      <Resources>
17        <AnyResource/>
18      </Resources>
19      <Actions>
20        <Action>
21          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
22            <AttributeValue
23              DataType="http://www.w3.org/2001/XMLSchema#string">write</AttributeValue>
24            <ActionAttributeDesignator
25              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
26              DataType="http://www.w3.org/2001/XMLSchema#string"/>
27          </ActionMatch>
28        </Action>
29      </Actions>
30    </Target>
31    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
32      <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
33      <AttributeValue
34        DataType="http://www.w3.org/2001/XMLSchema#string">3 Street A</AttributeValue>
35      <AttributeSelector
36        RequestContextPath="//am:CityModel/gml:featureMember/am:Building/am:address"
37        DataType="http://www.w3.org/2001/XMLSchema#string"/>
38    </Condition>
39  </Rule>

```

Listing 5.9: Example XACML restriction that denies Bob to write the building object, characterized by the address ‘3 Street A’

5.4.1 The Authorization Decision Request

The XACML authorization decision request is an XML document, valid according to the XML schema is shown in figure 3.2. According to this schema, a request contains one or more attribute value pairs that build the subject’s attribute assertion. The attribute value pairs are encoded, using the <AttributeValue> tag. A similar encoding applies to the remaining attribute value pairs for specifying the operation³, the environment and resource attributes. XACML provides a native support for the enforcement of access restrictions, based on XML encoded resources. Therefore, resources can be represented in two ways: As attribute value pairs, encoded using the <AttributeValue> tag or as an XML, resp. GML encoded document, inserted in the <ResourceContent> tag.

Applying this capability to the introduced model that is based on the service oriented

³In XACML, an operation is encoded, using the <Action> tag.


```

1 <Rule RuleId="diss:kapitel5:rule2" Effect="Permit">
2   <Target>
3     <Subjects>
4       <Subject>
5         <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
6           <AttributeValue
7             DataType="http://www.w3.org/2001/XMLSchema#string">Bob</AttributeValue>
8           <SubjectAttributeDesignator
9             SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
10            AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
11            DataType="http://www.w3.org/2001/XMLSchema#string"/>
12         </SubjectMatch>
13       </Subject>
14     </Subjects>
15     <Resources>
16       <AnyResource/>
17     </Resources>
18     <Actions>
19       <Action>
20         <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
21           <AttributeValue
22             DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
23           <ActionAttributeDesignator
24             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
25             DataType="http://www.w3.org/2001/XMLSchema#string"/>
26         </ActionMatch>
27       </Action>
28     </Actions>
29   </Target>
30   <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
31     <Function FunctionId="http://www.andreas-matheus.de/geoxacml/1.0/function#within"/>
32     <AttributeValue DataType="http://www.opengis.net/gml#polygon">
33       <Polygon gid="P2" srsName="foo">
34         <outerBoundaryIs>
35           <LinearRing>
36             <coordinates>3 0,6 1,6 5,1 5,0 2,3 0</coordinates>
37           </LinearRing>
38         </outerBoundaryIs>
39       </Polygon>
40     </AttributeValue>
41     <AttributeSelector
42       RequestContextPath="//am:CityModel/gml:featureMember/am:Building/am:shape"
43       DataType="http://www.opengis.net/gml#polygon"/>
44   </Condition>
45 </Rule>

```

Listing 5.10: Example GeoXACML Rule expressing a spatial restriction

infrastructure has implications on the functionality of the Enforcement Service. The Enforcement Service is responsible to create an authorization decision, based on available information. Depending on the capabilities of the protected service, different options exist for encoding resource information.

Service returns a GML encoded response

This is the default case: A Geo Web Service takes the request and its parameters and creates a GML encoded response. Declared restrictions must be enforced for this response as it represents the resource content.

This procedure has the advantage that the Enforcement Service must not interpret the

GML encoded response. This provides a generic handling of the response, independent from the structure of the resource content.

Service does not return a GML encoded response

In this case, an authorization decision cannot be created from the service response. The Enforcement Service must use the service request parameters to create an authorization decision. With this approach, the information about accessed resources can be encoded using the `<AttributeValue>` or the `<ResourceContent>`. Arguing with the functionality of the enforcement Service, it is not possible to make a general assumption how to encode the resource information. However, a commitment must exist between the programmed functionality of the Enforcement Service and the resource matching in a declared permission. If the policy writer assumes that a particular resource information is encoded as an `<AttributeValue>` and declares according matching, the Enforcement Service must use the `<AttributeValue>` option to encode the information.

For example, this work introduces the Enforcement Service for the Web Feature Service and the Web Map Service. Because the WMS response is not GML encoded, the authorization decision request must be created from the request. For the introduced Enforcement Service, a GML encoding of a resource content is defined, based on the structure of the geodata of the WMS. The Enforcement Service creates the resource content from the request parameters and inserts it to the authorization decision as a valid GML document, using the `<ResourceContent>` tag.

The question, how resource information must be encoded, can be answered based on the XACML capabilities for matching a permission to an authorization decision request. For resources, which are encoded by using the `<AttributeValue>` tag, only simple string matching is allowed. For resources that are encoded as an XML, resp. GML document, a permission matching can be declared in a more complex way. This matching takes place in the `<Condition>` element, which provides nesting of matching conditions using simple and complex string matching, bag and Boolean expressions as well as Xpath matching.

This results in the following recommendation: The use of explicit encoded resource information using the `<AttributeValue>` tag allows to add information for checking if a permission is applicable in the first place. For example, this work uses the service/operation identification information for switching between policy sets as it is encoded by using the `<AttributeValue>` tag. Any resources that exist in the geodata, as it belongs to a service, shall be encoded as an XML, resp. GML document and inserted in the authorization decision request by using the `<ResourceContent>` tag. This ensures the enforcement of the class-based and possibly object-based and spatial restrictions.

5.4.2 An Enforcement Service for the Web Feature Service

A Web Feature Service provides the following operations: `GetCapabilities`, `DescribeFeatureType`, `GetFeature`, `LockFeature`, `GetFeatureWithLock` and `Transaction`. Some of these operations must not be monitored by an Enforcement Service: The operation `GetCapabilities` returns the capabilities of a particular service instance. The result of this operation does not return the personalized capabilities of the actual subject, according to her/his access permissions. If this result is desired, a new operation, e.g. `GetUserCapabilities` is recommended, which requires

the identification of the user as input. Also, the `DescribeFeatureType` does not require to be monitored by an access control system, because it does not provide access to the protected resources. It returns the GML encoding of the features, according to the GML application schema. For the other operations (`GetFeature`, `LockFeature`, `Transaction` and `GetFeatureWithLock`) all introduced kinds of restrictions can be enforced.

Definition of a Resource Content Structure

For this work, it is assumed that the response of the WFS represents a valid GML encoding. In particular, the GML version 2 is assumed. This assumption results in a straight forward definition of the structure of the resource content. The GML application schema defines the XML markup of the service response, thus the resource content. This approach has the benefit that the Enforcement Service must not interpret the service response and not reformat it. This can save memory and processing time, because the response of a WFS can become Mega- or even Giga-Bytes. The Enforcement Service simply copies the fetched service response into the `<ResourceContent>` tag of the XACML authorization decision request.

Creating an Authorization Decision Request for the GetFeature Operation

Among the fetching and copying of the service response, the Enforcement Service must interpret service request parameters. For the `GetFeature` request, the following parameters are important, as they provide information required in the authorization request. The combination of used parameters and their values also directs the Enforcement Service if the creation of an authorization decision request is possible without the actual service response (see figure 4.1, page 81).

The Enforcement Service has the functionality to create the corresponding resource content and fill required subject and action related attributes. The subject related resource is the identification of the subject, which is assigned to the `<AttributeValue>`, identified with the XACML identifier `...:subject:subject-id`. The requested operation from the `GetFeature` parameter is `read`. This requires that the XACML `<AttributeValue>` with the XACML identifier `...:action:action-id` must be assigned with the value `read`. In addition, the `<AttrivuteValue>` for the service identifier `.../resource#service-id` must be set.

The following mandatory parameters provide the basic information for the authorization decision.

REQUEST: This parameter identifies the intended operation on the resources. In this case, the value to be assigned to the `...:action:action-id` is `read` and the operation identifier `.../resource#operation-id` is assigned to `GetFeature`.

TypeName: This parameter indicates the feature type to be queried. In this work, this parameter defines the class of a resource object.

The mutual exclusive parameters for the `GetFeature` operation are

FeatureID: A comma separated list of feature IDs. The parameter identifies the resource objects by their unique identifier. In combination with the `TypeName` parameter, a

valid authorization decision cannot be created. This is because no characterizing information about the resource object is known, as it can be required for object-oriented or spatial restrictions. If this parameter is used, the Enforcement Service must fetch the service response and create an authorization decision by copying the response to the `<ResourceContent>` tag.

BBOX: This parameter defines the area of interest, in which the WFS shall query for particular resource objects. In addition with this parameter, the class information from the **TypeName** parameter is available. From this information, no authorization request can be created. The argument is identical to the **FeatureID** parameter case. If this parameter is used, the Enforcement Service must fetch the service response and create an authorization decision by copying the response to the `<ResourceContent>` tag.

FILTER: This parameter carries a complex SQL-like statement, which must not be interpreted by the Enforcement Service. The logic, required to interpret the filter is part of the specific WFS instance. If this parameter is used, the Enforcement Service must fetch the service response and create an authorization decision by copying the response to the `<ResourceContent>` tag.

Creating an Authorization Decision Request for the Transaction Operation

The Transaction operation of the WFS supports three different operations on resources: **Write**, **Insert** and **Delete**. For each of these operations, the creation of the authorization decision request must be handled different.

Delete/Update For these operations, no authorization decision request can be created from the request. Therefore, the Enforcement Service must change the operation from **Delete/Update** to **Read**, invoke the service and fetch the result. That result – as it defines the resource content – can be used in the next step to create the actual authorization decision request for the **Delete/Update** operation. The XACML Attribute-Value `...:action:action-id` for the corresponding authorization request must be set to *delete/write*.

Insert The insert request carries information about the feature(s) that are to be inserted in the resource repository. Each feature is encoded in GML according to the GML application schema, which can be requested by the WFS service operation **DescribeFeatureType**. Therefore, it is possible to create an XACML encoded authorization request based on the request itself. The resource content can be created, using the request features to be inserted. An example WFS Transaction request, defining the **Insert** operation, is shown in listing 5.11. The XACML encoding for the corresponding authorization decision request is shown in listing 5.12.

Creating an Authorization Decision Request for the LockFeature Operation

The WFS **LockFeature** operation is logically connected to the **Transaction**, **GetFeatureWithLock** and **GetFeature** operations. Due to the stateless communication with the service, this operation is not tightly coupled to the other operations. This prohibits the correlation of

```
1 <?xmlversion="1.0"?>
2 <wfs:Transactionversion="1.0.0" service="WFS" xmlns="http://www.in.tum.de/am"
3   xmlns:wfs="http://www.opengis.net/wfs" xmlns:am="http://www.in.tum.de/am"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://www.opengis.net/wfs 1.0.0/WFS-transaction.xsd" >
6   <wfs:Insert>
7     <Intersection fid="IntersectionG">
8       <name>Isolated intersection</name>
9       <location srsName="foo">
10        <gml:coordinates>10 0</gml:coordinates>
11      </location>
12    </Intersection>
13  </wfs:Insert>
```

Listing 5.11: Example for a WFS Insert operation

the write, delete and insert access modes with this operation. For access control purposes, a separate operation (e.g. lock, unlock) is required, which allows the separation. The policy writer must take care that a subject, who is entitled to write, delete or insert resource objects might require to use the lock operation and must therefore have corresponding permissions.

Because this operation selects the features to be locked in the same fashion as the GetFeature operation, the same considerations apply to create an authorization decision request.

Creating an Authorization Decision Request for the GetFeatureWithLock Operation

The GetFeatureWithLock operation combines the read and the lock operations on selected resource objects. In order to grant or deny this request, the Enforcement Service can send one authorization decision request if the read-lock operation is supported. If this is not the case, the Enforcement Service must first modify the request to GetFeatureWithLock and fetch the response. For this fetched response, which represents the resource content, the enforcement service must send two authorization decision requests to the Authorization Service: One for the read, the other for the lock operation. If both results are *True*, the fetched response can be forwarded to the user. If one of the authorization decisions result in *False*, the Enforcement Service must request the service to release the lock. Even though it is not required, it is recommended that the read-lock operation is supported, because it simplifies the processing of the Enforcement Service and improves the processing performance.

5.4.3 An Enforcement Service for the Web Map Service

The Web Map Service provides three operations: GetCapabilities, GetMap and GetFeatureInfo. No access control is required for the GetCapabilities operation (see argument for WFS). The operations, which require to be monitored by an Enforcement Service are GetMap and GetFeatureInfo.

The WMS does not support an object-oriented data model. Therefore, no object-based restrictions can be enforced. However, the internal structuring of the queryable layers can be interpreted as classes, which allow the enforcement of the class-based restrictions. Because both operations (GetMap and GetFeatureInfo) query information according to geometry, spa-

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
3   xmlns:xacml-context="urn:oasis:names:tc:xacml:1.0:context"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
6   cs-xacml-schema-context-01.xsd" >
7   <Subject>
8     <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
9       DataType="http://www.w3.org/2001/XMLSchema#string" >
10      <AttributeValue>Bob</AttributeValue>
11    </Attribute>
12  </Subject>
13  <Resource>
14    <ResourceContent>
15      <WMSResourceContent xmlns="http://www.in.tum.de/am"
16        xmlns:gml="http://www.opengis.net/gml"
17        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
18        xsi:schemaLocation="http://www.in.tum.de/am WMS.xsd" >
19        <gml:boundedBy><gml:null/></gml:boundedBy>
20        <gml:featureMember>
21          <Intersection fid="IntersectionG" >
22            <name>Isolated intersection</name>
23            <location srsName="foo" >
24              <gml:coordinates>10 0</gml:coordinates>
25            </location>
26          </Intersection>
27        </gml:featureMember>
28      </WMSResourceContent>
29    </ResourceContent>
30    <Attribute
31      AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#service-id"
32      DataType="http://www.w3.org/2001/XMLSchema#anyURI" >
33      <AttributeValue>http://foo.xyz</AttributeValue>
34    </Attribute>
35    <Attribute
36      AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#operation-id"
37      DataType="http://www.w3.org/2001/XMLSchema#anyURI" >
38      <AttributeValue>Transaction</AttributeValue>
39    </Attribute>
40  </Resource>
41  <Action>
42    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
43      DataType="http://www.w3.org/2001/XMLSchema#string" >
44      <AttributeValue>insert</AttributeValue>
45    </Attribute>
46  </Action>
47 </Request>

```

Listing 5.12: XACML authorization decision request for the WFS Insert operation

tial restrictions can also be enforced. Due to the missing object-oriented data model, object-based restrictions can not be enforced. Even though the `GetFeatureInfo` interface supports the request of additional information for individual geospatial information objects. This is, because the selection is restricted to the location of the feature.

Because the service output is typically a binary image or a vector graphic, the enforcement of access restrictions must be based on the parameter values of the service request. Even for vector graphic formats, it is impossible to identify the comprising geospatial information objects through machine processing. This is due to the missing object-oriented data model.

Definition of a Resource Content Structure

The XML formatted resource content, as it is contained in an authorization decision request (`<ResourceContent>`), can be structured according to the capabilities of the service instance. Each layer of the service is represented by a global element in the resource content. Unlike to the resource content for the WFS, only one element with no characterizing attributes or properties for each layer exists.

For example, let's take the WMS from [OGC 2001-068r3, p. 4] that provides the following layers: BUILTUPA, COASTL, POLBNDL. Using a GML feature collection encoding, the XML Schema from listing 5.13 defines the resource content template structure.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <schema targetNamespace="http://www.in.tum.de/am" xmlns:gml="http://www.opengis.net/gml"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:am="http://www.in.tum.de/am" xmlns:xs="http://www.w3.org/2001/XMLSchema"
5   xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
6   <import namespace="http://www.opengis.net/gml" schemaLocation="feature.xsd"/>
7   <import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlinks.xsd"/>
8   <!-- Definition of the root element for the resource content -->
9   <element name="WMSResourceContent" type="am:WMSFeatureCollectionType"
10     substitutionGroup="gml:FeatureCollection"/>
11   <complexType name="WMSFeatureCollectionType">
12     <complexContent><extension base="gml:AbstractFeatureCollectionType"/></complexContent>
13   </complexType>
14   <!-- Definition of layers, represented by features -->
15   <element name="BUILTUPA" type="am:WMSFeatureType" substitutionGroup="gml:Feature"/>
16   <element name="COASTL" type="am:WMSFeatureType" substitutionGroup="gml:Feature"/>
17   <element name="POLBNDL" type="am:WMSFeatureType" substitutionGroup="gml:Feature"/>
18   <!-- Definition of the common WMS feature type -->
19   <complexType name="WMSFeatureType">
20     <complexContent><restriction base="gml:AbstractFeatureType">
21       <sequence minOccurs="0" maxOccurs="1">
22         <!-- The element PointOfInterest represents the location for requesting additional
23           information, using the GetFeatureInfo interface -->
24         <element name="PointOfInterest" type="gml:PointType"/>
25       </sequence>
26     </restriction></complexContent>
27   </complexType>
28 </schema>

```

Listing 5.13: XML Schema definition for a WMS resource content (WMS.xsd)

Creating an Authorization Decision Request for the GetMap Operation

The GetMap operation has different parameters, which can be used to create an authorization decision request and the comprising resource content. In this context, it is the duty of the Enforcement Service to interpret the parameter and create the corresponding resource content by filling the required subject, operation, resource and resource content related attributes. The subject related resource is the identification of the subject, which is assigned to the `<AttributeValue>`, identified with the XACML identifier `...:subject:subject-id`. The operation from the GetMap is identified as `map`. This requires that the XACML `<AttributeValue>` with the XACML identifier `...:action:action-id` must be assigned with the value `map` and the `.../resource#service-id` must be assigned with the service identification.

REQUEST: This parameter has the value `GetMap`. This denotes a map access to the resources and therefore requires to assign the `...:action:action-id` with the value `map` and the service operation identification `.../resource#operation-id` to `GetMap`.

SRS: This parameter defines the Coordinate Reference System, which has been used for the request. The values of the parameters `BBOX` depend on this parameter. The value of this parameter is associated to the `srsName` attribute of the `<Box>` element of the resource content. In addition, the CRS identifier `.../resource#crs-id` must also be assigned.

BBOX: This parameter defines the rectangular area, for which a map of resources is requested. The value of this parameter is associated to the `<coordinates>` element of the resource content, which is a sub-element of the `<Box>` element.

LAYERS: This parameter denotes the quasi classes for which information is requested. Each value of the comma-separated list can be mapped to one feature type element of the resource content.

In order to illustrate the creation of an XACML authorization decision request, let's use the complete example from [OGC 2001-068r3, p.4] and assume that the subject Bob has initiated the following request:

```

1 http://b-maps.com/map.cgi?VERSION=1.1.0&REQUEST=GetMap&
2 SRS=EPSG:4326&BBOX=-97.105,24.913,-78.794,36.358&
3 WIDTH=560&HEIGHT=350&LAYERS=BUILTUPA,COASTL,POLBNDL&
4 STYLES=0XFF8080,0X101040,BLACK&FORMAT=image/png&BGCOLOR=0xFFFFFFFF&
5 TRANSPARENT=TRUE&EXCEPTIONS=application/vnd.ogc.se_inimage

```

Listing 5.14: WMS example request for the operation `GetMap`⁴

From this WMS request, the parameter `VERSION` is important for the Enforcement Service, because it determines the syntax and semantics of the request parameters. The parameter `REQUEST=GetMap` commands the Enforcement Service to use the operation `map`. The parameter `EXCEPTIONS` directs the Enforcement Service to return information about inappropriate authorization as an image. For the given example, the image has the format that is defined by the MIME type `'application/vnd.ogc.se_inimage'`. For the enforcement of the class-based and spatial restrictions, the parameters `WIDTH`, `HEIGHT`, `STYLES`, `BGCOLOR`, `TRANSPARENT` are not important. Under these considerations, the corresponding XACML encoding of an authorization request is shown in listing 5.15.

Creating an Authorization Decision Request for the `GetFeatureInfo` Operation

The `GetFeatureInfo` operation allows to request additional metadata of information objects, which are available on a map, previously requested from the same WMS instance. For the request, the user selects the location of interest on the displayed map and selects a particular layer from which the metadata are to be queried. This request can be represented as a request

⁴WMS request from [OGC 2001-068r3, p. 4]. The layer names have been shortened for simplicity and the third value of the `BBOX` parameter has been changed to `-78.794`.


```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
3   xmlns:xacml-context="urn:oasis:names:tc:xacml:1.0:context"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
6   cs-xacml-schema-context-01.xsd">
7   <Subject>
8     <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
9       DataType="http://www.w3.org/2001/XMLSchema#string">
10      <AttributeValue>Bob</AttributeValue>
11    </Attribute>
12  </Subject>
13  <Resource>
14    <ResourceContent>
15      <WMSResourceContent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
16        xmlns="http://www.in.tum.de/am" xmlns:gml="http://www.opengis.net/gml"
17        xsi:schemaLocation="http://www.in.tum.de/am WMS.xsd">
18        <gml:boundedBy>
19          <gml:Box srsName="EPSG:4326">
20            <gml:coordinates decimal=".">-97.105 24.913,-78.794 36.358</gml:coordinates>
21          </gml:Box>
22        </gml:boundedBy>
23        <gml:featureMember><BUILTUPA/></gml:featureMember>
24        <gml:featureMember><COASTL/></gml:featureMember>
25        <gml:featureMember><POLBNDL/></gml:featureMember>
26      </WMSResourceContent>
27    </ResourceContent>
28    <Attribute AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#service-id"
29      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
30      <AttributeValue>http://b-maps.com</AttributeValue>
31    </Attribute>
32    <Attribute AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#operation-id"
33      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
34      <AttributeValue>GetMap</AttributeValue>
35    </Attribute>
36  </Resource>
37  <Action>
38    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
39      DataType="http://www.w3.org/2001/XMLSchema#string">
40      <AttributeValue>map</AttributeValue>
41    </Attribute>
42  </Action>
43 </Request>

```

Listing 5.15: XACML encoded authorization decision request for the operation GetMap⁵

to a geospatial information object, where the class and the location of the object is known. The following parameters are to be used for this request:

The Enforcement Service has the duty to create the corresponding resource content and fill required subject and action related attributes. The subject related resource is the identification of the subject, which is assigned to the `<AttributeValue>`, identified with the XACML identifier `...:subject:subject-id`. The requested operation from the `GetFeatureInfo` parameter is identified with `read`. This requires that the XACML `AttributeValue` with the XACML identifier `...:action:action-id` must be assigned with `read`. Also, the service identification must be assigned to the XACML attribute `.../resource#service-id`.

REQUEST: This parameter equals `GetFeatureInfo`. This denotes a read access to the resources and therefore requires to assign the `...:action:action-id` with the value `read` and

the service operation identification `.../resource#operation-id` to *GetFeatureInfo*.

QUERY_LAYERS: This parameter defines the quasi class for which additional information is requested. It can be handled in a similar fashion to the `LAYER` parameter for the *GetMap* request.

X,Y: X and Y define the position on the displayed map, from which the real-world's location can be calculated, using the original BBOX request parameters, which have been used to request the displayed map. Due to this calculation, the entire *GetMap* request must be included in the *GetFeatureInfo* request as it provides the required information about the CRS and BBOX.

For example, the previously used WMS supports *GetFeatureInfo* requests for the layer BUILTUPA. The following request queries information for the `QUERY_LAYERS=BUILTUPA` and the position `X=280` and `Y=175`, which represents a location Point $\approx \{\text{EPSG:4326}, -88.9, 30.6\}$. This location must and can be calculated by the Enforcement Service based on the BBOX information, the `HEIGHT` and `WIDTH` as well as on the X and Y parameter values, as shown in figure 5.3. For the calculation, it is important to note that the values of X and Y are measured from the left top corner of the displayed map⁶.

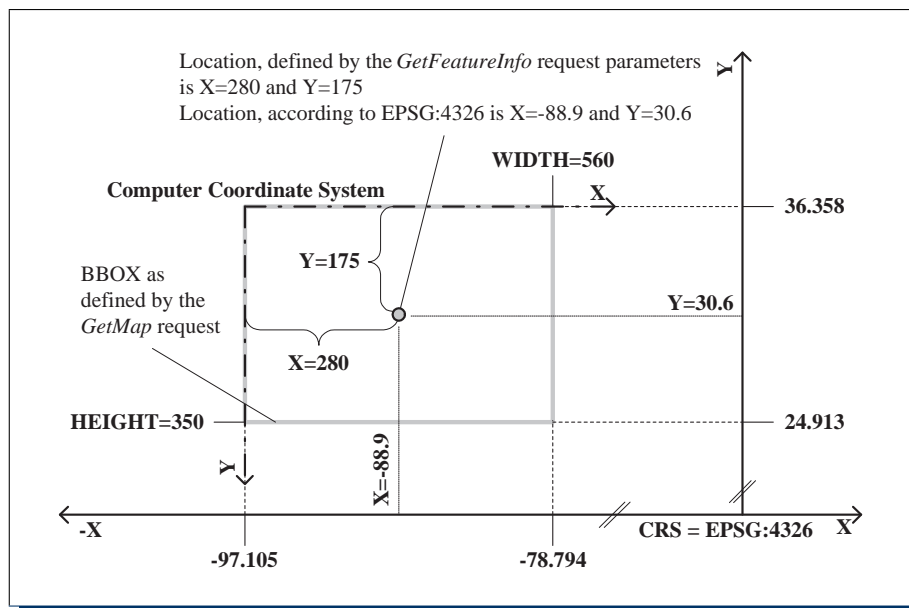


Figure 5.3: Calculation of the real-world location for the *GetFeatureInfo* request

This request can be encoded as a GeoXACML authorization decision request, as shown in listing 5.17. The bounding box is set according to the previous map request and the feature type is set to BUILTUPA. The location of the feature is set to the calculated location `{EPSG:4326,-88.9,30.6}`.

The use of the `PointOfInterest` element within the `WMSResourceContent` enables the declaration and enforcement of spatial restrictions for the information objects, fetched by the *GetFeatureInfo* request. For example, the following restriction defines that Bob can

⁶See [OGC 2001-068r3, table 9, page 40].

```

1 http://b-maps.com/feature_info.cgi?VERSION=1.1.0&REQUEST=GetMap&
2 SRS=EPSG:4326&BBOX=-97.105,24.913,-78.794,36.358&
3 WIDTH=560&HEIGHT=350&LAYERS=BUILTUPA,COASTL,POLBNDL&
4 STYLES=0xFF8080,0X101040,BLACK&FORMAT=image/png&BGCOLOR=0xFFFFFFFF&
5 TRANSPARENT=TRUE&EXCEPTIONS=application/vnd.ogc.se_inimage&
6 QUERY_LAYERS=BUILTUPA&X=280&Y=175

```

Listing 5.16: WMS example request for the operation GetFeatureInfo

read Information Objects of the quasi class BUILTUPA if within the permission area $G_P = \{\text{EPSG:4326}, -100\ 30, -70\ 40\}$:

$$R1 = \{Bob, read, \epsilon, //BUILTUPA, C1\} \rightarrow Permit$$

$$C1 = \{./PointOfInterest, Within, \{\text{EPSG:4326}, -100\ 30, -70\ 40\}\}$$

5.5 Implementation and Evaluation of a Geospatial Authorization Service

A Policy Decision Point (PDP) is a software program that accepts an XACML encoded authorization decision request and returns an XACML authorization decision, according to the XACML standard [OASIS 2003]. A Geospatial Policy Decision Point (SpatialPDP) is an extension of the PDP in that sense that it implements the functionality to allow the deriving of authorization decisions from spatial access restrictions. In particular, it implements the handling of the GeoXACML specific attribute values and spatial functions, as presented in section 3.1, page 55.

The SpatialPDP is implemented in Java, version 1.4.2, available from SUN Microsystems at <http://java.sun.com/j2se/1.4.2/index.jsp>. The implementation is based on Sun's XACML implementation, version 1.1 ([SUN 2003]) as it can be downloaded from <http://sun-xacml.sourceforge.net>. The implementation of the spatial functions is based on the Java Topology Suite (JTS), version 1.4 ([VIVID 2003]) as it can be downloaded from <http://www.vividsolutions.com/jts/jtshome.htm>.

5.5.1 Implementation of the SpatialPDP Main Class

The implementation of the main class is based on the available example that is included in the XACML implementation, as it can be downloaded from sourceforce.net. The example implementation provides the class SimplePDP, which was changed to SpatialPDP and extended in order to provide the GeoXACML capabilities.

The corresponding source code, providing the spatial capabilities is partially shown in listing 5.18. Lines 3-10 show the creation of the proxy for the spatial attribute PointAttribute. In line 14, the methods addSpatialFunctions() is called that registers the spatial functions as listed in table 5.2. In line 17, the AttributeFinder class is instantiated for supporting the handling of XACML standard attributes. The specific handling for spatial attributes is loaded by instantiating the SpatialSelectorModule in line 20.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Request xmlns="urn:oasis:names:tc:xacml:1.0:context"
3   xmlns:xacml-context="urn:oasis:names:tc:xacml:1.0:context"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context
6   cs-xacml-schema-context-01.xsd">
7   <Subject>
8     <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
9       DataType="http://www.w3.org/2001/XMLSchema#string">
10      <AttributeValue>Bob</AttributeValue>
11    </Attribute>
12  </Subject>
13  <Resource>
14    <ResourceContent>
15      <WMSResourceContent xmlns="http://www.in.tum.de/am"
16        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
17        xmlns:gml="http://www.opengis.net/gml"
18        xsi:schemaLocation="http://www.in.tum.de/am WMS.xsd">
19        <gml:boundedBy>
20          <gml:Box srsName="EPSG:4326">
21            <gml:coordinates decimal=".">-97.105 24.913,-78.794 36.358</gml:coordinates>
22          </gml:Box>
23        </gml:boundedBy>
24        <gml:featureMember>
25          <BUILTUPA>
26            <PointOfInterest srsName="EPSG:4326">
27              <gml:coordinates decimal=".">-88.9 30.6</gml:coordinates>
28            </PointOfInterest>
29          </BUILTUPA>
30        </gml:featureMember>
31      </WMSResourceContent>
32    </ResourceContent>
33    <Attribute AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#service-id"
34      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
35      <AttributeValue>http://b-maps.com</AttributeValue>
36    </Attribute>
37    <Attribute AttributeId="http://www.andreas-matheus.de/geoxacml/1.0/resource#operation-id"
38      DataType="http://www.w3.org/2001/XMLSchema#anyURI">
39      <AttributeValue>GetFeatureInfo</AttributeValue>
40    </Attribute>
41  </Resource>
42  <Action>
43    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
44      DataType="http://www.w3.org/2001/XMLSchema#string">
45      <AttributeValue>read</AttributeValue>
46    </Attribute>
47  </Action>
48 </Request>

```

Listing 5.17: GeoXACML encoding of an authorization decision for a WMS example request for the operation `GetFeatureInfo`

5.5.2 Implementation of the GeoXACML Attribute Values

The GeoXACML spatial attributes are implemented as listed in table 5.3. Each class is derived from the class `com.sun.xacml.attr.AttributeValue`, as it is illustrated in figure 5.4. Each class has a private attribute that holds the geometry. The data type of the geometry is defined by the corresponding class of the JTS library.

⁸The package for all JTS geometry classes is `com.vividsolutions.jts.geom`

```

1  ...
2  // point
3  AttributeFactory.addAttributeProxy(PointAttribute.identifier, new AttributeProxy() {
4      public AttributeValue getInstance(Node root) {
5          return PointAttribute.getInstance(root);
6      }
7      public AttributeValue getInstance(String value) {
8          return PointAttribute.getInstance(value);
9      }
10 });
11 // all other spatial attributes
12 ...
13 // add spatial functions
14 SpatialFunction.addSpatialFunctions();
15
16 // setup the AttributeFinder
17 AttributeFinder attributeFinder = new AttributeFinder();
18
19 // setup spatial attribute finder modules supporting the spatial functionality
20 SpatialSelectorModule selectorAttributeModule = new SpatialSelectorModule();
21 ...

```

Listing 5.18: Code segment for the main class SpatialPDP

Attribute URI	Class name	Geometry type ⁸
http://www.opengis.net/gml#point	PointAttribute	Point
http://www.opengis.net/gml#box	BoxAttribute	Box
http://www.opengis.net/gml#linestring	LineStringAttribute	LineString
http://www.opengis.net/gml#linearring	LinearRingAttribute	LinearRing
http://www.opengis.net/gml#polygon	PolygonAttribute	Polygon

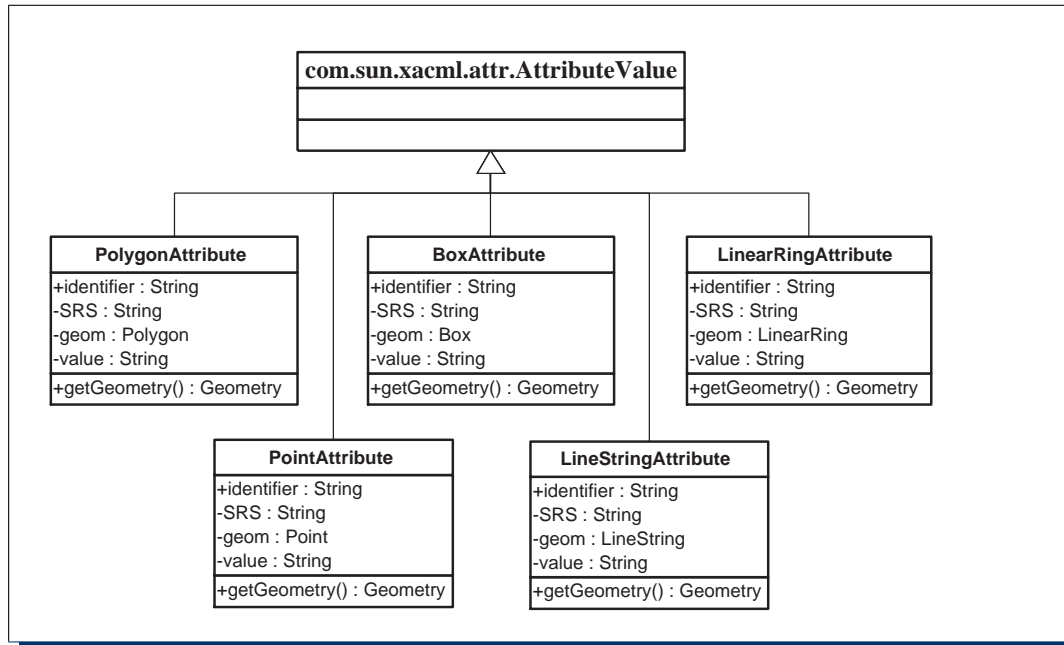
Table 5.3: URI, class name and geometry data type for the spatial attributes

Each class provides two constructors with the parameter of type `java.lang.String` and `org.w3c.dom.Node`. The first constructor transforms a string value, representing a GML geometry encoding, into the corresponding JTS geometry. The second constructor transforms the information from a given DOM node, holding a GML geometry encoding, into the corresponding JTS geometry. The geometry can be read, using the function `getGeometry()`. The return type corresponds to the represented JTS geometry type.

Each spatial attribute is added to the `SpatialPDP` by using the `addAttributeProxy(...)` method of the `AttributeFactory` class. This enables the use of the spatial attributes, using the `<AttributeValue>` tag. In order to use the spatial values by the `AttributeSelector`, another handling is required. This handling is implemented in the `SpatialSelectorModule`, which extends the XACML class `AttributeFinderModule`.

In listing 5.19, the source for the constructor of the class `PointAttribute` is shown. The constructor parameter has the data type `org.w3c.dom.Node`. The constructor fetches the `gml` namespace as it is required for querying the sub-nodes that hold the GML encoding for the `Point` geometry. This is achieved in lines 2-6. In line 8, the CRS value of the resource

⁸The object-oriented data model for the geometry classes is illustrated in figure 2.5

Figure 5.4: Class diagram of the GeoXACML attributes⁹

```

1 public PointAttribute (Element root) {
2     Namespace gml = null;
3     List additionalNamespaces = root.getAdditionalNamespaces();
4     for (int ix = 0; ix < additionalNamespaces.size(); ix++)
5         if ("gml".equals(((Namespace)additionalNamespaces.get(ix)).getPrefix()))
6             gml = (Namespace)additionalNamespaces.get(ix);
7
8     srsName = root.getAttributeValue("srsName");
9     String X = root.getChild("coord",gml).getChild("X",gml).getValue();
10    String Y = root.getChild("coord",gml).getChild("Y",gml).getValue();
11
12    try {
13        geometry = (Point) new WKTReader().read("POINT(" + X + " " + Y + ")");
14    }
15    catch (Exception e) {
16        e.printStackTrace();
17    }
18 }
  
```

Listing 5.19: Code segment of the *PointAttribute* constructor

geometry is fetched. The values of the X and Y coordinates of the `Point` are fetched in lines 9 and 10. The creation of the JTS geometry is handled in line 13. The possible exceptions are caught in lines 15-17.

5.5.3 Implementation of the SpatialSelectorModule

The `SpatialSelectorModule` class supports the selection of information from a GML document that is a sub-node to the `<ResourceContent>`. That GML document is called the resource content that has a root tag, according to the corresponding GML application schema.

The input parameters to the `findAttribute(...)` method include the Xpath expression, the `<ResourceContent>` node and the URI representation of the resulting attribute. The class `SpatialSelectorModule` replaces the original XACML class `SelectorModule`, because it supports the namespace sensitive selection of DOM nodes. This is required if the resource content carries different namespaces, as a GML feature collection does. For example, the feature collection for the City Model example uses the XML namespace `am="http://www.in.tum.de"` and `gml="http://www.opengis.net/gml"` for the GML pre-defined elements.

The functionality of the `SpatialSelectorModule` class can be described in sequential steps, using the partial source code listings 5.20 and 5.21.

1. Selecting the root node of the resource content, being a sub-node to the `<ResourceContent>` tag can be achieved by the source code from listing 5.20.

Line 1 gets a new instance of `org.jdom.input.SAXBuilder`. From the SAX-Builder, the `org.jdom.Document` representation is created in line 2. Line 3 handles the fetching of the `<ResourceContent>` tag. The path is hardcoded, as defined in the XML Schema for the authorization decision request. Line 5 handles the selection of the root tag of the resource content. With this root node, as it is the root element of the GML document, the fetching of attributes can take place. It is important to note that this root node keeps additional namespace information, which is required for further processing of the spatial attributes.

```

1 SAXBuilder builder = new SAXBuilder();
2 Document doc = builder.build(new StringBufferInputStream(ctx.getRequestRoot().toString()));
3 Xpath xPath = Xpath.newInstance("/xacml-context:Request/xacml-context:Resource/
4                               xacml-context:ResourceContent/*");
5 resourceContentRoot = (Element) xPath.selectSingleNode(doc);

```

Listing 5.20: Selection of the root node of the resource content

2. Fetching the nodes from the resource content that match the Xpath expression is handled in lines 1 and 2, source code listing 5.21. For each fetched node, it is essential to add the additional namespaces, because the processing of the sub-elements requires the `gml` namespace. This is achieved in lines 7-9. The actual creation of the attributes is handled in lines 10 and later.

5.5.4 Implementation of the GeoXACML Rule Combining Algorithms

The combining algorithms

`http://www.andreas-matheus.de/geoxacml/1.0/rule-combining-algorithm#and` and `http://www.andreas-matheus.de/geoxacml/1.0/rule-combining-algorithm#or` are implemented by extending the existing XACML class `RuleCombiningAlgorithm` from the package `com.sun.xacml.combine`. The implementing classes have the names `ORRuleAlg` and `ANDRuleAlg`. Both exist in the package `de.andreasmatheus.combine`, as shown in figure 5.5.

The implementation of both combining algorithms obeys to the logical requirement for AND or OR. Also, the implementation obeys to the XACML processing that defines the

```

1 Xpath xPath2 = Xpath.newInstance(path);
2 matches = xPath2.selectNodes(resourceContentRoot);
3 ...
4 for (int i = 0; i < matches.size(); i++) {
5   if (Element.class == matches.get(i).getClass()){
6     Element node = (Element)matches.get(i);
7     for (int ix = 0; ix < resourceContentRoot.getAdditionalNamespaces().size(); ix++)
8       node.addNamespaceDeclaration(
9         (Namespace)resourceContentRoot.getAdditionalNamespaces().get(ix));
10
11    // handle spatial attribute types
12    if (PointAttribute.identifier.equals(type.toString()))
13      list.add(new PointAttribute(node));
14    else if (BoxAttribute.identifier.equals(type.toString()))
15      list.add(new BoxAttribute(node));
16    else if
17      ...
18  }
19 }

```

Listing 5.21: Code segment for the processing of the resource content nodes, matching the Xpath expression and creation of the corresponding spatial attributes

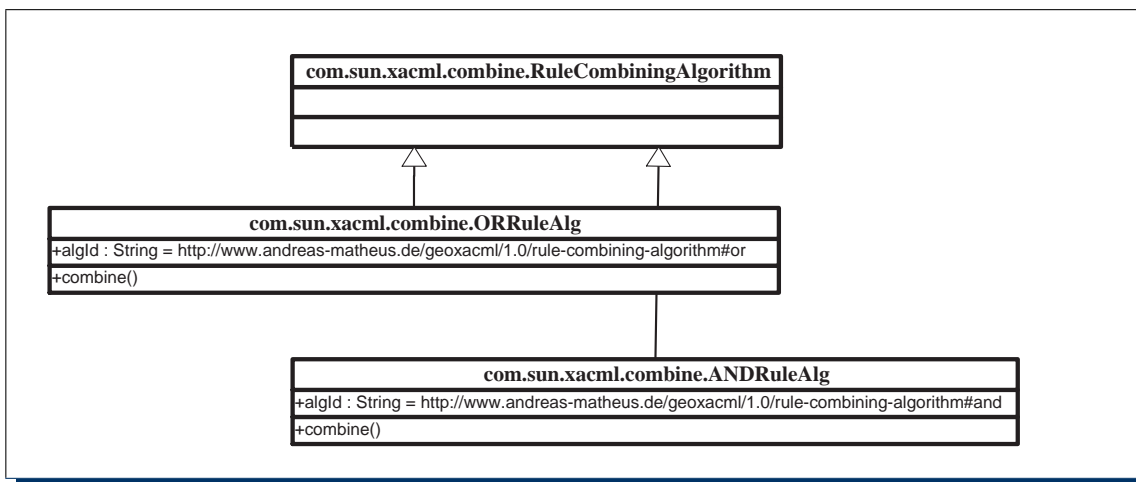


Figure 5.5: Class diagramm of the GeoXACML combining algorithms and and or

outcome Indeterminate if the processing of at least one rule results in an error and NotApplicable if no rule applies to the given request.

5.5.5 Implementation of the GeoXACML Spatial Functions

The implementation of the spatial functions is handled by the class `SpatialFunction`, which is derived from the XACML class `FunctionBase`. The static function `addSpatialFunctions()` supports the registration of the spatial functions, which is called from the class `SpatialPDP`.

Each spatial function accepts two parameters. All spatial functions in this implementation are limited to support only 2D policy geometries. The first parameter of a spatial function is restricted to the `PolygonAttribute`, which represents the permission geometry. The second

argument must be an instance of any GeoXACML spatial attribute type. Each function returns a XACML boolean attribute (`BooleanAttribute`). The value is *True* if the tested topological relation is present and *False* in all other cases¹⁰.

5.6 Evaluation of the Implemented System

The implemented Geospatial Authorization Service (SpatialPDP) allows the declaration of the class-based, object-based and spatial restrictions as introduced earlier. For each different kind of restriction, use cases are introduced that define tests for the declaration of a particular access restriction. There are two different subjects available: **Bob** and **Alice**. They can use the **read** or **write** operation on resources. The resource content is a subset of the City Model example, satisfying the structure as defined in `CityModel.xsd` (see [B.2](#), page 206).

5.6.1 Evaluation of Class-Based Restrictions

The evaluation of the implementation for the class-based restrictions is based on the existence of the following subjects, operations, resources and resource objects as well as the defined access statements:

1. $\mathcal{S} = \{\text{Alice}, \text{Bob}\}$
2. $\mathcal{O} = \{\text{read}, \text{write}\}$
3. $\mathcal{RA} = \{\epsilon\}$
4. $\mathcal{RO} = \{\text{Building}, \text{Intersection}\}$
5. Statement1 = ‘‘Bob can write information objects of class **Building**’’:
 $\{\text{Bob}, \text{write}, \epsilon, //\text{Building}, \epsilon\} \rightarrow \text{Permit}.$
6. Statement2 = ‘‘Alice cannot write information objects of class **Intersection**’’:
 $\{\text{Alice}, \text{write}, \epsilon, //\text{Intersection}, \epsilon\} \rightarrow \text{Deny}.$
7. Statement3 = ‘‘All users can read information objects of class **Building**’’:
 $\{*, \text{read}, \epsilon, //\text{Building}, \epsilon\} \rightarrow \text{Permit}.$

Assuming the all-explicit declaration strategy, the above access statements declare an incomplete set. According to the definitions of section [4.5.3](#), page 134, the following requests are not matched: $\{\text{Alice}, \text{write}, \text{Intersection}\}$, $\{\text{Alice}, \text{read}, \text{Intersection}\}$, $\{\text{Bob}, \text{read}, \text{Intersection}\}$, $\{\text{Bob}, \text{write}, \text{Intersection}\}$. The result of the test cases is listed in table [5.4](#). Please note that the unmatched requests result in the authorization decision *N/A*.

¹⁰The implementation does not evaluate the applicability of a spatial function based on the geometry dimensions, as illustrated in table [4.5](#), page 85.

Test case identification				Decision	Explanation
No.	Sub.	Op.	Res.		
1	Alice	read	Building	Permit	The resource content contains a restricted resource. Alice is allowed to read the resource (Statement3).
2	Alice	write	Building	N/A	No explicit statement is made about this resource, which results in N/A.
3	Alice	read	Intersection	N/A	No explicit statement is made about this resource, which results in N/A.
4	Alice	write	Intersection	Deny	The resource content contains a restricted resource. Alice is not allowed to write that resource (Statement2).
5	Bob	read	Building	Permit	The resource content contains a restricted resource. Bob is allowed to read the resource (Statement3).
6	Bob	write	Building	Permit	The resource content contains a restricted resource. Bob is allowed to write the resource (Statement1).
7	Bob	read	Intersection	N/A	No explicit statement is made about this resource, which results in N/A.
8	Bob	write	Intersection	N/A	No explicit statement is made about this resource, which results in N/A.

Table 5.4: Test cases for the evaluation of class-based restrictions

5.6.2 Evaluation of Object-Based Restrictions

The proper evaluation of the object-based restrictions requires that different instances of the same class are available. For the test purposes, the instances of the **Building** class from the City Model example are used. For simplification purposes, the access statements use the address of the **Building** objects to select specific instances. Assuming the same subjects and operations as with the class-based restrictions, the resources are limited to two instances of the **Building** class. The following statements describe the access control permissions:

1. $\mathcal{S} = \{\text{Alice}, \text{Bob}\}$
2. $\mathcal{O} = \{\text{read}, \text{write}\}$
3. $\mathcal{RA} = \{\epsilon\}$
4. $\mathcal{RO} = \{\text{Building}(\text{address}='3 \text{ Street A}'), \text{Building}(\text{address}='5 \text{ Street D}'), \text{Intersection}\}$
5. Statement1 = "Bob can write the information object of class **Building**, identified by address '3 Street A':
 $\{\text{Bob}, \text{write}, \epsilon, // \text{Building}, \{\text{boolean}, ./\text{address}, \{''3 \text{ Street A}''\}\}\} \rightarrow \text{Permit}.$
6. Statement2 = "Alice cannot write information object of class **Intersection**":
 $\{\text{Alice}, \text{write}, \epsilon, // \text{Intersection}, \epsilon\} \rightarrow \text{Deny}.$

7. Statement3 = ‘All users can read information objects of class Building’:
 $\{*, read, \epsilon, //Building, \epsilon\} \rightarrow Permit.$

Assuming the all-explicit declaration strategy, the above access statements declare an incomplete set. According to the definitions of section 4.5.3, page 134, the following requests are not matched: {Alice, write, Building (‘3 Street A’)}, Alice, {write, Building (‘5 Street D’)}, {Alice, read, Intersection}, {Bob, write, Building (‘5 Street D’)}, {Bob, read, Intersection}, {Bob, write, Intersection}. The authorization decision for these requests results in N/A.

Test case identification			Decision	Explanation
Alice	read	Building (‘3 Street A’)	Permit	The resource content contains a restricted resource. Alice is allowed to read the resource (Statement3)
Alice	write	Building (‘3 Street A’)	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	read	Building (‘5 Street D’)	Permit	The resource content contains a restricted resource. Alice is allowed to read the resource (Statement3)
Alice	write	Building (‘5 Street D’)	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	read	Intersection	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	write	Intersection	Deny	The resource content contains a restricted resource. Alice is not allowed to write the resource
Bob	read	Building (‘3 Street A’)	Permit	The resource content contains a restricted resource. Bob is allowed to read the resource (Statement3)
Bob	write	Building (‘3 Street A’)	Permit	The resource content contains a restricted resource. Bob is allowed to write the resource (Statement1)
Bob	read	Building (‘5 Street D’)	Permit	The resource content contains a restricted resource. Bob is allowed to read the resource (Statement3)
Bob	write	Building (‘5 Street D’)	N/A	No explicit statement is made about this resource, which results in N/A.
Bob	read	Intersection	N/A	No explicit statement is made about this resource, which results in N/A.
Bob	write	Intersection	N/A	No explicit statement is made about this resource, which results in N/A.

Table 5.5: Test cases for evaluation of object and class-based restrictions

5.6.3 Evaluation of Spatial Restrictions

The evaluation of the spatial restrictions require that specific instances of geospatial information objects are contained in the resource content. Depending on the resource geometry, the access is permitted or denied. Assuming the following subjects and access statements, exhaustive evaluation tests can be carried out by permutation of all possible constellations. The evaluation, documented in this work is limited to describe the test cases for the spatial methods *Within* and *Touches*.

1. $\mathcal{S} = \{\text{Alice}, \text{Bob}\}$
2. $\mathcal{O} = \{\text{read}, \text{write}\}$
3. $\mathcal{RA} = \{\epsilon\}$
4. $\mathcal{RO} = \{\text{Building}, \text{Intersection (location} = \{\text{foo}, 3\ 0\}), \text{Intersection (location} = \{\text{foo}, 3\ 4\})\}$
5. The policy geometry is defined by the following *Polygon*
 $G_P = \{\text{foo}, 3\ 0, 6\ 1, 6\ 5, 1\ 5, 0\ 2, 3\ 0\}$

5.6.4 Evaluation of the Spatial Method *Within*

The following set of permissions declares the access statements for this evaluation.

1. *Statement1* = "Bob can write information object of class *Intersection* if the location is within G_P ":
 $\{\text{Bob}, \text{write}, \epsilon, // \text{Intersection}, \{./\text{location}, \text{Within}, G_P\}\} \rightarrow \text{Permit}.$
2. *Statement2* = "Alice cannot write information objects of class *Building*":
 $\{\text{Alice}, \text{write}, \epsilon, // \text{Building}, \epsilon\} \rightarrow \text{Deny}.$
3. *Statement3* = "All users can read information objects of class *Intersection*":
 $\{*, \text{read}, \epsilon, // \text{Intersection}, \epsilon\} \rightarrow \text{Permit}.$

Assuming the all-explicit declaration strategy, the above access statements declare an incomplete set. According to the definitions of section 4.5.3, page 134, the following requests are not matched: $\{\text{Alice}, \text{write}, \text{Intersection (foo}, 0\ 0\)\}$, $\{\text{Alice}, \text{write}, \text{Intersection (foo}, 3\ 0\)\}$, $\{\text{Alice}, \text{read}, \text{Building}\}$, $\{\text{Bob}, \text{write}, \text{Intersection (foo}, 3\ 0\)\}$, $\{\text{Bob}, \text{read}, \text{Building}\}$, $\{\text{Bob}, \text{write}, \text{Building}\}$. These requests result in the authorization decision *N/A*.

5.6.5 Evaluation of the Spatial Method *Touches*

The following set of permissions declares the access statements for this evaluation.

1. *Statement1* = "Bob can write information object of class *Intersection* if the location is touching G_P ":
 $\{\text{Bob}, \text{write}, \epsilon, // \text{Intersection}, \{./\text{location}, \text{Touches}, G_P\}\} \rightarrow \text{Permit}.$

Test case identification			Decision	Explanation
Alice	read	Intersection (foo,3 0)	Permit	The resource content contains a restricted resource. Alice is allowed to read the resource (Statement3)
Alice	write	Intersection (foo,3 0)	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	read	Intersection (foo,3 4)	Permit	The resource content contains a restricted resource. Alice is allowed to read the resource (Statement3)
Alice	write	Intersection (foo,3 4)	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	read	Building	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	write	Building	Deny	The resource content contains a restricted resource. Alice is not allowed to write the resource (Statement2)
Bob	read	Intersection (foo,3 0)	Permit	The resource content contains a restricted resource. Bob is allowed to read the resource (Statement3)
Bob	write	Intersection (foo,3 0)	N/A	No explicit statement is made about this resource, which results in N/A.
Bob	read	Intersection (foo,3 4)	Permit	The resource content contains a restricted resource. Bob is allowed to read the resource (Statement3)
Bob	write	Intersection (foo,3 4)	Permit	The resource content contains a restricted resource. Bob is allowed to write the resource (Statement1)
Bob	read	Building	N/A	No explicit statement is made about this resource, which results in N/A.
Bob	write	Building	N/A	No explicit statement is made about this resource, which results in N/A.

Table 5.6: Test cases for evaluation of spatial restrictions using the spatial relation Within

2. Statement2 = ‘Alice cannot write information object of class **Building**’:
 $\{Alice, write, \epsilon, //Building, \epsilon\} \rightarrow Deny$.
3. Statement3 = ‘All users can read information objects of class **Intersection**’:
 $\{*, read, \epsilon, //Intersection, \epsilon\} \rightarrow Permit$.

Assuming the all-explicit declaration strategy, the above access statements declare an incomplete set. According to the definitions of section see 4.5.3, page 134, the following requests are not matched: $\{Alice, write, Intersection (foo,0 0)\}$, $\{Alice, write, Intersection (foo,3 0)\}$, $\{Alice, read, Building\}$, $\{Bob, write, Intersection (foo,3 4)\}$, $\{Bob, read, Building\}$, $\{Bob, write, Building\}$. These requests result in the authorization decision *N/A*.

Test case identification			Decision	Explanation
Alice	read	Intersection (foo,3 0)	Permit	The resource content contains a restricted resource. Alice is allowed to read the resource (Statement3)
Alice	write	Intersection (foo,3 0)	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	read	Intersection (foo,3 4)	Permit	The resource content contains a restricted resource. Alice is allowed to read the resource (Statement3)
Alice	write	Intersection (foo,3 4)	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	read	Building	N/A	No explicit statement is made about this resource, which results in N/A.
Alice	write	Building	Deny	The resource content contains a restricted resource. Alice is not allowed to write the resource (Statement2)
Bob	read	Intersection (foo,3 0)	Permit	The resource content contains a restricted resource. Bob is allowed to read the resource (Statement3)
Bob	write	Intersection (foo,3 0)	Permit	The resource content contains a restricted resource. Bob is allowed to write the resource (Statement1)
Bob	read	Intersection (foo,3 4)	Permit	The resource content contains a restricted resource. Bob is allowed to read the resource (Statement3)
Bob	write	Intersection (foo,3 4)	N/A	No explicit statement is made about this resource, which results in N/A.
Bob	read	Building	N/A	No explicit statement is made about this resource, which results in N/A.
Bob	write	Building	N/A	No explicit statement is made about this resource, which results in N/A.

Table 5.7: Test cases for evaluation of spatial restrictions using the spatial relation Touches

5.6.6 Evaluation of the Complex Spatial Restriction

The evaluation of the complex spatial restrictions requires to verify the correctness of the GeoXACML specific combining algorithms `or` and `and`. In order to do so, two different sets of permissions are declared. The first set declares spatial restrictions, using the combining algorithm `and` and the other set declares spatial permissions, using the combining algorithm `or`. Within each set, three different statements exist: The first statement declares a positive access right, the second statement defines a negative access right and the third statement defines an inconsistent permission.

Evaluation of the Complex Spatial Restriction using the combining algorithm and

The following set of permissions declares access statements for this evaluation. Statements 1 and 2 represent consistent access restrictions and statement 3 represents an inconsistent access restriction. This is required to test the error processing of the combining algorithm, according to the truth table 4.3, page 79.

1. Statement1 = ‘Alice can read spatial information objects of class `Intersection` if the location is not within G_{P1} and not within G_{P2} ’:

$$\begin{aligned}
 P1 &= \{*, *, \epsilon, //*, \text{and}, R11, R12\} \\
 R11 &= \{Alice, read, \epsilon, //Intersection, C11\} \rightarrow Permit \\
 R12 &= \{Alice, read, \epsilon, //Intersection, C12\} \rightarrow Permit \\
 C11 &= \{./location, \neg Within, \{foo, 3\ 0,6\ 1,6\ 5,1\ 5,0\ 2,3\ 0\}\} \\
 C12 &= \{./location, \neg Within, \{foo, -1\ -1,1\ -1,1\ 1,-1\ 1,-1\ -1\}\}
 \end{aligned}$$

2. Statement2 = ‘Alice cannot read spatial information objects of class `Intersection` if the location is not within G_{P1} and not within G_{P2} ’:

$$\begin{aligned}
 P2 &= \{*, *, \epsilon, //*, \text{and}, R21, R22\} \\
 R21 &= \{Alice, read, \epsilon, //Intersection, C21\} \rightarrow Deny \\
 R22 &= \{Alice, read, \epsilon, //Intersection, C22\} \rightarrow Deny \\
 C21 &= \{./location, \neg Within, \{foo, 3\ 0,6\ 1,6\ 5,1\ 5,0\ 2,3\ 0\}\} \\
 C22 &= \{./location, \neg Within, \{foo, -1\ -1,1\ -1,1\ 1,-1\ 1,-1\ -1\}\}
 \end{aligned}$$

3. Statement3 = ‘Alice can read spatial information objects of class `Intersection` if the location is not within G_{P1} and not within G_{P2} ’ (The declaration is inconsistent):

$$\begin{aligned}
 P3 &= \{*, *, \epsilon, //*, \text{and}, R31, R32\} \\
 R31 &= \{Alice, read, \epsilon, //Intersection, C31\} \rightarrow Permit \\
 R32 &= \{Alice, read, \epsilon, //Intersection, C32\} \rightarrow Deny \\
 C31 &= \{./location, \neg Within, \{foo, 3\ 0,6\ 1,6\ 5,1\ 5,0\ 2,3\ 0\}\} \\
 C32 &= \{./location, \neg Within, \{foo, -1\ -1,1\ -1,1\ 1,-1\ 1,-1\ -1\}\}
 \end{aligned}$$

Evaluation of the Complex Spatial Restriction using the combining algorithm or

The following set of permissions declares access statements for this evaluation. Statements 1 and 2 represent consistent access restrictions and statement 3 represents an inconsistent access restriction. This is required to test the error processing of the combining algorithm, according to the truth table 4.3, page 79.

1. Statement1 = ‘Alice can read spatial information objects of class *Intersection* if the location is within G_{P1} or within G_{P2} ’:

$$\begin{aligned}
 P1 &= \{*, *, \epsilon, //*, or, R11, R12\} \\
 R11 &= \{Alice, read, \epsilon, //Intersection, C11\} \rightarrow Permit \\
 R12 &= \{Alice, read, \epsilon, //Intersection, C12\} \rightarrow Permit \\
 C11 &= \{./location, Within, \{foo, 3 0,6 1,6 5,1 5,0 2,3 0\}\} \\
 C12 &= \{./location, Within, \{foo, -1 -1,1 -1,1 1,-1 1,-1 -1\}\}
 \end{aligned}$$

2. Statement2 = ‘Alice cannot read spatial information objects of class *Intersection* if the location is within G_{P1} or within G_{P2} ’:

$$\begin{aligned}
 P2 &= \{*, *, \epsilon, //*, or, R21, R22\} \\
 R21 &= \{Alice, read, \epsilon, //Intersection, C21\} \rightarrow Deny \\
 R22 &= \{Alice, read, \epsilon, //Intersection, C22\} \rightarrow Deny \\
 C21 &= \{./location, Within, \{foo, 3 0,6 1,6 5,1 5,0 2,3 0\}\} \\
 C22 &= \{./location, Within, \{foo, -1 -1,1 -1,1 1,-1 1,-1 -1\}\}
 \end{aligned}$$

3. Statement3 = ‘Alice can read spatial information objects of class *Intersection* if the location is within G_{P1} or within G_{P2} ’ (The declaration is inconsistent):

$$\begin{aligned}
 P3 &= \{*, *, \epsilon, //*, or, R31, R32\} \\
 R31 &= \{Alice, read, \epsilon, //Intersection, C31\} \rightarrow Permit \\
 R32 &= \{Alice, read, \epsilon, //Intersection, C32\} \rightarrow Deny \\
 C31 &= \{./location, Within, \{foo, 3 0,6 1,6 5,1 5,0 2,3 0\}\} \\
 C32 &= \{./location, Within, \{foo, -1 -1,1 -1,1 1,-1 1,-1 -1\}\}
 \end{aligned}$$

Test case identification			Decision	Explanation
Evaluation of statement 1				
Alice	read	Intersection (foo,3 0)	Permit	The resource geometry is not within G_{P1} and not within G_{P2} . Therefore, both rules (R1 and R2) apply.
Alice	read	Intersection (foo,0 0)	N/A	The resource geometry is not within G_{P1} but within G_{P2} . Therefore, rule R1 does but rule R2 does not apply.
Alice	read	Intersection (foo,3 4)	N/A	The resource geometry is within G_{P1} but not within G_{P2} . Therefore, rule R1 does not apply but rule R2 does apply.
Alice	read	Intersection (foo,0 0), Intersection (foo,3 4)	N/A	The resource geometries are within G_{P1} and within G_{P2} . Therefore, neither rule (R1 nor R2) apply.
Evaluation of statement 2				
Alice	read	Intersection (foo,3 0)	Deny	The resource geometry is not within G_{P1} and not within G_{P2} . Therefore, both rules (R1 and R2) apply.
Alice	read	Intersection (foo,0 0)	N/A	The resource geometry is not within G_{P1} but within G_{P2} . Therefore, rule R1 does but rule R2 does not apply.
Alice	read	Intersection (foo,3 4)	N/A	The resource geometry is within G_{P1} but not within G_{P2} . Therefore, rule R1 does not apply but rule R2 does apply.
Alice	read	Intersection (foo,0 0), Intersection (foo,3 4)	N/A	The resource geometries are within G_{P1} and within G_{P2} . Therefore, neither rule (R1 nor R2) apply.
Evaluation of statement 3				
Alice	read	Intersection (foo,3 0)	Indeterminate	The resource geometry is not within G_{P1} and not within G_{P2} . Therefore, both rules (R1 and R2) apply. But, the effects are different, so it results in Indeterminate
Alice	read	Intersection (foo,0 0)	N/A	The resource geometry is not within G_{P1} but within G_{P2} . Therefore, rule R1 does but rule R2 does not apply.
Alice	read	Intersection (foo,3 4)	N/A	The resource geometry is within G_{P1} but not within G_{P2} . Therefore, rule R1 does not apply but rule R2 does apply.
Alice	read	Intersection (foo,0 0), Intersection (foo,3 4)	N/A	The resource geometries are within G_{P1} and within G_{P2} . Therefore, neither rule (R1 nor R2) apply.

Table 5.8: Test cases for evaluation of complex spatial restrictions using the combining algorithm and

Test case identification			Decision	Explanation
Evaluation of statement 1				
Alice	read	Intersection (foo,3 0)	N/A	The resource geometry is not within G_{P1} and not within G_{P2} . Therefore, neither rule (R1 nor R2) applies.
Alice	read	Intersection (foo,0 0)	Permit	The resource geometry is not within G_{P1} but within G_{P2} . Therefore, rule R1 does but rule R2 does not apply.
Alice	read	Intersection (foo,3 4)	Permit	The resource geometry is within G_{P1} but not within G_{P2} . Therefore, rule R1 does not apply but rule R2 does apply.
Alice	read	Intersection (foo,0 0), Intersection (foo,3 4)	Permit	The resource geometries are within G_{P1} and within G_{P2} . Therefore, rules R1 and R2 apply.
Evaluation of statement 2				
Alice	read	Intersection (foo,3 0)	N/A	The resource geometry is not within G_{P1} and not within G_{P2} . Therefore, neither rule (R1 nor R2) applies.
Alice	read	Intersection (foo,0 0)	Deny	The resource geometry is not within G_{P1} but within G_{P2} . Therefore, rule R1 does but rule R2 does not apply.
Alice	read	Intersection (foo,3 4)	Deny	The resource geometry is within G_{P1} but not within G_{P2} . Therefore, rule R1 does not apply but rule R2 does apply.
Alice	read	Intersection (foo,0 0), Intersection (foo,3 4)	Deny	The resource geometries are within G_{P1} and within G_{P2} . Therefore, rules R1 and R2 apply.
Evaluation of statement 3				
Alice	read	Intersection (foo,3 0)	N/A	The resource geometry is not within G_{P1} and not within G_{P2} . Therefore, neither rule (R1 nor R2) applies.
Alice	read	Intersection (foo,0 0)	Permit	The resource geometry is not within G_{P1} but within G_{P2} . Therefore, rule R1 does but rule R2 does not apply.
Alice	read	Intersection (foo,3 4)	Deny	The resource geometry is within G_{P1} but not within G_{P2} . Therefore, rule R1 does not apply but rule R2 does apply.
Alice	read	Intersection (foo,0 0), Intersection (foo,3 4)	Indeterminate	The resource geometries are within G_{P1} and within G_{P2} . Therefore, rules R1 and R2 apply. But, both define different effects, which results in Indeterminate.

Table 5.9: Test cases for evaluation of complex spatial restrictions using the combining algorithm or

Chapter 6

Conclusion and Outlook

This chapter concludes with the results of this work, provides recommendations to the world of science and an outlook for possible additional research.

6.1 Conclusion

The starting point for this work was the lack of access control for the service-oriented infrastructure, enabling interoperable use of distributed and heterogenous geodata. International standards of the Open GIS Consortium, Inc. (OGC) are available that covers the construction of such an infrastructure. But, access control is not covered by the standards. A poll at the InterGeo in 2002 and the discussion with OGC members unveiled the urgent need of access control for such an infrastructure. In particular important was named the aspect to allow a fine-grained declaration of permissions, based on an object-oriented data model. It was also named important to declare permissions, based on the spatial characteristics of the geospatial information objects. These investigations caused the “initial ignition” for this work.

Before starting own research and development in this field, relevant standards, quasi standards and existing systems have been evaluated. The standards have been evaluated to see if they support the required declaration and enforcement of access restrictions and the implementation of a distributed access control system. The systems have been evaluated under the constraint if such a system meets the requirements. The result was that no standard and system is compliant with the introduced requirements. However, the eXtensible Access Control Markup (XACML) standard by OASIS was selected as a baseline for this work.

The given service-oriented infrastructure enables an interoperable use of distributed and heterogenous geodata. Because the use of the geodata is not always unrestricted, an access control system must enforce existing restrictions. This work has introduced requirements for such an access control system as it is important from the geodata providers perspective. Three of the important geodata provider requirements have been used to develop a distributed access control system: class-based, object-based and spatial requirements. Assuming an object-oriented data model that is marked-up in GML, a class-based access restriction controls the access to objects as they are instances of a particular class. The object-based restriction is more specific as it supports to restrict access to particular instances of information objects. Hence, it restricts the access to be based on non-spatial characteristics. The spatial access

restriction enables to declare restrictions on objects, based on their spatial characteristics; their geometry. In order to express such spatial restrictions, spatial attributes and functions are used that test a topological relation of two geometries: One geometry defines the boundary of the restricted area (the permission geometry) and the other is the geometry of a resource object.

This results in one of the main achievements of this work: Extending the existing XACML standard to GeoXACML and providing the means for the declaration and enforcement of spatial access restrictions. A prototype for GeoXACML has been implemented to verify the theoretical model. Different test cases for the evaluation of the class-based, object-based and spatial restrictions are enumerated.

The other important achievement of this work is to test an existing permission repository for correctness. Because it is most important for a complex access control system to ensure appropriate and error-free enforcement of declared permissions, the detection of different kinds of inconsistencies have been introduced. One kind of inconsistency results from unreachable permissions. Here, the declared positive or negative permission cannot be enforced. Another kind of inconsistency is that reachable permissions declare contrary rights: One permission represents a positive and the other a negative permission. The third kind of inconsistency is incompleteness, which characterizes the error that not all possible access requests result in an authorization decision of deny or permit. Based on the capabilities of the XACML standard, the detection of these kinds of inconsistencies have been investigated under the following assumptions: No knowledge about the possible requests is available and perfect knowledge about subjects, operations and resources is available. The first case results in an approximation of incorrectness. The result is a classification for two given permissions, which is either assured, likely or impossible for the tested kind of incorrectness. For the second case, where all possible requests are known, the detection outputs the cause for each kind of inconsistency and can therefore be used for correction.

In order to apply the developed model to the given service-oriented infrastructure, two different Enforcement Services have been developed: One for the Web Feature Service and one for the Web Map Service. Both Enforcement Services have been implemented according to the mediator pattern. This requires that the Enforcement Services mirror the operations of the facaded WFS and WMS in order to guarantee interoperability. For deriving authorization decisions according to the introduced model, the prototype of a geospatial Authorization Service according to the GeoXACML extension of the XACML standard, has also be implemented.

6.2 Recommendation to the World of (Geospatial) Science

Different approaches to build up a global infrastructure for geodata exist. The Global Spatial Data Infrastructure (GSDI) [GSDI 2004] is the attempt to make geodata interoperable. Different international and national attempts in such direction exist, such as the INfrastructure for SPatial InfoRmation in Europe (INSPPIRE) [European Commission] on the European level. All have in common to provide interoperable access to geodata.

The service-oriented approach provides interoperability by deploying Web Services that allow access to geodata in an interoperable way. This infrastructure has very strong potential to overcome the challenges of data integration, because it allows decoupling of the local data

formats and data models as they are used by the data providers and an interoperable data structure and model, required for combined use.

Even the service-oriented federation of distributed geodata is a promising architecture, it requires the implementation of security to become accepted. However, this service-oriented infrastructure requires access control, as it is one building block of the security, in order to allow data providers to make their protected data available. The foundation of access control, as introduced in this work is seen as the first building block to extend the service-oriented infrastructure for geodata in the required way. Access control on the information object level enables important kinds of geodata protection.

In such respect, this work is understood to cover the basic principles in order to apply access control to a service-oriented infrastructure that provides access to geospatial information objects. It is the hope of the author that this work makes the geospatial science aware that it is important to standardize access control for the service-oriented infrastructure of geodata, as it is one security building block. The author also hopes that this work triggers activities in this field.

6.3 Outlook

The aspect of this section is to highlight possible issues for further research. The highlighted aspects have been identified during preparation of this work, but have not been considered in more detail.

6.3.1 The ‘Not Authorized’ Response and the Adversary Issue

The main purpose of an access control system is to enforce access restrictions as declared. For a web-based access control as deployed today, restrictions are associated with an internal structure of the web server. In such a system, a user can typically request files that are located in a sub-tree of the restricted path. The access control does not verify the information objects, contained in the files. For the access control model as introduced in this work, the declaration of a ‘fine-grained’ or information-based access control is possible. Compared to the typical approach, it means that the access depends on the content of a file. Therefore, it is more likely that a user may not have the appropriate permission(s) for all resource objects, fetched by the request. In such a case, the entire request can be denied and the user receives the ‘not authorized’ information.

For the access control on web servers, it is typically clear what permission is missing. This can be derived from the directory path, contained in the request. The user is typically informed by the ‘not authorized’ page that insufficient permissions exist. But, what information is to be returned for the fine-grained access control? The ‘not authorized’ message is not very helpful for the user, because it is impossible to derive the reason. In order to find the missing permission(s), more information must be returned to the user. The problem with that is that exploiting internal access control information can be used by a possible adversary in order to gain unallowed access.

This results in some sort of rhetorical question: How much information can the access control system return to a user with insufficient permissions? Does the system have any

trustworthy information about the user upon which the system can determine a trusted user or must expect an adversary? If the system can trust the user, how much internal information about existing permissions can the system return, in order to help the user and not unveil confidential information? One possible approach can be based on a Public Key Infrastructure to identify the user and differentiate between a trusted and untrusted user; the possible adversary. Another approach can possibly rely on modeling social relationships between users, as introduced in [Galla 2004]. If the access control system can verify that a user with sufficient permissions manifests that the current user can be trusted, more intimate access control information can be exploited. If such a trust relationship cannot be verified, the standard “not authorized” message can be returned.

This research topic focuses on the approach to build up trust relationships between users that can be used to derive, how much information is returned to the user in case that insufficient permissions are present.

Another field of research, as it is related to the topic of unveiling access control information, can focus on the problem of exploiting internal information about the cause of the denial can be used by smart adversaries as illustrated in the following example. If denial information, such as “you are missing permissions to see the oil pipelines in the area of Iran” is returned, the adversary can unveil the path of the pipeline by sending “smart” requests without having the explicit permission: Knowing, that all requests for pipelines will be denied if a pipeline is comprised in the response, the requests can be modified in such a way that they cluster the space according to the QuadTree [Finkel et. al. 1974] approach. After appropriate clustering, the adversary gets the restricted information, even though not having adequate (explicit) permissions.

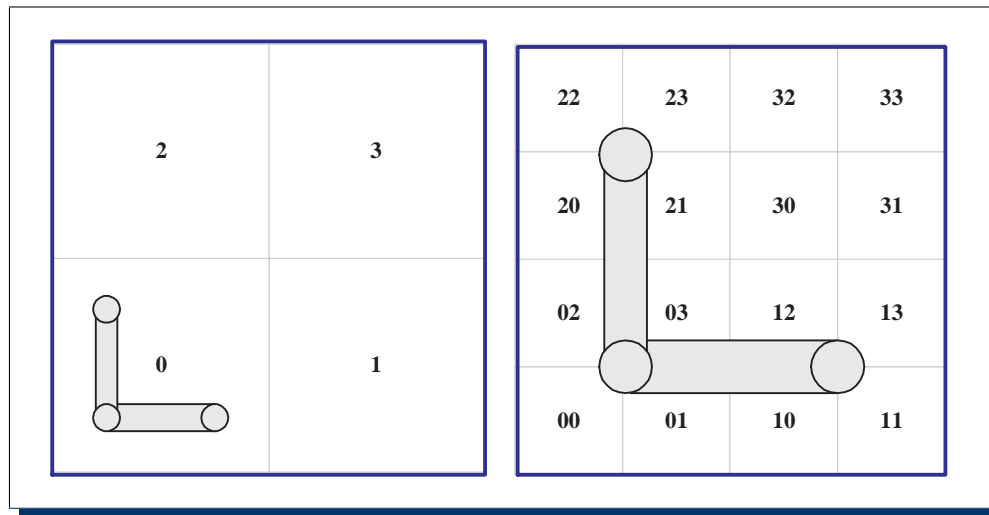


Figure 6.1: Unveiling of restricted information without permission by smart clustering of space

Figure 6.1 illustrates this approach: The first request covers the four large areas (0, 1, 2, 3), where a pipeline is within area zero. This results in one denied and three permitted requests, as illustrated on the left sub-figure. In the next step, the area for the denial is clustered and requests are issued for the areas 01, 02, 03, 04, . . . 30, 31, 32, 33. This results in permitted and denial requests as illustrated on the right sub-figure: The requests for the

areas 30, 31, 32 and 33 are permitted and for the area 01, 02, 03, 04, 10, 11, 12, 13, 20, 21, 22, 23 are denied. This clustering can be continued, until the unveiled path of the pipeline shows the desired detail.

The research for this aspect can probably focus on request pattern recognition. A sophisticated system is challenged to analyse the requests, issued by one subject if obeying to a particular pattern. Two possible questions can be identified:

- How can an adversary pattern be found, constructed and adopted that it allows to separate adversary requests from legitimate requests?
- If a particular ‘adversary’ pattern can be determined, what conclusion can an access control system derive in order to control the information that is returned to the potential adversary and handling further requests?

6.3.2 The Handling of Requests with Insufficient Permissions

The introduced access control model allows to restrict the access to geospatial information objects, based on different aspects: class-based, object-based and spatial restrictions. For such a model, a global access control strategy must define if a request with insufficient permissions is completely denied or if it is modified in such a way that the user receives the information objects, for which permissions are present.

The case, where the request is completely denied is the simplest case. However, for the introduced information object-based access control, it is not always suitable. For example, a spatial permission can restrict the access to an area, where the boundary is described by a linear ring. Due to limitations of available services, the user can only request information for a rectangular box. Now, the situation can appear that the user does not have any interest in receiving a map for the area, where the user does not have appropriate permission. But, due to the limitations of the service, the user cannot define the exact area of interest in such a way that it represents the interest and obeys to the existing permissions.

Figure 6.2 illustrates the problem: Let’s assume that a user has permissions to receive maps for the sovereign area of Bavaria but not for Baden-Württemberg. Because the used software allows only the request of rectangular boxes, the user must also request a map for Baden-Württemberg if getting close to the border line. But, because the permission for Baden-Württemberg is missing, the entire request is denied. This limitation is not acceptable and results in the question, under which circumstances the request can/must be modified, according to the user’s permissions. If it is essential for the user to receive maps, close to the border line it is required that the service response is modified according to the user’s permissions.

A technical solution to this problem is provided by XACML, called Obligations. An obligation can be described as an optional command that must be carried out by the Enforcement Service. It is added to an authorization decision result, generated by the Authorization Service. The command(s) of the obligation must either be applied to the actual service request or to the response. For GML (or XML) encoded service responses, an obligation can contain an Extensible Stylesheet Language (XSL) script that filters the service response, based on the permissions of the subject. For binary image maps, as they are returned by the WMS,

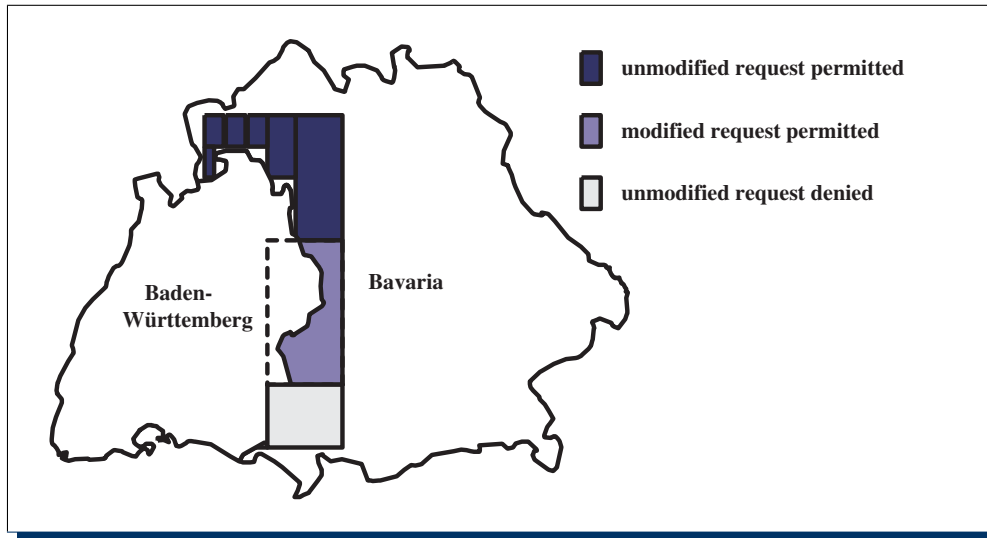


Figure 6.2: Software limitations for requesting a map for a restricted areas

image postprocessing must take place in order to modify the service response according to the user's permissions. In this case, it can be envisioned that obligations either describe a set of image processing commands or contain a 'blank' image that is to be overlaid to the actual service map as illustrated in figure 6.3.

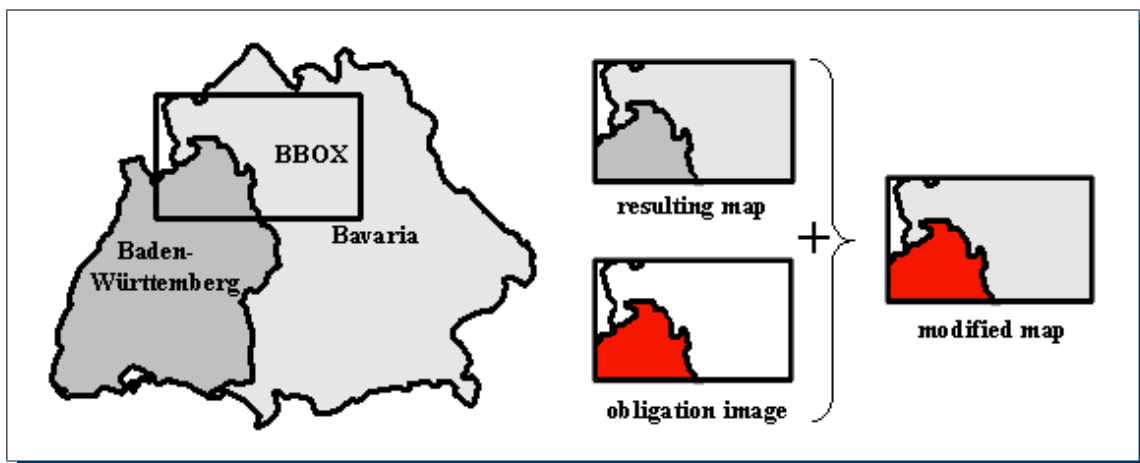


Figure 6.3: Applying an obligation image to the WMS map, eliminating the parts of the map for which the user does not have appropriate permissions

The focus for additional research is to identify situations, under which such a modification is required, essential or suitable and how they can be declared, using a particular access control declaration language. It can also be envisioned that a programming language can be developed that allows the generic definition of how to create obligation, what they must ensure, etc.

6.3.3 Permission Management and the Development of Service Orchestration

The service-based infrastructure provides interoperable access to heterogeneous and distributed geodata. As such, the services can be used by application developers to build more complex solutions. Hereby, it is the intention of the developer that a particular application solves a given problem. In order to achieve that, the application requires particular access to services and information objects. If the application interacts with services where the resources are protected, applicable permissions must exist. In this context of developing and using such applications, basically two different questions arise:

1. A user of the application likes to fulfill a particular task, which requires designated access to a particular set of resource objects. Which permissions are required in order to complete the task?
2. A user has a particular set of permissions. Which applications and resources can be accessed, using particular operations?

The first question refers to permission management, where a user has a particular job function. According to that job function, access roles can be defined which hold a particular set of permissions. Given a particular application that can be used by users, having a particular role, which permissions must be associated to the role, that the user can do the job?

The second question refers to the inverse: If a user has a particular role, which applications can be used and which services and information objects can be accessed? It is adequate for the given user, is it too little or too much? If the user has too little permissions, important tasks can not be performed. If the user has too many permissions, possible misuse can not be excluded.

The author envisions the possible fields of research in defining mechanisms that allow to answer both questions. Then, additional research can possibly focus on the evaluation of the appropriate set of permissions for a particular job assignment.

6.3.4 Context-based Permissions

For the development of service-based applications, the OGC has emphasized three possible chaining types in their document about service infrastructure [OGC 2002-112]: transparent, translucent and opaque chaining. Each chaining model has implications for the declaration of permissions, as introduced in more detail in the following sub-sections.

Transparent Chaining

Transparent chaining, as illustrated in figure 6.4 can be described in such a way that a user enquires appropriate services from a service broker (UDDI¹ Registry in figure 6.4). The user evaluates the description of the services and identifies the appropriate usage. Such a service chaining is typically used by an application developer, finding applicable services.

¹Universal Description, Discovery and Integration

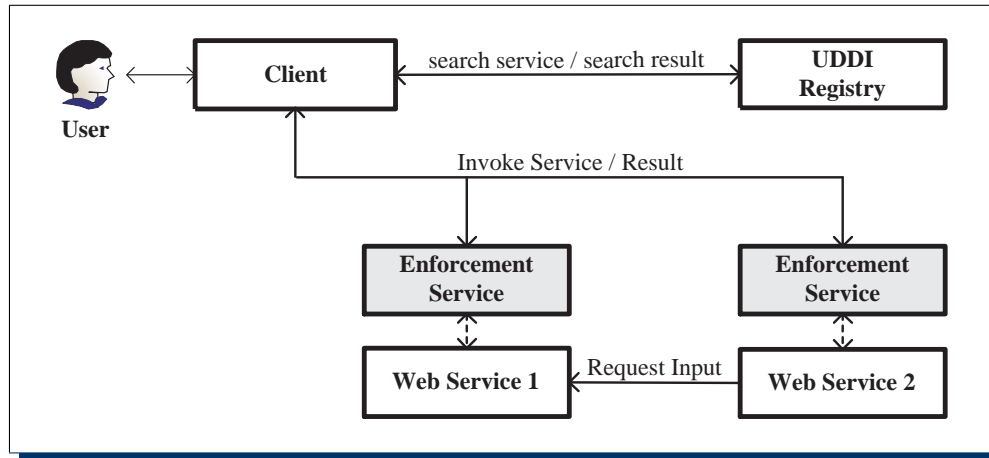


Figure 6.4: Transparent chaining of protected services

For the transparent service chaining as for the individual service use, the user's authentication assertions can be used to declare possible permissions. The author does not see the need for any additional research, because the introduced access control model provides the essential capabilities.

Translucent Chaining

Translucent chaining, as illustrated in figure 6.5 can be described in such a way that a Workflow Service exists that knows what to do. It knows the required services, their sequence and how to invoke them. It further can interpret the response of a previous service to eventually build the request for another service.

In this chaining scenario, the user's actions are limited to start and stop the Workflow Service. This will start, resp. interrupt the execution of the underlying services. In order to determine if an interruption is required, the user receives the processing states of each invoked service. For example, this monitoring capability is beneficiary if a service is invoked recursively and the user must judge for convergence. According to access control, one important piece of state information is if a request to a service was denied. In such a case, the processing of the remaining services can probably be interrupted.

For such a service chaining, where the Workflow Service invokes the underlying services, which authentication assertions can be used? The authentication assertions of the user, of the service or both?

In order to answer that question, the author sees two different possibilities: (i) Because the user receives state information from each underlying service, the declared permissions can be based on the subject authentication assertions. (ii) The Workflow Service invokes the underlying services and is therefore also known to those services. The declaration of permissions can also be based on the Workflow Service authentication assertions.

The author envisions additional research, focusing on these issues in more detail. The result of that research might possibly result in an extension to existing standards, such as the Web Services Flow Language (WSFL) [IBM 2001].

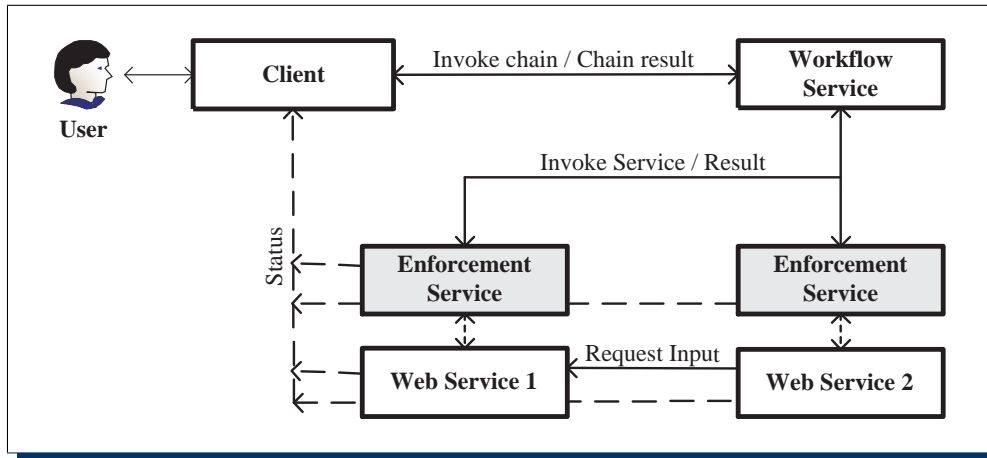


Figure 6.5: Translucent chaining of protected services

Opaque Chaining

Opaque chaining eliminates the user's awareness, which services are being executed, the sequence and what the intermediate processing states are. In addition, the user has no awareness about the requests and responses, exchanged between the Aggregate Service and the aggregated services. The Aggregate Service has the capability to handle "everything" in an autonomous way. It acts like a software agent on behalf of the user.

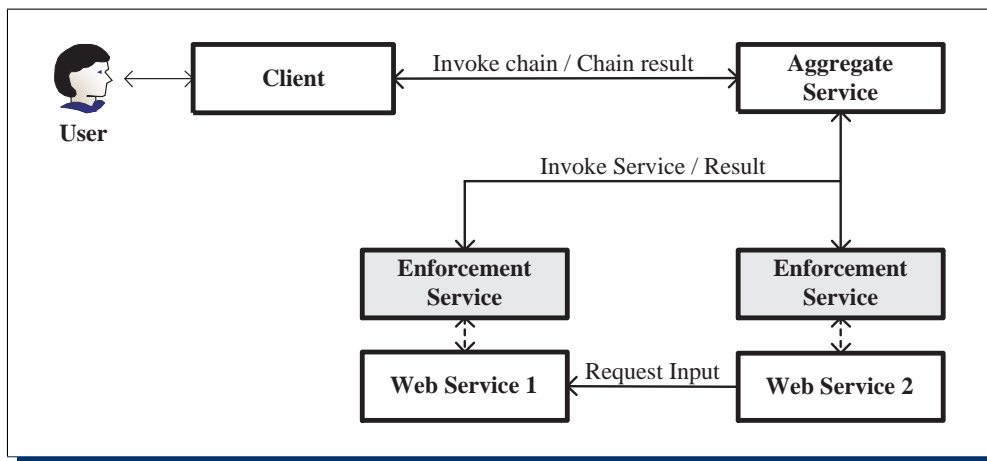


Figure 6.6: Opaque chaining of protected services

For this type of service chaining, the subject's authentication assertions can not be used to model the permissions for the aggregated services. This is because the user is not known to the aggregated services. However, associating the permissions of the aggregated service to the authentication assertions of the Aggregate Service has the impact that different users of the Aggregate Service have the same permissions on the aggregated services. This can not be practical and the author therefore envisions that context-based permissions are required. A context-based permission can possibly declare that the Aggregate Service has particular permissions, if executed by a particular user. The author sees a second argument for the

declaration of context specific permissions instead of using the subject's or Aggregate Service's authentication assertions: It is possible that the Aggregate Service might require additional permissions that the user does not have. This, in order to complete a particular sub-task that is required. For example, let's take an application that allows the user to request a map for a given address. The user might therefore have the permission to request maps, but might not have permissions for the required geocoding step that converts the address to the bounding box, required for requesting the map. But, the Aggregate Service must have the permission to invoke a service that returns a bounding box for a given address.

Using context-based permissions allows to declare the restriction that the geocoding service can only be invoked, if the Aggregate Service is used by a particular subject. The Aggregate Service itself and the user itself can not use the geocoding service.

The author envisions that additional research can focus on a model that allows to declare and enforce access restrictions, based on the context-based permissions.

6.3.5 Dynamic Negotiation of Authentication Information under Consideration of Privacy

Another field of research can be seen with dynamic evaluation of permissions, based on attribute assertions. For such an infrastructure, the Authorization Service requests assertions about the subject from an Authentication Service, as implemented in the Cardea system. Here, the Authorization Service requests attribute assertions from the Authentication Service without control and awareness of the user.

In order to protect the privacy of the user, particular attribute assertions might not be released to a specific Authorization Service. This requires that the user can configure access restrictions for her/his profile at the Authentication Service as it is being used to answer dynamic assertion requests. For example, the user configures that credit card information is never released if not manually accepted. Or, that the home phone number is only released if the work and mobile phone number do not provide the required permissions.

In this field of research, social relationships as introduced in [Galla 2004] can be used to model social relationships between subjects, upon which the release of attribute assertions is reasoned. For example, the user configures the release of assertions in such a way that a particular Authorization Service can request assertions if a friend also trusts that Authorization Service. This requires to model the social relationships between the subjects and the trusting relationship to particular Authorization Services.

A baseline for starting the research in the field of handling privacy issues can be found in [Wörndl 2003]. Here, the negotiation of user profile attributes takes place under the aspect of compliance to privacy. Further research can adopt the introduced mechanism according to the usage requirements for dynamic exchange of attribute assertions in the service-oriented infrastructure for accessing distributed and heterogenous geodata.

Appendix A

Notation

A.1 Nomenclature

$2^{\mathcal{R}\mathcal{O}}$ Powerset of available resource objects

\angle Operator that defines the subordinate relation: e.g.: $A \angle B$: A is subordinate to B

\cup Union

\vee Logical OR

\wedge Intersection

\bigwedge Logical AND

\mathcal{CL} Set of classifications for the approximate detection of contrary spatial restrictions

\mathcal{CL}_s Subset of classifications ($\mathcal{CL}_f \subseteq \mathcal{CL}$)

$\mathcal{L}(\text{reg.-expr.})$ Language of the regular expression *reg.-expr.*

$\mathcal{L}(\text{str.})$ Language that contains of one word: $\{\text{str.}\}$

\mathcal{P} Set of permissions, encoded as Policy constructs

\mathcal{P}_A Set of applicable policies, based on matching the SOR tuple

\mathcal{PS} Set of permissions, encoded as PolicySet constructs

\mathcal{R} Set of permissions, encoded as Rule constructs

\mathcal{R}_A Set of applicable rules, based on matching the SOR tuple

\mathcal{R}_E Set of enforceable rules

\mathcal{RA} Set of resource attributes

- \mathcal{RO} Total set of protected resource objects, representing the maximum resource content
- \mathcal{RT} Resource content template
- SOR Set of all possible request tuples
- \mathcal{V} Tuple of values, used for comparison
- \vec{o} Vector of outcomes from enforceable rules where the sequence of the entries is according to the sequence of the rules
- \vec{sm} Spatial method vector
- \vec{sm}_I Instance of a spatial method vector
- \times Cartesian product
- Max_{\setminus} Function that returns the top most superordinate classification from a set of sub/superordinate classifications
- \mathcal{RO}_S Subset of the protected resource objects ($\mathcal{RO}_S \subseteq \mathcal{RO}$)
- ^C Superscript C denotes expressions, related to a class-based restriction
- ^O Superscript O denotes expressions, related to a object-based restriction
- ^S Superscript S denotes expressions, related to a spatial restriction
- $B(G)$ Boundary of the geometry G
- C Condition element of a rule construct
- c Operation `create` that can be applied to a resource object
- C1, C2, ... Example conditions, named C1, C2, etc.
- C^C Condition for declaring a class-based restriction
- C^O Condition, expressing a matching condition of non-spatial characteristics of objects, resp. features
- C^S Condition, restricting the matching on spatial characteristics of objects, resp. features
- cl A single classification
- $CL(SC)$ A classification function that returns the classification of the spatial condition SC
- CM_R Condition matching element of a rule R
- CRS Coordinate Reference System
- d Operation `delete` that can be applied to a resource object

-
- DAC Discretionary Access Control
- $\dim(G)$ Dimension of the geometry G
- DOM Document Object Model
- DRM Digital Rights Management
- DTD Document Type Definition
- $E(G)$ Exterior of the geometry G
- E_{R_i} The effect of the rule R_i
- EPSG European Petroleum Survey Group
- EPSG:31494 EPSG code for Gauss-Krüger projection, zone 4
- EPSG:4326 EPSG code for WGS84 projection
- G_P Permission geometry
- G_R Resource geometry
- GIS Geographic Information System
- GML Geography Markup Language
- GPS Global Positioning System
- GWS Geo Web Service
- $I(G)$ Interior of the geometry G
- I_{R_1, R_2}^C Incorrect pair of rules R_1 and R_2 that encode class-based restrictions, where the effect of R_1 is different from the effect of R_2
- I_{R_1, R_2}^O Incorrect pair of rules R_1 and R_2 that encode object-based restrictions, where the effect of R_1 is different from the effect of R_2
- ISO International Organization for Standardization
- JTS Java Topology Suite
- LAT Latitude; measuring northward and southward of the Equator
- LON Longitude; measuring eastward and westward the prime meridian Greenwich
- m Operation **map** that can be applied to a resource object
- MAC Mandatory Access Control
- Match_O Boolean match function for the operation element of a permission and the request

- Match_R Boolean match function for the resource element of a permission and the request
 Match_S Boolean match function for the subject element of a permission and the request
 Match_{XR} Boolean function for testing if a given resource content contains particular objects
 O Operation element of the request tuple
 o Outcome of a rule
OASIS Organization for the Advancement of Structured Information Standards
OGC Open GIS Consortium, Inc. (recently changed to Open Geospatial Consortium, Inc. (OGC))
 OM_R Operation matching element of a rule construct
 OM_{PS} Operation matching element of a PolicySet PS
 OM_P Operation matching element of a Policy P
 OM_R Operation matching element of a rule R
P1, P2, ... Example policies, named P1, P2, etc.
 P_A Applicable policy, based on matching the SOR tuple
 P_{C_j} jth Policy (P_j) that defines an all-quantifier matching
 P_j jth Policy
 P^+ Positive permission
 P^- Negative permission
PDP Policy Decision Point
PEP Policy Enforcement Point
PS1, PS2, ... Example policy set constructs, named Ps1, PS2, etc.
 PS_{C_i} ith PolicySet (PS_i) that defines an all-quantifier matching
 PS_i ith PolicySet
 R Resource element of the request tuple
 r Operation read that can be applied to a resource object
R1, R2, ... Example rules, named R1, R2, etc.
 R_A Applicable rule, based on matching the SOR tuple
 R_{C_k} kth Rule (R_k) that defines an all-quantifier matching

R_k	kth Rule
RBAC	Role Based Access Control
RM_R	Resource matching element of a rule construct
RM_{PS}	Resource matching element of a PolicySet PS
RM_P	Resource matching element of a Policy P
RM_R	Resource matching element of a rule R
Rule ^C	Rule, encoding the class-based restriction
Rule ^O	Rule, encoding an object-based restrictions
Rule ^S	Rule, encoding a spatial restriction
S	Subject element of the request tuple
SC	Spatial condition to determine contrary spatial permissions
SM_R	Subject matching element of a rule construct
SM_{PS}	Subject matching element of a PolicySet PS
SM_P	Subject matching element of a Policy P
SM_R	Subject matching element of a rule R
SOR_{PS}	Subject-Operation-Resource matching tuple of a PolicySet PS
SOR_P	Subject-Operation-Resource matching tuple of a Policy P
SOR_R	Subject-Operation-Resource matching tuple of a Rule R
SOR_{PS}^+	Subject-Operation-Resource tuples, matched by the PolicySet PS
SOR_{PS}^-	Subject-Operation-Resource tuples, not matched by the PolicySet PS
SOR_P^+	Subject-Operation-Resource tuples, matched by the Policy P
SOR_P^-	Subject-Operation-Resource tuples, not matched by the Policy P
SOR_R^+	Subject-Operation-Resource tuples, matched by the Rule R
SRS	Spatial Reference System; synonym for CRS
SVG	Scalable Vector Graphics
TCF	Topological condition function, such as Disjoint, Touches, Crosses, Within, Overlaps, Intersects or Equals
UML	Unified Modeling Language
URI	Uniform Resource Identifier

URL	Uniform Resource Locator
URN	Uniform Resource Name
w	Operation <i>write</i> that can be applied to a resource object
WFS	Web Feature Service
WGS84	World Geodetic Standard 1984
WMS	Web Map Service
XACML	eXtensible Access Control Markup Language
XF	Xpath node-set function
XM _{PS}	Resource content matching element of a PolicySet PS
XM _P	Resource content matching element of a Policy P
XM _R ^C	Xpath expression of a Rule for matching resource content objects that represent a class, resp. a GML feature type
XM ^C	Xpath expression of a Condition for matching resource content objects that represent a class, resp. a GML feature type
XM _C	Xpath expression of a Condition
XM _C ^O	Xpath expression of a Condition for matching objects, resp. GML features according to their non-spatial characteristics
XM _C ^S	Xpath expression of a Condition for matching objects, resp. GML features according to their non-spatial characteristics

A.2 Font Types

Building The type writer font refers to a class of an object-oriented data model, it's properties and methods.

GetFeature The sans serif family font denotes to constructs such as operations on resource objects, subjects and XML elements. For example, if the operation *read* is meant, it is put in sans serif font.

read the slanted shape font denotes value expressions as they are associated to constructs, class properties, etc. For example, the value *read* is put in slanted shape font and not in sans serif family font.

Citation The italic shape font denotes citations as they come from other document sources.

SET The calligraphic letters denote the use of a set. For example, \mathcal{RO} represents the set of resource content objects.

A.3 XML Spy Diagram Notation

Figure A.1 illustrates the notation of the XML Spy editor as it is used throughout this work for the graphical representation of XML schemas or parts of an XML schema.










	A single mandatory element.
	A single mandatory element with child elements.
	A single mandatory element containing Parsed Character Data (#PC-Data). The content may be simple content or mixed complex content.
	A single optional element.
	A multiple optional element (in this case, zero to infinity) with child elements.
	A placeholder for any element from any namespace.
	A complex type.
	A sequence. Solid lines connect this symbol to required elements within the sequence. Dotted lines connect this symbol to optional elements in the sequence.
	A choice.

Figure A.1: Conventions for diagrams, taken with Altova XML Spy (<http://www.xmlspy.com>)

Appendix B

The City Model

The City Model is the example used throughout this work. This chapter provides a map that shows the geometry and location of the used geospatial resource objects. Section B.2 provides the GML application schema that defines the GML markup of the underlying object-oriented model and the geometry of the spatial information objects. Section B.3 provides the GML encoding of the City Model as a GML feature collection.

B.1 Map of the City Model

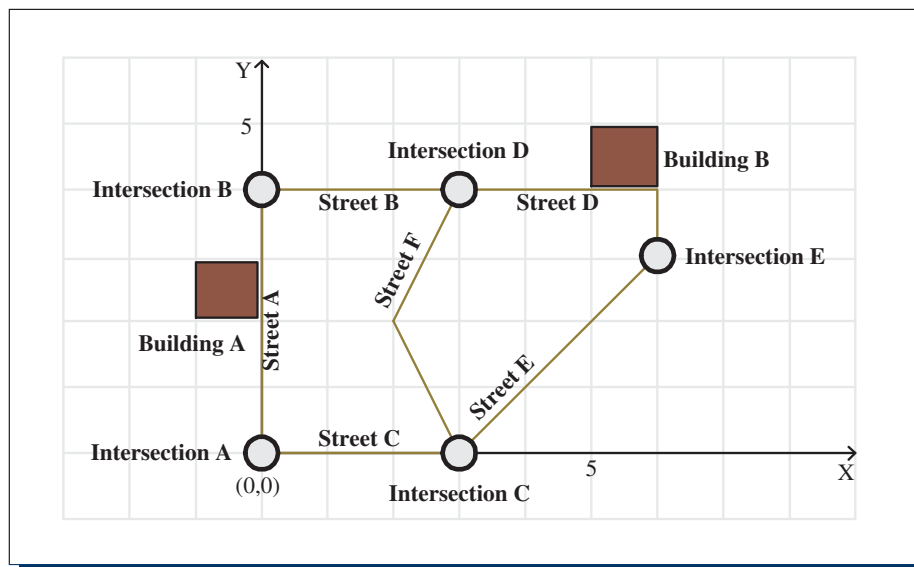


Figure B.1: The City Model map

B.2 City Model Application Schema

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema targetNamespace="http://www.in.tum.de/am"
3     xmlns="http://www.w3.org/2001/XMLSchema"
4     xmlns:am="http://www.in.tum.de/am"
5     xmlns:gml="http://www.opengis.net/gml"
6     xmlns:xs="http://www.w3.org/2001/XMLSchema"
7     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8     elementFormDefault="qualified">
9 <xs:import namespace="http://www.opengis.net/gml" schemaLocation="feature.xsd"/>
10 <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlinks.xsd"/>
11 <!-- -->
12 <xs:element name="CityModel" type="am:CityModelType" substitutionGroup="gml:_FeatureCollection"/>
13 <xs:element name="Street" type="am:StreetType" substitutionGroup="gml:_Feature"/>
14 <xs:element name="Intersection" type="am:IntersectionType" substitutionGroup="gml:_Feature"/>
15 <xs:element name="Building" type="am:BuildingType" substitutionGroup="gml:_Feature"/>
16 <xs:complexType name="CityModelType">
17 <xs:complexContent>
18 <xs:extension base="gml:AbstractFeatureCollectionType"/>
19 </xs:complexContent>
20 </xs:complexType>
21 <xs:complexType name="StreetType">
22 <xs:complexContent>
23 <xs:extension base="gml:AbstractFeatureType">
24 <xs:sequence minOccurs="0">
25 <xs:element name="Name"/>
26 <xs:element ref="gml:LineString"/>
27 </xs:sequence>
28 </xs:extension>
29 </xs:complexContent>
30 </xs:complexType>
31 <xs:complexType name="IntersectionType">
32 <xs:complexContent>
33 <xs:extension base="gml:AbstractFeatureType">
34 <xs:sequence minOccurs="0">
35 <xs:element name="Name"/>
36 <xs:element ref="gml:Point"/>
37 </xs:sequence>
38 </xs:extension>
39 </xs:complexContent>
40 </xs:complexType>
41 <xs:complexType name="BuildingType">
42 <xs:complexContent>
43 <xs:extension base="gml:AbstractFeatureType">
44 <xs:sequence minOccurs="0">
45 <xs:element name="Address"/>
46 <xs:element name="Shape" type="gml:PolygonType"/>
47 </xs:sequence>
48 </xs:extension>
49 </xs:complexContent>
50 </xs:complexType>
51 </xs:schema>

```

B.3 City Model GML Document

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <CityModel xmlns="http://www.in.tum.de/am" xmlns:xlink="http://www.w3.org/1999/xlink"
3   xmlns:gml="http://www.opengis.net/gml" xmlns:am="http://www.in.tum.de/am"
4   xsi:schemaLocation="http://http://www.in.tum.de/am CityModel.xsd"
5   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" fid="CityModel" >
6   <gml:boundedBy>
7     <gml:Box gid="box1" srsName="foo">
8       <gml:coord><gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
9       <gml:coord><gml:X>4</gml:X><gml:Y>4</gml:Y></gml:coord>
10    </gml:Box>
11  </gml:boundedBy>
12  <gml:featureMember>
13    <Intersection fid="I1">
14      <Name>Intersection A</Name>
15      <gml:Point srsName="foo">
16        <gml:coord><gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
17      </gml:Point>
18    </Intersection>
19  </gml:featureMember>
20  <gml:featureMember>
21    <Intersection fid="I2">
22      <Name>Intersection B</Name>
23      <gml:Point srsName="foo">
24        <gml:coord><gml:X>0</gml:X><gml:Y>4</gml:Y></gml:coord>
25      </gml:Point>
26    </Intersection>
27  </gml:featureMember>
28  <gml:featureMember>
29    <Intersection fid="I3">
30      <Name>Intersection C</Name>
31      <gml:Point srsName="foo">
32        <gml:coord><gml:X>2</gml:X><gml:Y>0</gml:Y></gml:coord>
33      </gml:Point>
34    </Intersection>
35  </gml:featureMember>
36  <gml:featureMember>
37    <Intersection fid="I4">
38      <Name>Intersection D</Name>
39      <gml:Point srsName="foo">
40        <gml:coord><gml:X>2</gml:X><gml:Y>4</gml:Y></gml:coord>
41      </gml:Point>
42    </Intersection>
43  </gml:featureMember>
44  <gml:featureMember>
45    <Intersection fid="I5">
46      <Name>Intersection E</Name>
47      <gml:Point srsName="foo">
48        <gml:coord><gml:X>4</gml:X><gml:Y>2</gml:Y></gml:coord>
49      </gml:Point>
50    </Intersection>
51  </gml:featureMember>
52  <gml:featureMember>
53    <Building xsi:type="BuildingType" fid="BuildingA">
54      <Address>Street A</Address>
55      <Shape srsName="foo">
56        <gml:outerBoundaryIs>
57          <gml:LinearRing>
58            <gml:coord><gml:X>-1</gml:X><gml:Y>2</gml:Y></gml:coord>
59            <gml:coord><gml:X>0</gml:X><gml:Y>2</gml:Y></gml:coord>
60            <gml:coord><gml:X>0</gml:X><gml:Y>3</gml:Y></gml:coord>
61            <gml:coord><gml:X>-1</gml:X><gml:Y>3</gml:Y></gml:coord>
62            <gml:coord><gml:X>-1</gml:X><gml:Y>2</gml:Y></gml:coord>
63          </gml:LinearRing>
64        </gml:outerBoundaryIs>

```

```

65     </Shape>
66   </Building>
67 </gml:featureMember>
68 <gml:featureMember>
69   <Building xsi:type="BuildingType" fid="BuildingB" >
70     <Address>Street D</Address>
71     <Shape srsName="foo" >
72       <gml:outerBoundaryIs>
73         <gml:LinearRing>
74           <gml:coord><gml:X>5</gml:X><gml:Y>4</gml:Y></gml:coord>
75           <gml:coord><gml:X>6</gml:X><gml:Y>4</gml:Y></gml:coord>
76           <gml:coord><gml:X>6</gml:X><gml:Y>5</gml:Y></gml:coord>
77           <gml:coord><gml:X>5</gml:X><gml:Y>5</gml:Y></gml:coord>
78           <gml:coord><gml:X>5</gml:X><gml:Y>4</gml:Y></gml:coord>
79         </gml:LinearRing>
80       </gml:outerBoundaryIs>
81     </Shape>
82   </Building>
83 </gml:featureMember>
84 <gml:featureMember>
85   <Street fid="S1" >
86     <Name>Street A</Name>
87     <gml:LineString>
88       <gml:coord><gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
89       <gml:coord><gml:X>0</gml:X><gml:Y>4</gml:Y></gml:coord>
90     </gml:LineString>
91   </Street>
92 </gml:featureMember>
93 <gml:featureMember>
94   <Street fid="S2" >
95     <Name>Street B</Name>
96     <gml:LineString>
97       <gml:coord><gml:X>0</gml:X><gml:Y>4</gml:Y></gml:coord>
98       <gml:coord><gml:X>2</gml:X><gml:Y>4</gml:Y></gml:coord>
99     </gml:LineString>
100  </Street>
101 </gml:featureMember>
102 <gml:featureMember>
103   <Street fid="S3" >
104     <Name>Street C</Name>
105     <gml:LineString>
106       <gml:coord><gml:X>0</gml:X><gml:Y>0</gml:Y></gml:coord>
107       <gml:coord><gml:X>2</gml:X><gml:Y>0</gml:Y></gml:coord>
108     </gml:LineString>
109   </Street>
110 </gml:featureMember>
111 <gml:featureMember>
112   <Street fid="S4" >
113     <Name>Street D</Name>
114     <gml:LineString>
115       <gml:coord><gml:X>2</gml:X><gml:Y>4</gml:Y></gml:coord>
116       <gml:coord><gml:X>4</gml:X><gml:Y>2</gml:Y></gml:coord>
117     </gml:LineString>
118   </Street>
119 </gml:featureMember>
120 <gml:featureMember>
121   <Street fid="S5" >
122     <Name>Street E</Name>
123     <gml:LineString>
124       <gml:coord><gml:X>2</gml:X><gml:Y>0</gml:Y></gml:coord>
125       <gml:coord><gml:X>4</gml:X><gml:Y>2</gml:Y></gml:coord>
126     </gml:LineString>
127   </Street>
128 </gml:featureMember>
129 </CityModel>

```


Bibliography

- [Akker et. al. 2001] Ty van den Akker, Quinn O. Snell, Mark J. Clement: The YGuard Access Control Model: Set-Based Access Control, ACM 1-58113-350-2/01/0005
- [AAP 2000] Association of American Publishers, Inc.: Digital Rights Management for Ebooks: Publisher Requirements, AAP 2000
- [Bhatti et. al.] Rafae Bhatti, James B.D. Joshi, Elisa Bertino, Arif Ghafoo: Access Control in Dynamic XML-based Web-Services with X-RBAC
- [Berliner Senatsverwaltung 2002] Berliner Senatsverwaltung: IT-Sicherheitsstandards (gemäß IT-Sicherheitsrichtlinie) Senatsverwaltung für Inneres, Ressortübergreifendes It-Management, Version 3.3, Stand: 27.06.02
- [Bertino et. al.] Elisa Bertino, Silvana Castano, Elena Ferrari, Marco Mesiti: Controlled Access and Dissemination of XML Documents
- [Bishr 1999] Yaser A. Bishr, University of Münster, Institute for Geoinformatics: A Global Unique Persistent Object ID for Geospatial Information Sharing, in Lecture Notes in Computer Science no. 1580, Interoperating Geographic Information Systems, Second International Conference INTEROP'99, Zurich, Switzerland, pages 55-64
- [Boström 2002] Erik Boström: Redefined Access Control in a Distributed Environment
- [Britannica] Britannica Concise Encyclopedia, <http://www.britannica.com/ebc/article?eu=390739&query=database&=gen1>
- [Burghardt et. al. 2003] Markus Burghardt, Svenja Hagenhoff: Sicherheitsaspekte bei der Nutzung von Web Services, Georg-August-Universität Göttingen, Arbeitsbericht Nr. 23/2003
- [Chen 2004] Willy Chen: Sicherheitsarchitektur einer XACML-basierten Zugriffskontrolle für Web Service, Diplomarbeit an der Technischen Universität München, 10.08.2004
- [Cohn et. al.] A. G. Cohn, N. M. Gotts, Z. Cui, D. A. Randell, B. Bennett, J.M. Gooday: Exploiting Temporal Continuity in Qualitative Spatial Calculi
- [Comer 2000] Douglas E. Comer: Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architectures, Prentice Hall, 2000
- [ContentGuard 2004] Content Guard Holdings Inc.: eXtensible rights Markup Language (XrML), 20 November 2001, http://www.xrml.org/get_XrML.asp

- [Cuppens et. al.] Frédéric Cuppens, Alexandre Miège: Modeling Contexts in the Or-BAC Model
- [Coetzee et. al.] M. Coetzee, Technikon Witwatersrand, J.H.P. Eloff, University of Pretoria: Virtual Enterprise Access Control Requirements
- [Clementini 1993] Clementini, Eliseo, Di Felice, P., van Oostrom, P.: A Small Set of Formal Topological Relationships Suitable for End-User Interaction, in D. Abel and B. C. Ooi (Ed.), Advances in Spatial Databases Third International Symposium. SSD 93. LNCS 692. Pp. 277-295. Springer-Verlag. Singapore (1993)
- [Damiani et. al. 2001] Ernesto Damiani, Sabrina de Capitani Di Vimercati, Stefano Paraboschi, Pierangela Samarati: Fine Grained Access Control for SOAP E-Services
- [Damiani et. al. 2002] Ernesto Damiani, Sabrina de Capitani Di Vimercati, Stefano Paraboschi, Pierangela Samarati: A Fine-Grained Access Control System for XML Documents, Data: 2002
- [Dastjerdi et. al. 1995] Ahmad Baraani-Dastjerdi, Reihaneh Safavi-Naini, Josef Perprzyk, Janusz R. Getta: A Model of Content-based Authorization in Object-Oriented Databases based on Object Views
- [Donaubauer 2004] Andreas Donaubauer: Interoperable Nutzung verteilter Geodatenbanken mittels standardisierter Geo Web Services, Dissertation, Technische Universität München
- [Eckert 2001] Claudia Eckert: IT-Sicherheit, Konzept - Verfahren - Protokolle, Oldenburg Verlag München, Wien
- [Egenhofer et. al.] Max J. Egenhofer, Reginald G. Golledge: Spatial And Temporal Reasoning In Geographic Information Systems
- [EPSG 2004] European Petroleum Survey Group: EPSG Geodesy Parameters V 6.5, 13 January 04, <http://www.epsg.org>
- [Internet2] N.N.: Shibboleth Introduction , <http://shibboleth.internet2.edu/shib-intro.html>
- [Erds et. al. 2002] Marlena Erds, Scott Cantor: Shibboleth-Architecture DRAFT v05, May 2, 2002
- [European Commission] European Commission: INfrastructure for SPatial InfoRmation in Europe, Homepage, <http://inspire.jrc.it/home.html>
- [Finkel et. al. 1974] Raphael A. Finkel, Jon Louis Bentley, Quad Trees: A Data Structure for Retrieval on Composite Keys, 1974
- [Galla 2004] Michael Galla: Social Relationship Management in Internet-based Communication and Shared Information Spaces, Dissertation, Technische Universität München
- [Guttman 1984] Antonin Guttman, R-TREES: A Dynamic Index Structure For Spatial Searching, 1984
- [GSDI 2004] Global Spatial Data Infrastructure Association: Homepage, <http://www.gsdi.org/default.asp>

- [Hada et. al. 2000] Satoshi Hada and Michiharu Kudo: XML Access Control Language: Provisional Authorization for XML Documents, www.tr1.ibm.com/projects/xml/xacl/xacl-spec.html
- [IBM 2001] IBM: Web Services Flow Language (WSFL 1.0), May 2001, <http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>
- [IBM et. al. 2002] IBM: XML Access Control Language: Provisional Authorization for XML Documents, October 16, 2000
- [IEEE 1990] N.N.: Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY, 1990
- [IETF 1997] R. Moats: AT&T, RFC 2141, May 1997, <http://community.roxen.com/developers/idocs/rfc/rfc2141.html>
- [ISO 1992] ISO: SQL 2-Standard (ISO 9075 - 1992)
- [ISO] ISO, SQL/MM Standard (ISO 13249)
- [Johnston et.al. 1998] William Johnston, Srilekha Mudumbai, Mary Thompson: Authorization and Attribute Certificates for a Widely Distributed Access Control IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises - WETICE '98
- [Kraft 2002] Reiner Kraft: Designing a Distributed Access Control Processor for Network Services on the Web, ACM 1-58113-632-3/02/0011
- [Kwok 2002] Sai Ho Kwok: digital Rights Management for the Online Music Business, ACM 1073-0516/01/0300-0034
- [Lepro 2003] Rebekah Lepro: NASA Advanced Supercomputing (NAS) Division, Cardea: Dynamic Access Control in Distributed Systems, November 2003
- [Liu et. al. 2003] Qiong Liu, Reihaneh Sfavi-Naini, Nicholas Paul Sheppard: Digital Rights Management for Content Distribution, Australian Computer Society, Inc. 2003
- [Lorch et. al. 2003] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, Sumit Shah: First Experiences Using XACML for Access Control in Distributed Systems, ACM 1-58113-777-X/03/0010
- [IDC 2003] Steve McClure: Oracle's Solution for Heterogeneous Data Integration, White Paper, August 2003, http://www.oracle.com/technology/products/dataint/pdf/idc_integration_wp.pdf
- [Microsoft 2004] Microsoft Corp.: Architecture of Windows Media Rights Manager, May 2004, <http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx>
- [Miller 2000] Paul Miller: Interoperability, What is it and why should I want it?, June 2000, <http://www.ariadne.ac.uk/issue24/interoperability>

- [Murata et al. 2003] Makoto Murata, Akihiko Tozawa, Michiharu Kudo: XML Access Control Using Static Analysis, ACM 1-58113-738-9/03/0010
- [Neumann et. al. 2003] Gustaf Neumann, Mak Strembeck: An Approach to Engineer and Enforce Context Constraints in an RBAC Environment
- [OASIS 2004a] OASIS: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
- [OASIS 2003] OASIS: eXtensible Access Control Markup Language (XACML), Version 1.1, Date: 24 July 2003
- [OASIS 2004b] OASIS: Security Assertion Markup Language (SAML) V1.1, <http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip>
- [OASIS 2004c] OASIS homepage <http://www.oasis-open.org>
- [OASIS 2004d] OASIS: XACML Profile for Role Based Access Control (RBAC), Committee Draft 01, 13 February 2004 Organization for the Advancement of Structured Information Standards (OASIS) homepage, <http://www.oasis-open.org/who/>
- [OASIS 2004e] OASIS: Web Services Security 2, UsernameToken Profile 1.0 3, OASIS Standard 200401, March 2004, <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>
- [OASIS 2004f] OASIS: Web Services Security: 2, SOAP Message Security 1.0 3 (WS-Security 2004) 4, OASIS Standard 200401, March 2004, <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>
- [OASIS 2004g] OASIS: Web Services Security 2, X.509 Certificate Token Profile 3, OASIS Standard 200401, March 2004, <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>
- [OGC 2004] Open GIS Inc. homepage, <http://www.opengis.org>
- [OGC 2000-028] Open GIS Consortium Inc.: Web Map Service Implementation Specification, Version: 1.0.0, Date: 2000-04-19, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 2001-068r3] Open GIS Consortium Inc.: Web Map Service Implementation Specification, Version: 1.1.1, Date: 2002-01-16, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 2002-058] Open GIS Consortium Inc.: Web Feature Service Implementation Specification, Version: 1.0.0, Date: 19-September-2002, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 2002-069] Open GIS Consortium Inc.: OpenGIS Geography Markup Language (GML) Implementation Specification, Version 2.1.2, Date: 17 September 2002, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 1999-049] Open GIS Consortium Inc.: OpenGIS Simple Features Specification For SQL, Revision 1.1, Release Date: May 5, 1999, <http://www.opengeospatial.org/specs/?page=specs>

- [OGC 1999-100r1] Open GIS Consortium Inc.: The OpenGIS Abstract Specification Topic 0: Abstract Specification Overview, Version 4, Date: 23 June 1999, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 2001-101] Open GIS Consortium Inc.: The OpenGIS Abstract Specification Topic 1: Feature Geometry (ISO 19107 Spatial Schema), Version 5, Date: 10 May 2001, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 1999-105r2] Open GIS Consortium Inc.: The OpenGIS Abstract Specification Topic 5: Features, Version 4, Date: 24 March 1999, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 1999-110] Open GIS Consortium Inc.: The OpenGIS Abstract Specification Topic 10: Feature Collections, Version 4, Date: 7 April 1999, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 2002-112] Open GIS Consortium Inc.: The OpenGIS Abstract Specification Topic 12: The OpenGIS Service Architecture, Version 4.3, Date: January 2002, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 2003-010r9] Open GIS Consortium Inc.: Recommended XML encoding of coordinate reference system definitions, Version: 2.1.0, Date: 2003-10-16, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 02-059] Open GIS Consortium Inc.: Filter Encoding Implementation Specification, Version 1.0.0, Date: 17-MAY-2001, <http://www.opengeospatial.org/specs/?page=specs>
- [OGC 02-087r3] Open GIS Consortium Inc.: Catalog Services Specification, Version 1.1.1, Date: 2002-12-13, <http://www.opengeospatial.org/specs/?page=specs>
- [Kenn Gardels] Kenn Gardels: The Open GIS Approach to Distributed Geodata and Geoprocessing, http://www.ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/sf_papers/gardels_kenn/ogismodl.html
- [Park et. al.] Joon S. Park and Ravi Sandhu, George Mason University, Gail-Joon Ahn, University of North Carolina at Charlotte: Role-Based Access Control on the Web
- [Richards 1979] I. A. Richards: <http://bradley.bradley.edu/~ell/iarichar.html>
- [Schlichter 2000] Johann Schlichter: Verteiltes Problemlösen, Vorlesung WS2000/2001
- [SecureWeb 2001] SecureWeb: Protecting Web Applications, 2001, <http://elitesecureweb.com/dta/wthreats/sol1.html>
- [SUN 2003] SUN: XACML implementation version 1.1, 2003-11-07, <http://sunxacml.sourceforge.net>, http://sourceforge.net/project/showfiles.php?group_id=73884&package_id=74038&release_id=196336
- [VIVID 2003] VIVID Solutions: Java Topology Suite implementation version 1.4, November 4, 2003, <http://www.vividsolutions.com/jts/jtshome.htm>
- [W3C 2004] World Wide Web Consortium (W3C): Homepage, <http://www.w3c.org>

- [W3C 2001a] W3C: Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001, <http://www.w3.org/TR/wsdl>
- [W3C 2001b] W3C: XML Base W3C Recommendation 27 June 2001, <http://www.w3.org/TR/xmlbase/>
- [W3C 2001c] W3C: XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001 , <http://www.w3.org/TR/xmlschema-0/>
- [W3C 2001d] W3C: XML Schema Part 1: Structures, W3C Recommendation, 2 May 2001 , <http://www.w3.org/TR/xmlschema-1/>
- [W3C 2001e] W3C: XML Schema Part 2: Datatypes, W3C Recommendation, 2 May 2001 , <http://www.w3.org/TR/xmlschema-2/>
- [W3C 2002a] W3C: XML Encryption Syntax and Processing, W3C Proposed Recommendation 03 October 2002, <http://www.w3.org/TR/2002/PR-xmlenc-core-20021003/>
- [W3C 2002b] W3C: XML-Signature Syntax and Processing, W3C Recommendation 12 February 2002, <http://www.w3.org/TR/xmldsig-core/>
- [W3C 2003a] W3C: SOAP, Version 1.2, Date: 24 June 2003
- [W3C 1999] W3C: XML Path Language (XPath), Version 1.0, Date: 16 November 1999
- [W3C 2003b] W3C: XQuery 1.0: An XML Query Language, W3C Working Draft 12 November 2003
- [Wörndl 2003] Wolfgang Wörndl: Privatheit bei dezentraler Verwaltung von Benutzerprofilen, Dissertation, Fakultt für Informatik, Technische Universität München