Testbed-12 Low Bandwidth & Generalization Engineering Report

Table of Contents

Publication Date: 2017-05-12

Approval Date: 2017-02-20

Posted Date: 2016-11-09

Reference number of this document: OGC 16-021r1

Reference URL for this document: http://www.opengis.net/doc/PER/t12-U001

Category: Public Engineering Report

Editor: Benjamin Pross

Title: Testbed-12 Low Bandwidth & Generalization Engineering Report

---

Testbed-12 Low Bandwidth & Generalization Engineering Report (OGC 16-021r1)

COPYRIGHT

WARNING

destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Abstract

For delivering of data that is offered by OGC services over (very) low bandwidth, two options may be considered: On the one hand, the geospatial features remain the same, but compression techniques are used to reduce the size of the data that needs to be transferred. On the other hand, generalization techniques may be used by reducing the details of geometries and/or attributes in order to reduce the amount of data. The aim of this ER is to summarize the results of implementing sample services using compression techniques for DGIWG WFS (U002) and providing generalization processes using WPS (U003). The ER compares the results of the different approaches and infers recommendations and best practices for supporting data delivery of standard data and complex 3D data from OGC services over low and very low bandwidth.

Business Value

The evaluated compression techniques for WFS could lead to further documents like a profile or a best practice document. The WPS Generalization Implementation will be a use case for web based processing. Furthermore, the findings summarized in this ER could lead to a WPS 2.0 profile for generalization.

Technology Value

The Generalization Profile that is described in this ER serves as proof of concept for the WPS 2.0 profiling approach. With the generalization processes a use case for web-based processing is given.

Keywords

ogcdocs, testbed-12, low bandwidth, generalization, wfs, wps

Proposed OGC Working Group for Review and Approval

WPS 2.0 SWG, WFS/FES SWG

Chapter 1. Introduction

1.1. Scope

This ER provides an analysis of the prototype implementations, approaches, test architectures and performance aspects of geospatial data generalization and compression techniques explored in OGC Testbed 12 and findings. OGC Testbed 12 investigated extending WFS with Efficient XML Interchange (EXI) output formats as a method of providing for WFS data delivery in low bandwidth environments. Also it was investigated, how generalization methods can be encapsulated in WPS processes and how WPS profiles for generalization can be specified.

1.2. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Table 1. Contacts

Name	Organization
Benjamin Pross	52°North GmbH
Jeff Harrison	The Carbon Project

1.3. Future Work

See section 10.

1.4. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC 06-121r9, OGC® Web Services Common Standard
- OGC 09-025r2 OGC® Web Feature Service 2.0 Interface Standard
- OGC 14-065 OGC® WPS 2.0 Interface Standard

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9] shall apply. In addition, the following terms and definitions apply.

3.1. Compression

```
Process of reducing the size of data.
```

3.2. Generalization

```
Process of reducing the detail in data.
```

Chapter 4. Conventions

4.1. Abbreviated terms

- API -Application Program Interface
- AOI - Area-of-Interest
- DGIWG - Defense Geospatial Information Working Group
- EXI - Efficient XML Interchange
- FTP - File Transfer Protocol
- GML - Geography Markup Language
- JSON - JavaScript Object Notation
- OGC - Open Geospatial Consortium
- W3C - World Wide Web Consortium
- WFS - Web Feature Service
- WPS - Web Processing Service
- XML - Extensible Markup Language

Chapter 5. Overview

This ER provides an analysis of the prototype implementations, approaches, test architectures and performance aspects of geospatial data generalization and compression techniques explored in OGC Testbed 12 and findings. OGC Testbed 12 investigated extending WFS with Efficient XML Interchange (EXI) output formats as a method of providing for WFS data delivery in low bandwidth environments. Also it was investigated, how generalization methods can be encapsulated in WPS processes and how WPS profiles for generalization can be specified.

First, some background information will be provided, followed by a description of the Low Bandwidth WFS and the Generalization WPS. In the final section, recommendations for future work are given.

Chapter 6. Background

In this chapter, the DGIWG profile as well as compression and generalization techniques are introduced.

6.1. DGIWG - Web Feature Service 2.0 Profile

The DGIWG - Web Feature Service 2.0 Profile document provides recommended implementation profiles for the ISO 19142:2010 Web Feature Service / Open Geospatial Consortium Web Feature Service Interface Standard (WFS) 2.0 – With Corrigendum. The WFS standard provides an interface allowing requests for geospatial features across the web using platform-independent mechanisms and is independent of the underlying data store. One can think of geospatial features as the "source code" behind a map. Whereas the OGC Web Map Service (WMS) interface or online mapping portals return only an image, which end-users cannot edit or spatially analyze, the WFS provides XML-based Geography Markup Language (GML) as the default payload-encoding for transporting geospatial features. In other words, rather than sharing geographic information at the file level using File Transfer Protocol (FTP), for example, the WFS offers direct fine-grained access to geographic information at the feature and feature property level.

The WFS standard specifies discovery operations, query operations, locking operations, transaction operations and operations to manage stored parameterized query expressions. The WFS interface permits users to access and manipulate geospatial feature information from distributed network sources. Technical specifications sometimes have optional features, such that two conforming implementations may not inter-operate completely due to choosing different sets of optional features to support. Even when no formal optional features exist within a standard, there is still a risk that vendors will not implement functionality that is most important to the military community. Also, some standards contain vague or ambiguous wording thus the development and use of profiles can enforce one possible interpretation.

To limit the number of interpretations by implementers and improve interoperability it is possible to define profiles. In standardization, a profile consists of an agreed-upon subset and specific interpretation of a specification. The intention of the DGIWG WFS 2.0 profiles is to minimize such interoperability issues with a specific view to a military context and to mandate a minimum set of service requirements necessary to ensure usability in an operational coalition environment. These profiles are designed to both increase interoperability between WFS servers and to improve the ease of implementation of the WFS standard.

A survey of DGIWG nations was conducted to determine implementation requirements for WFS. These profiles are in response to those survey results. Nations were asked to identify specific requirements for the type of WFS required (Simple, Basic, Transactional, Transactional with Locking, Manage Stored Queries). Based on the results of this survey the profiles define requirements for both a Basic WFS and a Transactional with Locking WFS. The survey also asked respondents to identify requirements for query filters, bindings, bandwidth constraints, output formats and quality of service.

One of the areas for profile assessment was using WFS in low bandwidth environments. This is because results from previous OGC activities and operational deployments indicate that transferring large volumes of geodata from a WFS over a network with poor or very low bandwidth can take a significant amount of time, and network capacity.

To help meet this challenge OGC Testbed 12 developed prototype implementations and conducted Technology Integration Experiments to assess optimizing data transfer under bandwidth-constraint conditions. This document discusses geospatial data size reduction and compression techniques relevant to JSON over GML, zipped XML, EXI etc. Development and testing in Testbed 12 focused on enhancing WFS for EXI compression, with a focus on capabilities that may be considered for possible DGIWG WFS profiling.

6.2. Compression Techniques and Software

The W3C Recommendation Efficient XML Interchange (EXI) Format 1.0 (Second Edition) is a representation for the Extensible Markup Language (XML) Information Set. EXI is intended to optimize performance and the utilization of computational resources. From a practical viewpoint, EXI is designed to reduce the size of XML data exchanged between computers.

EXI uses a grammar-driven approach designed to achieve 'efficient encodings representations'. Consequently, EXI processors are described by the W3C as ‘relatively simple’ and ‘can be implemented on devices with limited capacity.’ An EXI processor is used by application programs to encode their structured data into EXI streams and/or to decode EXI streams to make the data accessible.

EXI is schema ‘informed’, meaning that it can use available schema information to improve compactness and performance. However, the W3C indicates that EXI does not depend on accurate, complete or current schemas to work – a statement which must be considered carefully when using EXI for geographic feature data.

6.2.1. W3C EXI Documents

Despite W3C statements that EXI processors are ‘relatively simple’, Efficient XML Interchange (EXI) is a very complex topic. The reader is encouraged to review the following W3C documents for a complete background -

- Efficient XML Interchange (EXI) Format 1.0 (Second Edition), http://www.w3.org/TR/2014/REC-exi-20140211/Efficient The latest version is available at http://www.w3.org/TR/exi/.
- Efficient XML Interchange (EXI) Best Practices, http://www.w3.org/TR/2007/WD-exi-best-practices-20071219/

6.3. Schema-informed vs Schema-less Compression

Review of the references above indicates the W3C describes EXI as not 'dependent on schemas'. However, prior investigations have assessed that EXI may compress XML more efficiently if schemas exist describing the format of the expected XML. As background, it is important to understand there are two main ways in which EXI encodes XML documents -

- **Schema-less** - In the schema-less mode, EXI encodes an XML document whether or not a schema is available to the encoder.
- **Schema-informed** - In the schema-informed mode, EXI encoding can utilize available schema

information to improve compactness and performance, but does not depend on accurate, complete or current schemas to work.

EXI uses a set of built-in grammars to encode XML documents and XML fragments when no schema information is available.

The two modes, schema-less and schema-informed, are important for GML and WFS compression because prior reports noted that coordinates in GML 2 may be defined in a schema as *string* ('text'). Since schema optimization cannot improve on 'text' this may result in poor compression when there are many coordinates in GML data described by a GML 2 schema. However, the report noted that coordinates in GML 3 may be defined in a schema as *float*. Schema optimization can improve on *float*, with the results being good compression with lots of coordinates. The key point being that compression performance may be dependent on the design of the schema as well as the XML data itself.

6.3.1. EXI Streams

EXI represents the contents of an XML document as an **EXI stream.** An EXI stream consists of an EXI header followed by an EXI body.

The EXI header conveys format version information and may also include the set of options that were used during encoding. If these options are omitted, it is assumed that the decoder has access to them out of band.

The EXI body comprises an event sequence describing the document (or document fragment) that is encoded.

6.3.2. EXI Option Values

In addition to the different compression performance that may be obtained with or without schemas, different types **option values** may be used while encoding XML documents in an EXI stream. Option values are part of the EXI header and provide a way to specify the options used to encode the body of an EXI stream. There are many option values outlined in the W3C EXI specification and the reader is again encouraged to review them as needed.

The most significant option values for Testbed 12 WFS Compression testing are presented and defined in the following table.

Table 2. EXI Options Values

Option Value	Description
BIT_PACKED	If the alignment option value is bit packed, that indicates that event codes and associated content are packed in bits without any padding in-between.

Option Value	Description
BYTE_PACKED	The alignment option value byte-alignment indicates that the event codes and associated content are aligned on byte boundaries. While byte-alignment generally results in EXI streams of larger sizes compared with their bit-packed equivalents, byte-alignment may provide a help in some use cases that involve frequent copying of large arrays of scalar data directly out of the stream. It can also make it possible to work with data in-place and can make it easier to debug encoded data by allowing items on aligned boundaries to be easily located in the stream.
PRE_COMPRESSION	This alignment option value indicates that all steps involved in compression are to be done with the exception of the final step of applying the DEFLATE algorithm.
COMPRESSION	This compression option is used to increase compactness using additional computational resources (via DEFLATE algorithm).

6.3.3. Compression Software for WFS

In OGC Testbed 12 EXI participants extended WFS with software capable of producing an output format in EXI. The software tested in WFS implementations included the packages listed in the following table.

Table 3. EXI Compression Software

EXI Software	Description
Nagasena	Nagasena is an implementation of the EXI specification, available both for Java and .Net platforms. http://openexi.sourceforge.net/
EXIficient	EXIficient is a set of implementations of the EXI format specification available for Java, Javascript, C/C++. http://exificient.github.io/
OSS	OSS is an implementation of the EXI specification, available both for Java and .Net platforms. http://www.oss.com/xml/products/exi-c/exi-c.html

6.4. Generalization Techniques

Generalization in GIS is used to reduce the detail in data. With this reduction in detail, also the amount of data can be reduced, thus improving the usage in low bandwidth environments. [1] lists the following twelve categories of operators for cartographic generalization:

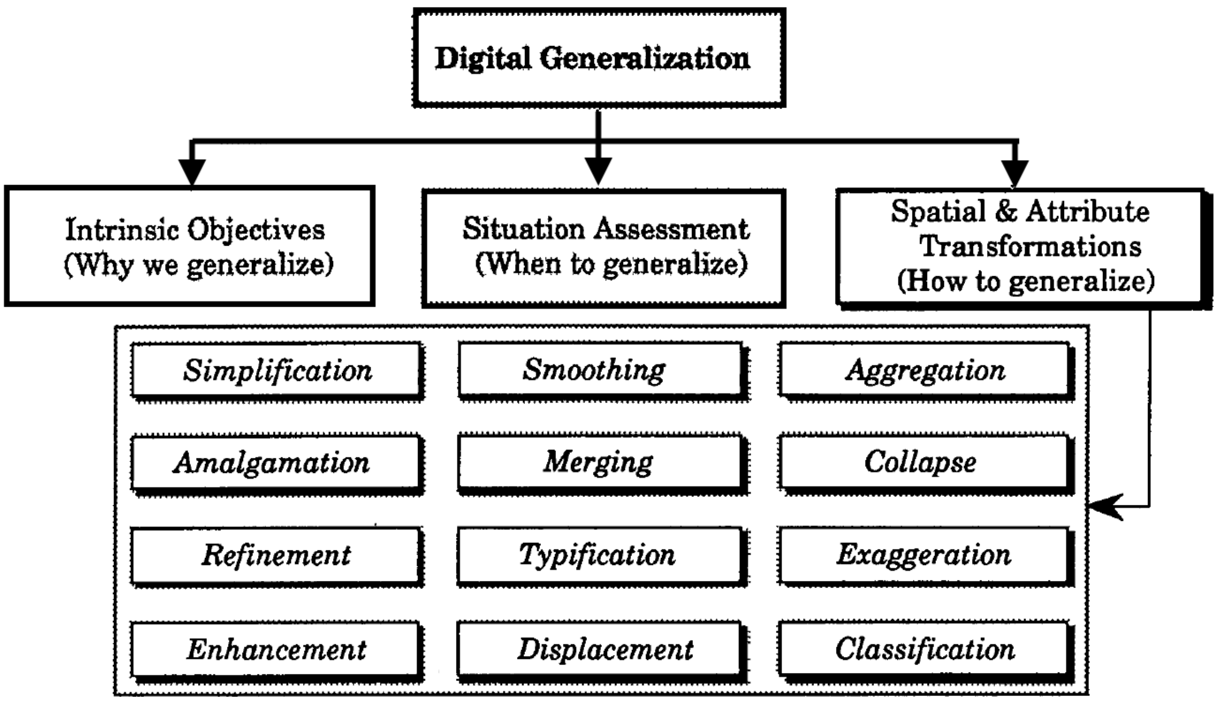
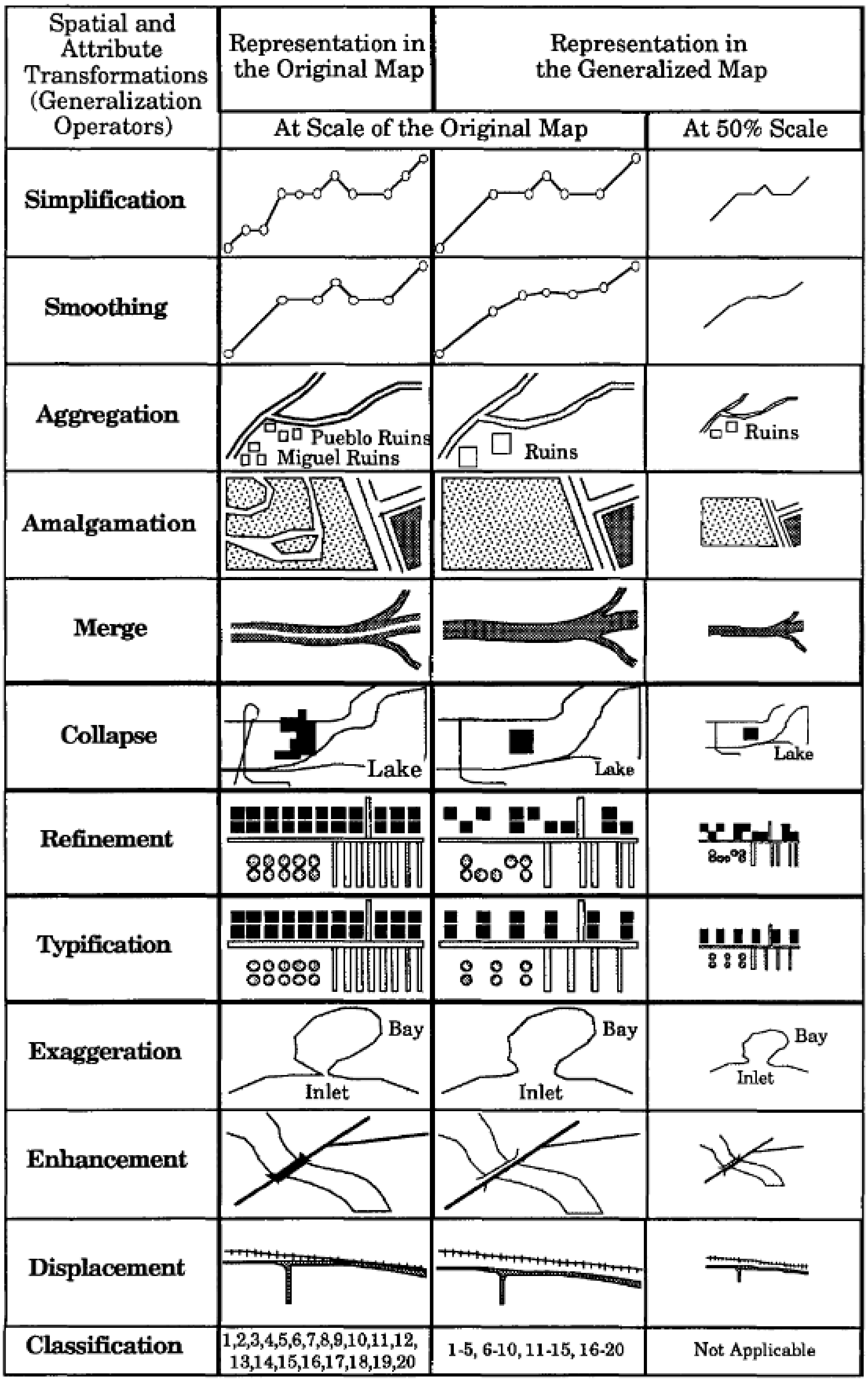


Figure 1. Twelve categories of generalization operators. (source [1])

Examples for the twelve operators are shown in the following image:



Spatial and Attribute Transformations (Generalization Operators)	Representation in the Original Map	Representation in the Generalized Map	
	At Scale of the Original Map		At 50% Scale
Simplification			
Smoothing			
Aggregation	Pueblo Ruins Miguel Ruins	Ruins	Ruins
Amalgamation			
Merge			
Collapse	Lake	Lake	Lake
Refinement			
Typification			
Exaggeration	Bay Inlet	Bay Inlet	Bay Inlet
Enhancement			
Displacement			
Classification	1,2,3,4,5,6,7,8,9,10,11,12, 13,14,15,16,17,18,19,20	1-5, 6-10, 11-15, 16-20	Not Applicable

Figure 2. Sample spatial and attribute transformations of cartographic generalization. (source [1])

There are three types of geometry for that generalization techniques will be investigated in this ER: point, line and polygon. Several techniques can be applied to each of the geometry types. We chose to investigate the following operations further:

- Aggregation/Merging: A digitized representation of a map feature should be accurate in its representation of the feature (shape, location, and character), yet also efficient in terms of retaining the least number of data points necessary to represent the character. A profligate density of coordinates captured in the digitization stage should be reduced by selecting a subset of the original coordinate pairs, while retaining those points considered to be most representative of the line (Jenks, 1981). Glitches should also be removed. Simplification operators will select the characteristic, or shape-describing, points to retain, or will reject the redundant point considered to be unnecessary to display the line's character. Simplification operators produce a reduction in the number of derived data points which are unchanged in their x,y coordinate positions. Some practical considerations of simplification includes reduced plotting time, increased line crispness due to higher plotting speeds, reduced storage, less problems in attaining plotter resolution due to scale change, and quicker vector to raster conversion (see [2]).
- Simplification: There are many instances when the number or density of like point features within a region prohibits each from being portrayed and symbolized individually within the graphic. This notwithstanding, from the perspective of the map's purpose, the importance of those features requires that they still be portrayed. To accomplish that goal, the point features must be aggregated into a higher order class feature areas and symbolized as such. For example, if the intervening spaces between houses are smaller than the physical extent of the buildings themselves, the buildings can be aggregated and re-symbolized as built-up areas (see [3]).
- Refinement/Elimination: In many cases, where like features are either too numerous or too small to show to scale, no attempt should be made to show all the features. Instead, a selective number and pattern of the symbols are depicted. Generally, this is accomplished by leaving out the smallest features, or those which add little to the general impression of the distribution. Though the overall initial features are thinned out, the general pattern of the features is maintained with those features that are chosen by showing them in their correct locations. Excellent examples of this can be found in [4]. This refinement process retains the general characteristics of the features at a greatly reduced complexity.

The implementation of the three operations is described in section 8.

Chapter 7. Low Bandwidth WFS

This section describes a WFS fitted for low bandwidth environments.

7.1. WFS Compression Experiments

In OGC Testbed 12 EXI participants investigated compression techniques for geospatial data sets delivered by WFS Servers and Clients by augmenting WFS with software capable of producing an output format in EXI (described above).

The testing architecture for WFS Compression in OGC Testbed 12 was configured using a combination of the following components -

- **EXI Pre-Processors and Processors** - Software program modules used by application programs to encode their structured data into EXI streams and/or to decode EXI streams to make the structured data accessible.
- **Compression WFS** - WFS augmented with EXI Pre-Processors and Processors and loaded with test data. Provides the ability to request test data as GML, GeoJSON, GZIP and EXI (among other output formats).
- **Compression WFS Clients** - Application clients with the ability to request EXI encoded data from a Compression WFS, with a performance recording module to gather metrics on time taken to perform the encoding and, most importantly, size of the resulting EXI stream. Includes the ability to decode EXI streams from Compression WFS.

These components were configured for testing Compression WFS and EXI as described in the following sequence diagram –

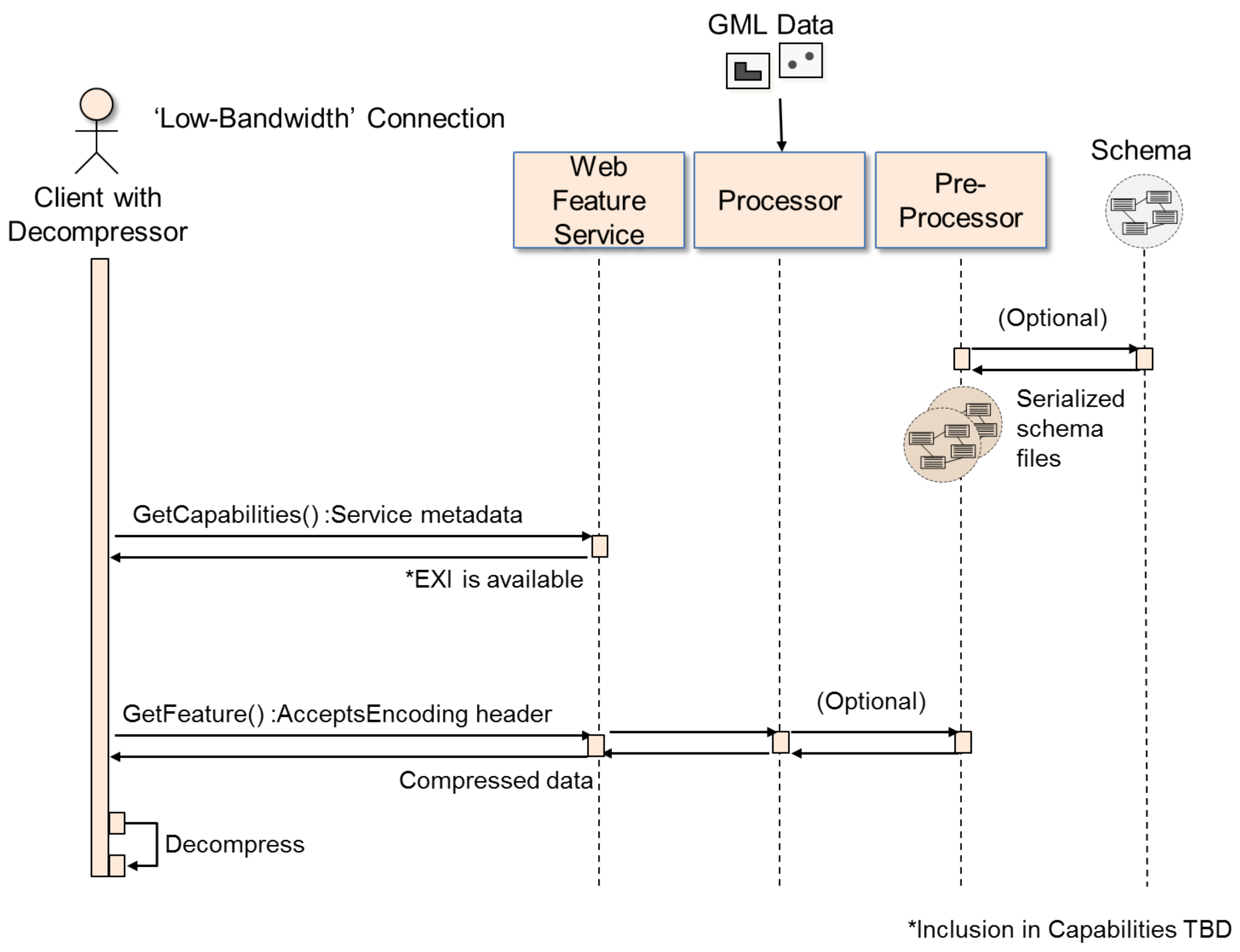
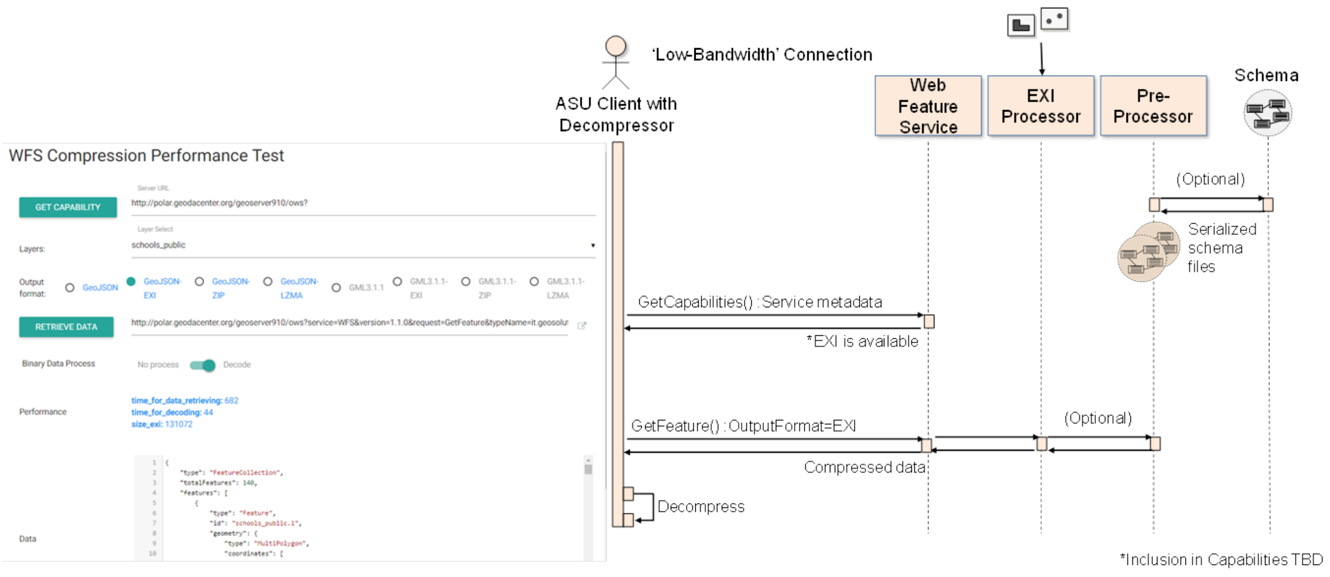


Figure 3. Sequence diagram for testing Compression with WFS and EXI

7.1.1. Test Suite 1

For compression testing implemented Compression Tests WFS Servers, Compression Clients, EXI Pre-Processors, EXI Processors in the following architecture -


WFS Compression Performance Test

GET CAPABILITY

Server URL: http://polar.geodacenter.org/geoserver910/ows?

Layers: Layer Select: schools_public

Output format: GeoJSON [✓ GeoJSON-EXI] GeoJSON-ZIP GeoJSON-LZMA GML3.1.1 GML3.1.1-EXI GML3.1.1-ZIP GML3.1.1-LZMA

RETRIEVE DATA

http://polar.geodacenter.org/geoserver910/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=it.geosolu

Binary Data Process: No process [✓ Decode]

Performance:
time_for_data_retrieving: 682
time_for_decoding: 44
size_exi: 131072

Data:
```
{
    "type": "FeatureCollection",
    "totalFeatures": 140,
    "features": [
        {
            "type": "Feature",
            "id": "schools_public.1",
            "geometry": {
                "type": "MultiPolygon",
                "coordinates": [
```
'Low-Bandwidth' Connection
ASU Client with Decompressor
Web Feature Service
EXI Processor
Pre-Processor
Schema
(Optional)
Serialized schema files
GetCapabilities() :Service metadata
*EXI is available
GetFeature() :OutputFormat=EXI
(Optional)
Compressed data
Decompress
*Inclusion in Capabilities TBD

Figure 4. Compression Tests Architecture and Sequence Diagram

The Compression WFS in Test Suite 1 was based on WFS, extended with OSS for .NET on the server and client, and EXIeffienct using Java and Javascript wrappers on the server side as needed. Compression software was employed for encoding plain-text based GML and GeoJSON content into binary EXI files on the server side. The data preparation process is: features -→ XML-based data stream (GML) -→ Encoding into .exi file -→ Transfer to client application for decoding and rendering.

Additional types of compressing methods were implemented on the server including GZIP and LZMA(.7z) for comparison.

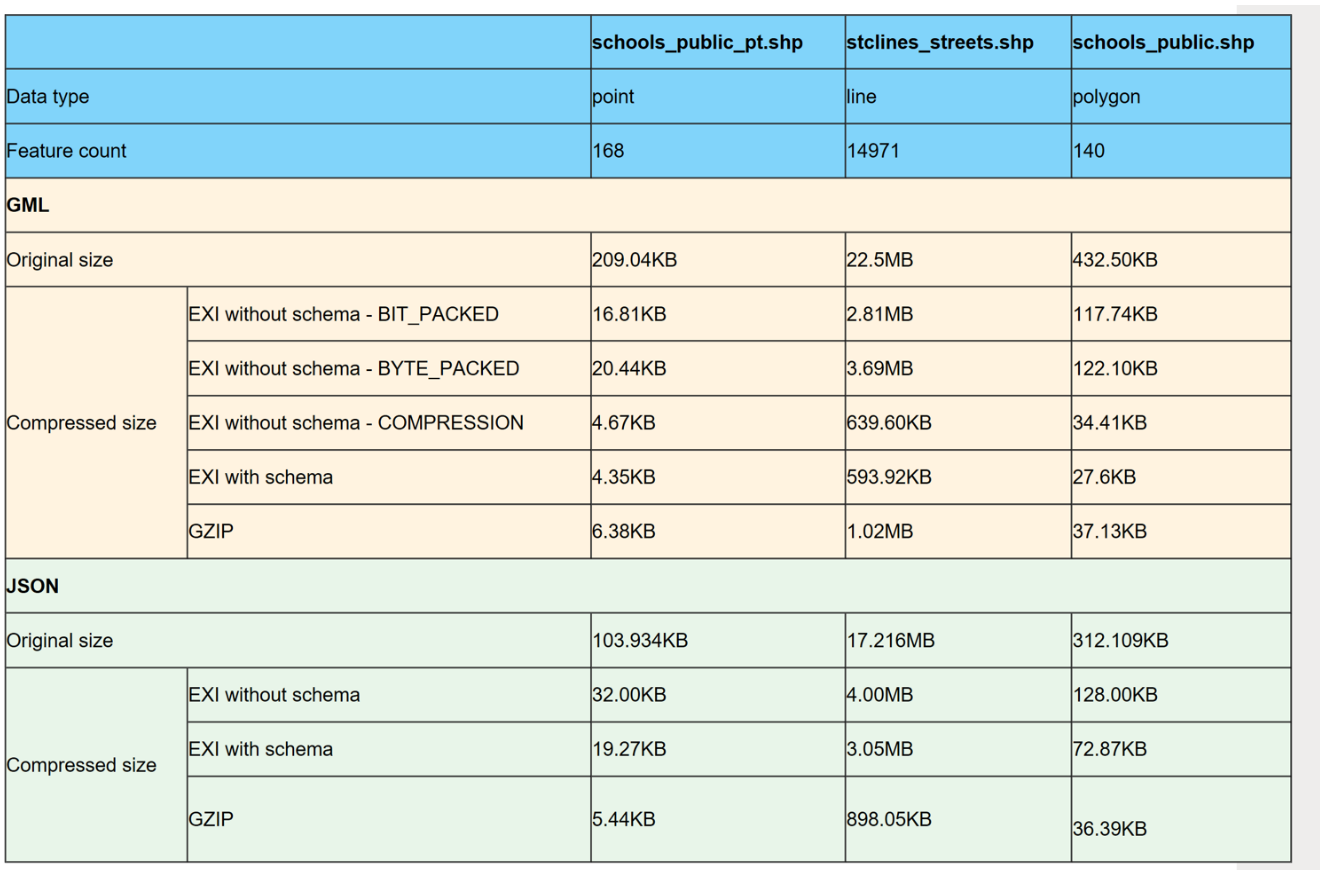
For compression testing processors implemented both Schema-less and schema-informed modes.

Feature data over San Francisco representing points (schools_public_pt.shp), lines (stclines_streets.shp) and polygons (schools_public.shp) formed the test baseline. Other data sets were assessed as well.

Performance recording modules were implemented for comparison of EXI performance. Three formats are supported, including GeoJSON -LZMA, GML3.1.1-EXI and GML3.1.1-LZMA. Browser side data decompressing functions were implemented with the format of GeoJSON -ZIP, GeoJSON -LZMA, GML3.1.1-ZIP and GML3.1.1-LZMA realized.

Participants attempted to integrate Nagasena into Compression WFS but did not continue the TIE due to performance issues.

Using the performance recording module information about different compression methods and datasets were developed. Initial test results for Compression WFS on Test Suite 1 are presented in the table below.

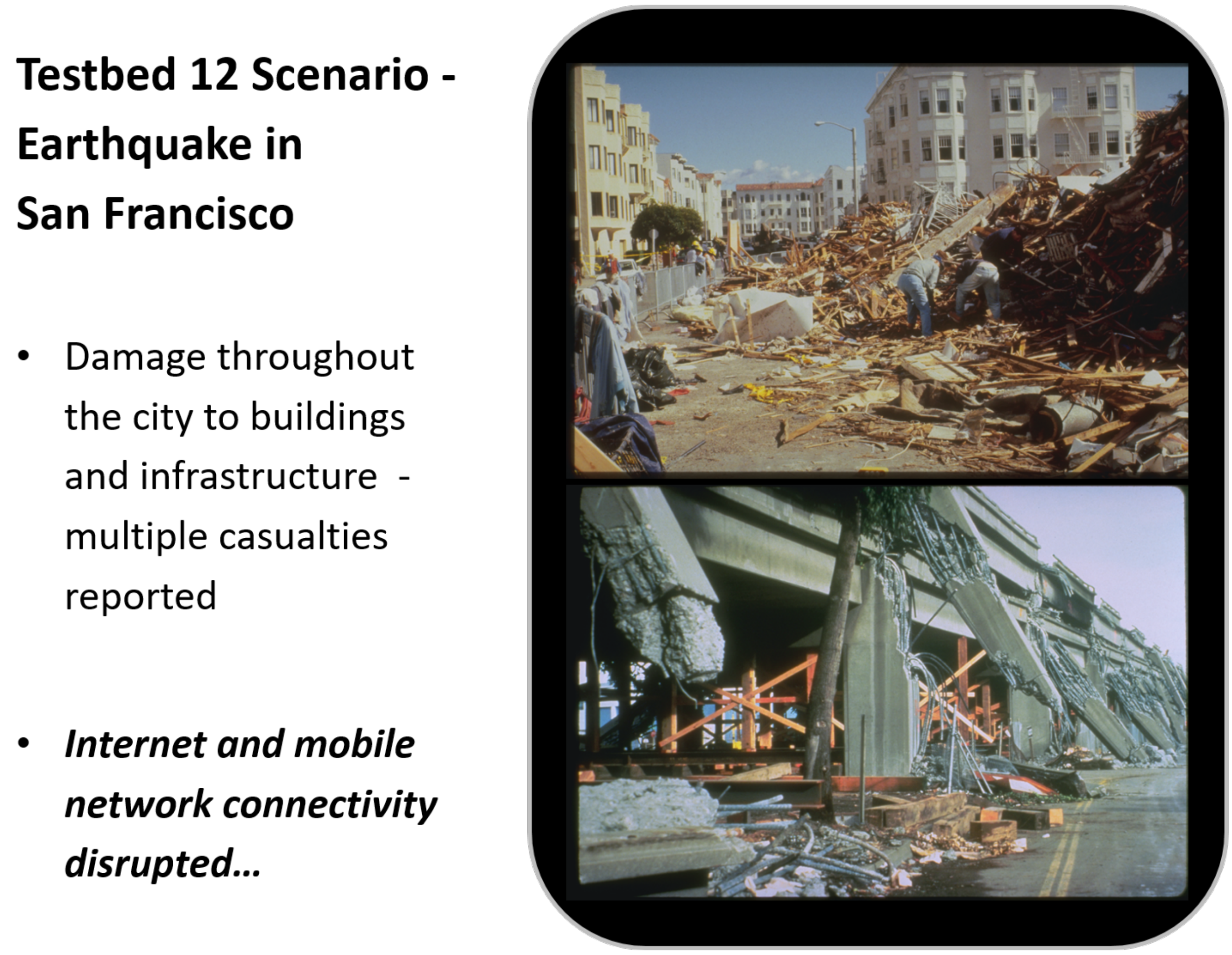


		schools_public_pt.shp	stclines_streets.shp	schools_public.shp
Data type		point	line	polygon
Feature count		168	14971	140
GML				
Original size		209.04KB	22.5MB	432.50KB
Compressed size	EXI without schema - BIT_PACKED	16.81KB	2.81MB	117.74KB
	EXI without schema - BYTE_PACKED	20.44KB	3.69MB	122.10KB
	EXI without schema - COMPRESSION	4.67KB	639.60KB	34.41KB
	EXI with schema	4.35KB	593.92KB	27.6KB
	GZIP	6.38KB	1.02MB	37.13KB
JSON				
Original size		103.934KB	17.216MB	312.109KB
Compressed size	EXI without schema	32.00KB	4.00MB	128.00KB
	EXI with schema	19.27KB	3.05MB	72.87KB
	GZIP	5.44KB	898.05KB	36.39KB

Figure 5. ASU Final Results

7.2. Low Bandwidth WFS Test Scenario

```
Prototype implementations, various approaches, test architectures and performance
aspects of geospatial data compression techniques explored in OGC Testbed 12 were
assessed in a simulated disaster response scenario. This scenario, and relevant
aspects of WFS in low bandwidth environments are described in the following graphics.
```



Testbed 12 Scenario - Earthquake in San Francisco

- Damage throughout the city to buildings and infrastructure - multiple casualties reported
- ***Internet and mobile network connectivity disrupted…***

Figure 6. Testbed 12 Scenario - Earthquake in San Francisco

Earthquake in San Francisco

- Response and recovery operations started – relief resources begin moving in
- Command Center established to coordinate efforts…

Figure 7. Earthquake in San Francisco - Command Center (Humvees)

Earthquake in San Francisco

- Updates Common Operating Picture needed throughout day
- Geospatial data needs to be sent from Command Center to multiple Humvees
- *Bandwidth limited*

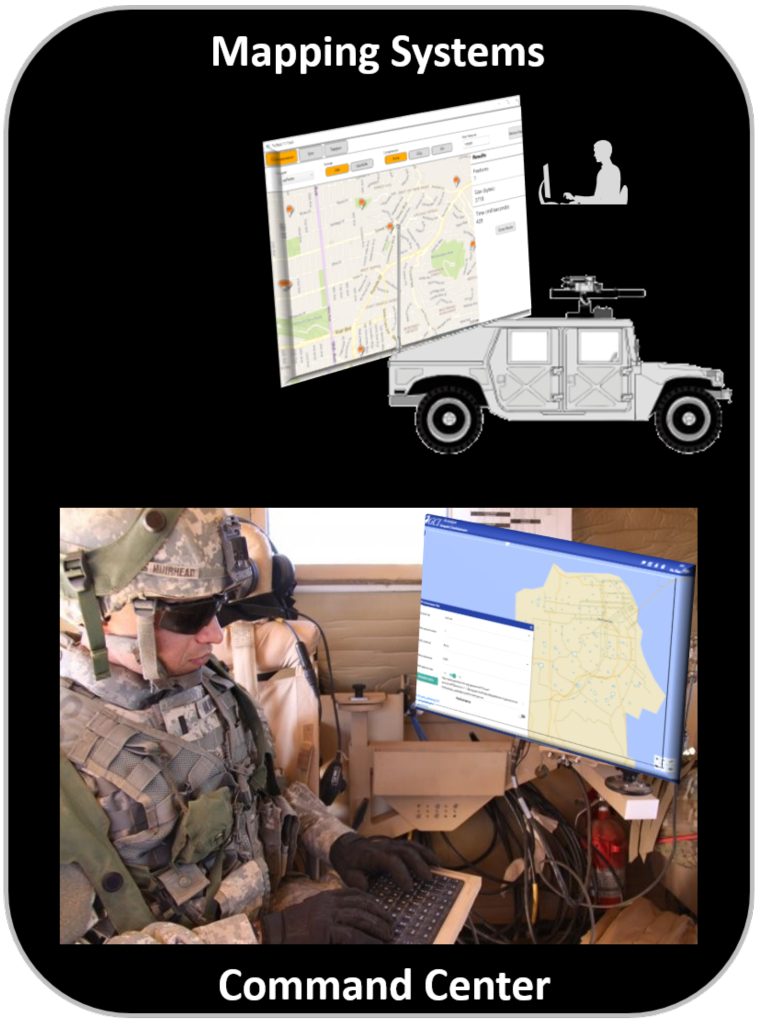
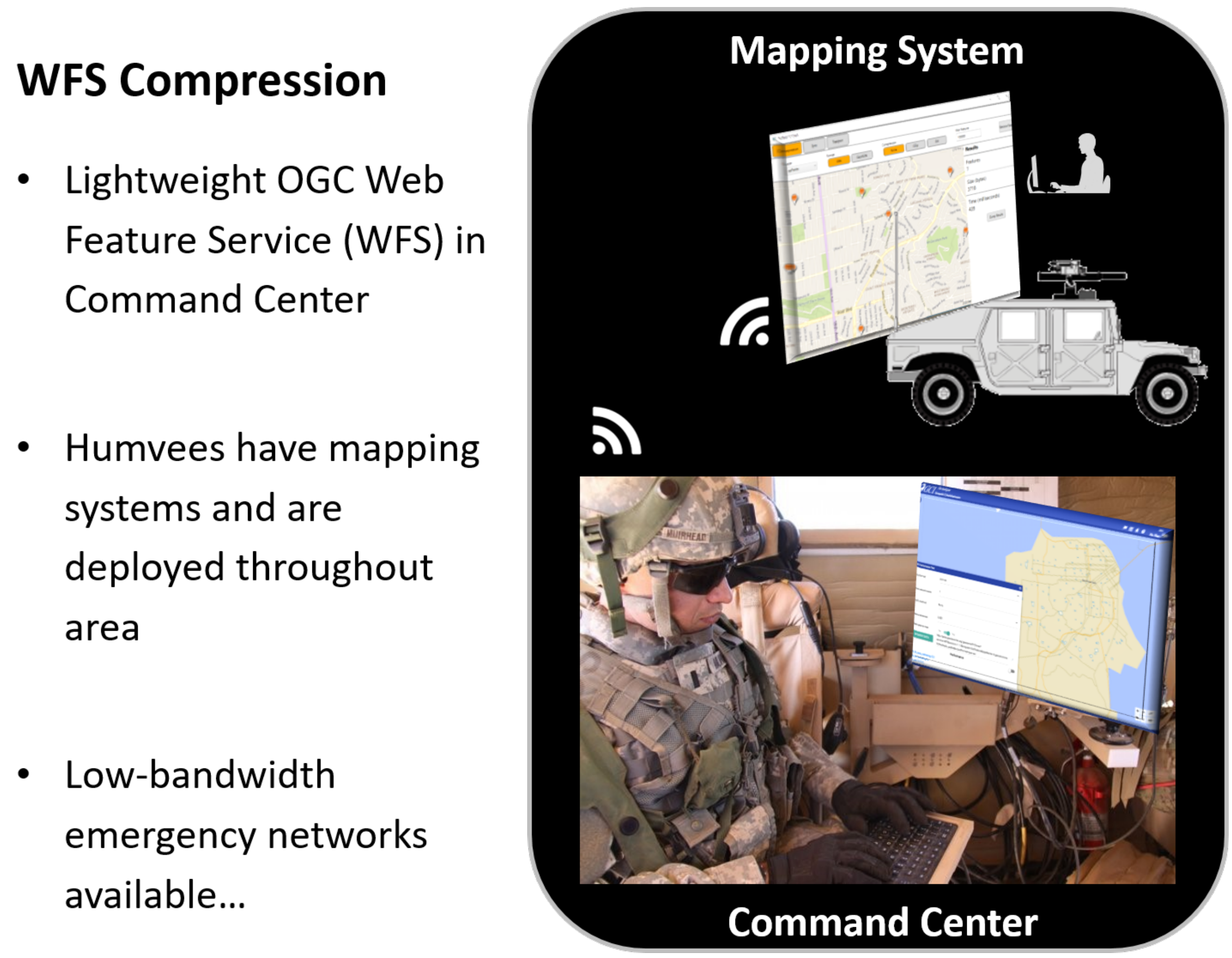


Figure 8. Earthquake in San Francisco - Command Center (Mapping Systems)

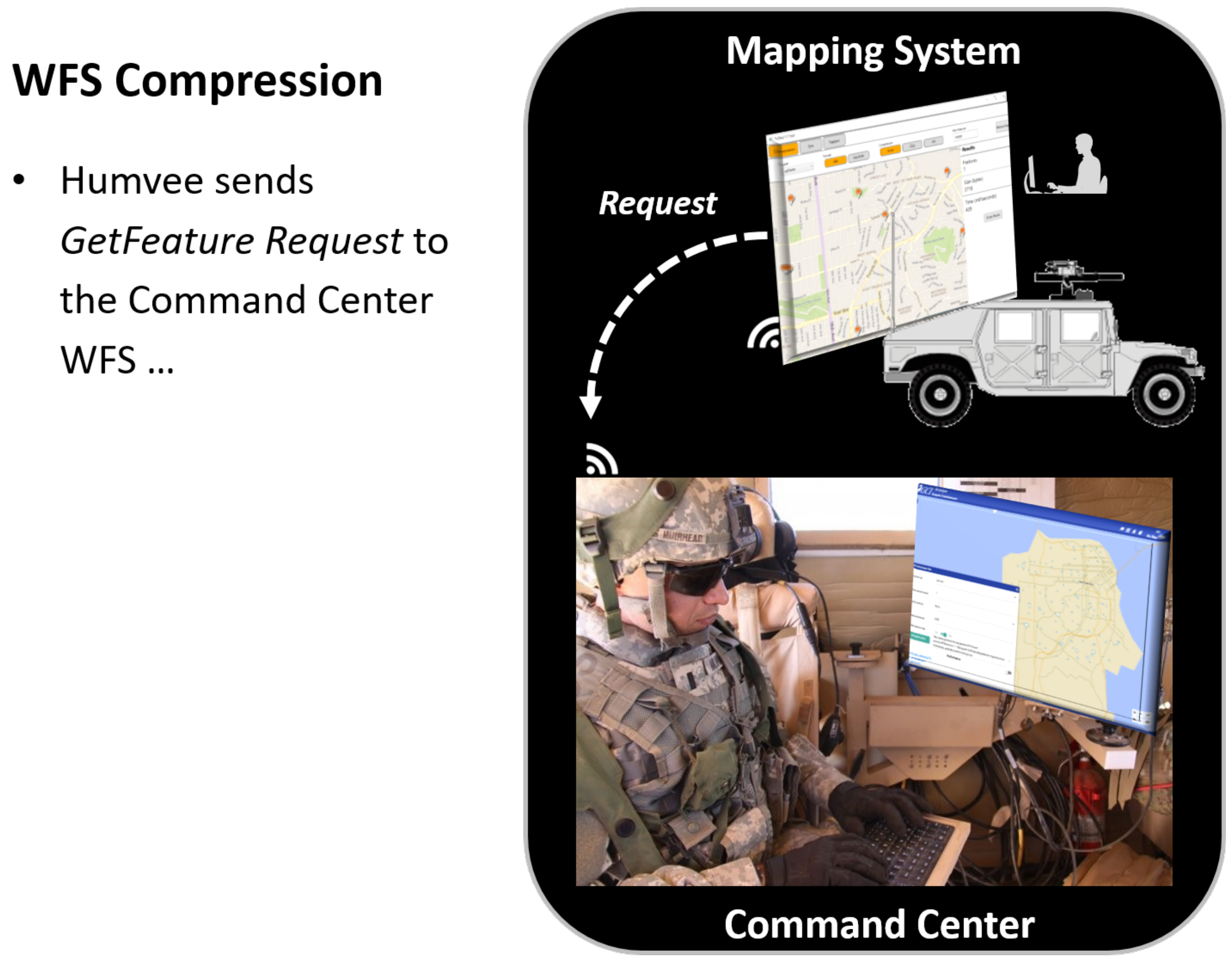


WFS Compression

- Lightweight OGC Web Feature Service (WFS) in Command Center
- Humvees have mapping systems and are deployed throughout area
- Low-bandwidth emergency networks available…

Mapping System
Command Center

Figure 9. WFS Compression - Command Center (Mapping Systems)


WFS Compression

- Humvee sends *GetFeature Request* to the Command Center WFS …

Figure 10. WFS Compression (GetFeature Request) - Command Center (Mapping Systems)

WFS Compression

- Humvees get latest geospatial data… compressed so it's easier to transmit across low bandwidth network

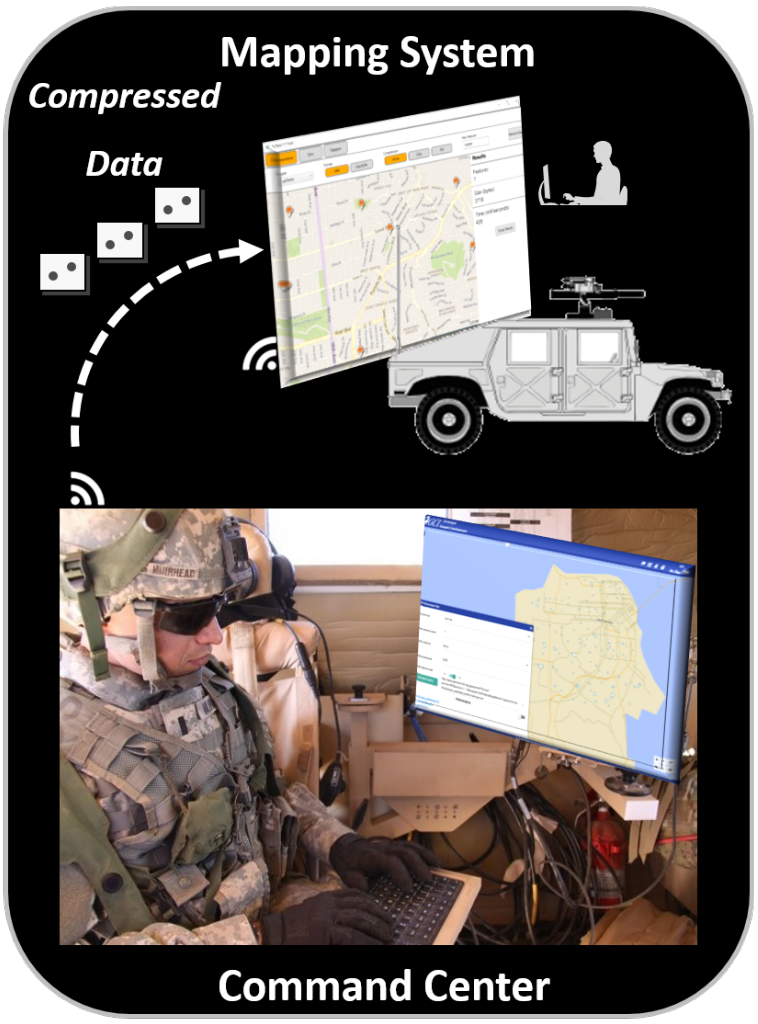
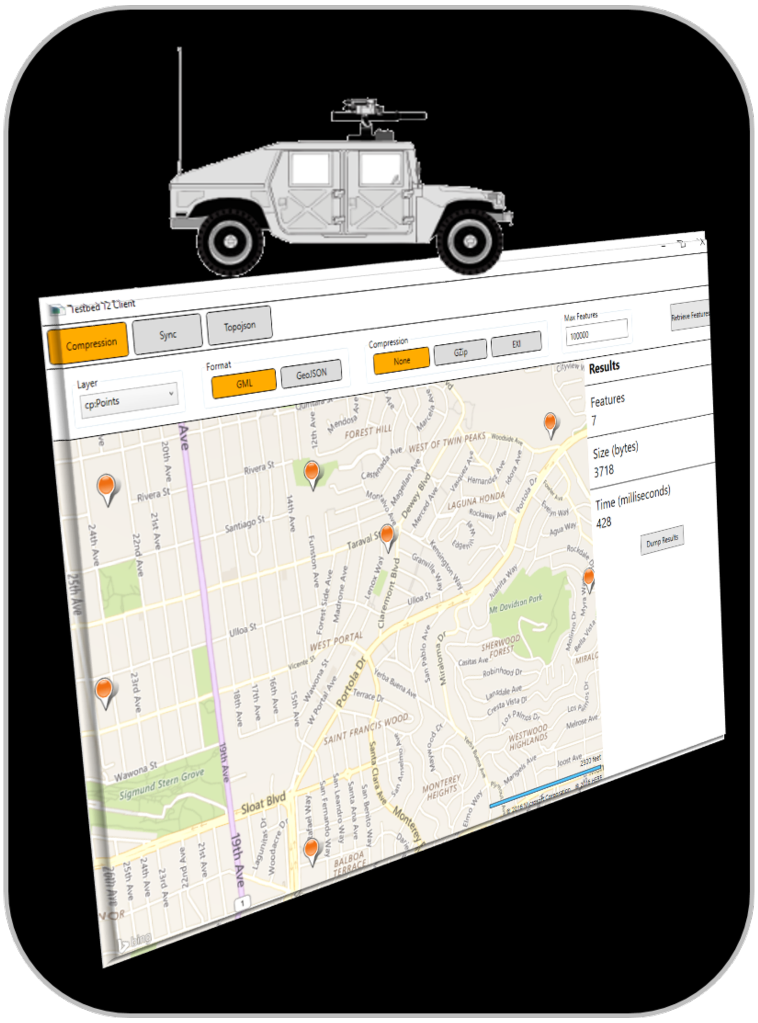


Figure 11. WFS Compression (Receive compressed data) - Command Center (Mapping Systems)

WFS Compression

- Command Center uses WFS Compression tools to select best methods, such as EXI or GZIP to transmit the updates…
- Carbon Project Compression WFS…



Layer: cp:Points
Format: GML
Compression: None
Max Features: 100000

Results
Features: 7
Size (bytes): 3718
Time (milliseconds): 428

Figure 12. WFS Compression - Select best compression method

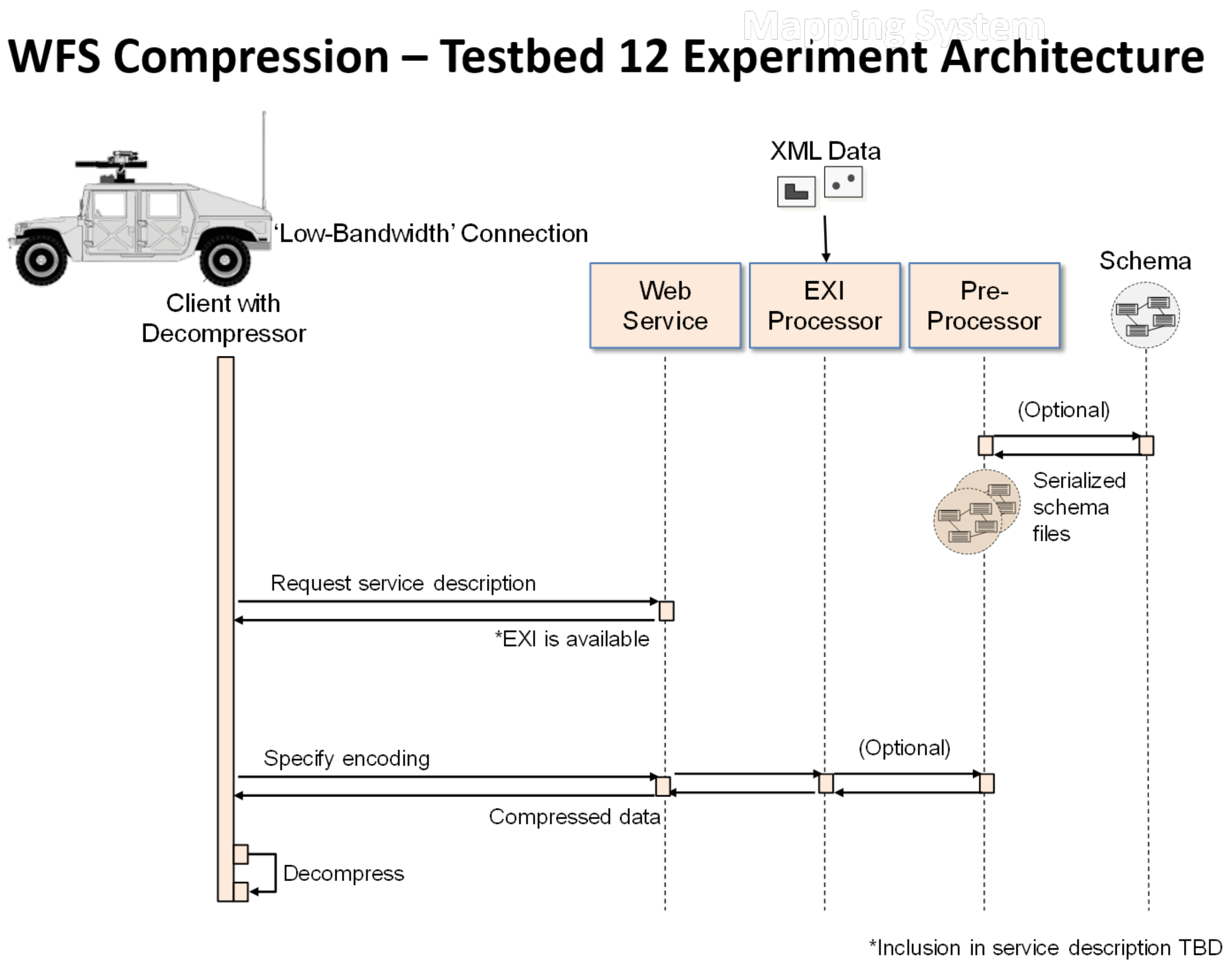


Figure 13. WFS Compression - Testbed 12 Experiment Architecture

7.3. Findings

Technology Integration Experiments conducted in Testbed 12 indicate -

1. It is possible for EXI on a Compression WFS to produce an encoding of GML that is smaller than a GZIP of the same data.
2. It is possible for EXI on a Compression WFS to produce an encoding of GML that is larger than a GZIP of the same data.
3. In no circumstances was the EXI compression of GML as efficient as that noted for tactical data formats in other studies.
4. EXI on a Compression WFS is able to encode GML whether or not a schema is available to the encoder.
5. On average, using EXI on a Compression WFS results in a GML file that is approximately 20 percent smaller than a GZIP of the same data if the schema is available to the encoder, but the numbers may be wrongly decoded. For EXificient, if schema is not used but "Compression" is selected as the coding mode, the results will be correct. And EXI file can also be smaller than a GZIP file by this method.
6. If the XML file is encoded with schema by Java code at the server side, it could not be correctly decoded at the client side using Javascript.

7. The schema for each FeatureType on a WFS must be preprocessed on the server. And the schemas must declare 'float' data types. Some do(GML 3), and some do not (GML 2).

8. If specified, the compressing work will be done at the server side by Tomcat and decompressed at the browser side by browsers like Chrome or Firefox. However, only the GZIP format is widely supported by both server and browsers. Other formats like EXI are not supported and require custom applications to function.

9. The WFS specification does not specifically discuss requesting compressed output. However, the WFS specification does normatively reference the HTTP specification which says the Accept-Encoding header can be used by the client to request compressed output. Having said that, some servers advertise vendor-specific outputFormat values for requesting compressed output via that parameter. What this means is Testbed 12 has identified an area for potential clarification in the WFS specification. Note - the WFS 2.5 specification allows both approaches (i.e. Accept header and outputFormat parameter) for negotiating the response representation and encoding.

10. If the schema is used on the server during the encoding, it is required during the decoding process on the client. This means that for each FeatureType on a WFS the GML schema needs to be present on the server and each client that wishes to use EXI. For example, the schema of an XML file usually describes the meaning and data type of the fields and attributes included in the XML file. For EXI processing, an XML file can be encoded with or without the schema file. If without schema file, the process will be a pure EXI-encoding: from plain-text into binary, and the file size could decrease since the resulting EXI file is binary. If with schema file, then the process will be EXI-encoding and compressing. Since the process could extract some commonly used strings from the schema file and use them for compressing during the EXI-encoding, consequently the result file informed by schema may be smaller than schema-less. In the other way, if the schema is used during the encoding, it is required during the decoding.

Chapter 8. WPS Generalization

This section describes a WPS 2.0 for Generalization. Generalization algorithms for the three geometry types point, line and polygon will be described. The algorithms were implemented based on the 52°North WPS 2.0 framework [1: https://github.com/52North/WPS-Extension-Skeleton/tree/project/testbed-12]. The algorithms were modified from the WebGen WPS project [2: https://github.com/TUD-IfC/WebGen-WPS], a WPS 1.0.0 implementation of generalization algorithms created by the Technical University of Dresden together with the International Cartographic Association (ICA) Commission on Generalisation and Multiple Representation [3: http://generalisation.icaci.org/]. The WPS processes implement one or more profiles that will be described in section 9. For testing the algorithms, datasets with different geometry types from the National Hydrography Dataset [4: http://nhd.usgs.gov/data.html] were used.

8.1. Point Generalization

8.1.1. Concept

For point generalization the following method was chosen: The points are merged by attribute and optional additional by distance. If the specified attribute of two points is equal (and optionally if they fall in the specified distance), a union of the points is done and the resulting point is added to the result set.

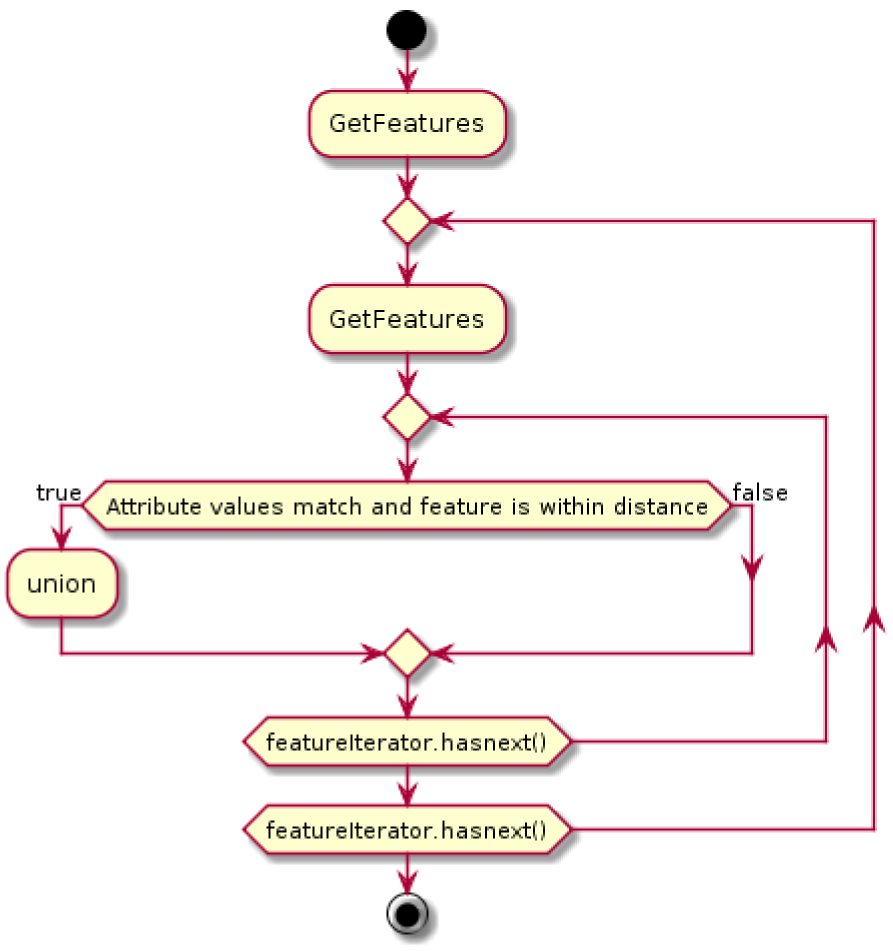

GetFeatures
GetFeatures
true
Attribute values match and feature is within distance
false
union
featureIterator.hasnext()
featureIterator.hasnext()

Figure 14. Activity diagram showing point generalization

8.1.2. Implementation

The following listing shows the algorithm:

Java-based MergeFeaturesByAttribute algorithm

```
        Quadtree qtree = new Quadtree();
        for (FeatureIterator<?> is = fc.features(); is.hasNext();) {
            SimpleFeature fs = (SimpleFeature) is.next();
            Geometry gs = (Geometry) fs.getDefaultGeometry();
            qtree.insert(gs.getEnvelope().getEnvelopeInternal(), fs);
        }

        int fcsize = fc.size();
        int i = 1;
        for (FeatureIterator<?> is = fc.features(); is.hasNext();) {
            LOGGER.info("processing feature " + i + " of " + fcsize);
            SimpleFeature fs = (SimpleFeature) is.next();
            String as = fs.getAttribute(classfield).toString().trim();
            Geometry gs = (Geometry) fs.getDefaultGeometry();
            Envelope genv = gs.getEnvelope().getEnvelopeInternal();
            if (qtree.remove(genv, fs)) {
                LOGGER.info("Envelope: " + genv);
                List<SimpleFeature> inEnvelope = qtree.query(genv);
                LOGGER.info("Checking features: " + inEnvelope.size());
                for (Iterator<SimpleFeature> iin = inEnvelope.iterator();
iin.hasNext();) {
                    SimpleFeature fiin = (SimpleFeature) iin.next();
                    Geometry giin = (Geometry) fiin.getDefaultGeometry();
                    if (as.equals(fiin.getAttribute(classfield).toString().trim()) &&
(gs.distance(giin) <= distance)) {
                        gs = gs.union(giin);
                        qtree.remove(((Geometry)
fiin.getDefaultGeometry()).getEnvelope().getEnvelopeInternal(), fiin);
                    }
                }
                fs.setDefaultGeometry(gs);
                qtree.insert(gs.getEnvelope().getEnvelopeInternal(), fs);
            }
            i++;
        }
```

The WPS 2.0.0 process description is shown in the following listing:

MergeFeaturesByAttribute process description

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:ProcessOfferings xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xlin="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/2.0
```

```
http://schemas.opengis.net/wps/2.0/wps.xsd">
  <wps:ProcessOffering processVersion="0.0.1" jobControlOptions="sync-execute async-
execute" outputTransmission="value reference">
    <wps:Process>
      <ows:Title>testbed12.fo.MergeFeaturesByAttribute</ows:Title>
      <ows:Identifier>testbed12.fo.MergeFeaturesByAttribute</ows:Identifier>
      <ows:Metadata xlin:role="http://www.opengis.net/spec/wps/2.0/def/process-
profile/concept" xlin:href="http://52north.github.io/wps-
profileregistry/concept/generalization.html"/>
      <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/merge-features-
by-attribute.html"/>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>distance</ows:Title>
        <ows:Identifier>distance</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/merge-features-
by-attribute.html#distance"/>
        <ns:LiteralData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="text/plain"/>
          <ns:Format mimeType="text/xml"/>
          <LiteralDataDomain>
            <ows:AnyValue/>
            <ows:DataType ows:reference="xs:double"/>
          </LiteralDataDomain>
        </ns:LiteralData>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>classfield</ows:Title>
        <ows:Identifier>classfield</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/merge-features-
by-attribute.html#classfield"/>
        <ns:LiteralData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="text/plain"/>
          <ns:Format mimeType="text/xml"/>
          <LiteralDataDomain>
            <ows:AnyValue/>
            <ows:DataType ows:reference="xs:string"/>
          </LiteralDataDomain>
        </ns:LiteralData>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>data</ows:Title>
        <ows:Identifier>data</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/merge-features-
```

```
by-attribute.html#data"/>
        <ns:ComplexData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="application/vnd.google-earth.kml+xml"
schema="http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd"/>
          <ns:Format mimeType="application/x-zipped-shp"/>
          <ns:Format mimeType="application/x-zipped-shp" encoding="base64"/>
          <ns:Format mimeType="application/x-zipped-wkt" encoding="Base64"/>
          <ns:Format mimeType="application/vnd.geo+json"/>
          <ns:Format mimeType="text/json"/>
          <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
        </ns:ComplexData>
      </wps:Input>
      <wps:Output>
        <ows:Title>result</ows:Title>
        <ows:Identifier>result</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/merge-features-
by-attribute.html#result"/>
        <ns:ComplexData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="application/vnd.google-earth.kml+xml"
schema="http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd"/>
          <ns:Format mimeType="application/x-zipped-shp"/>
          <ns:Format mimeType="application/x-zipped-shp" encoding="base64"/>
          <ns:Format mimeType="text/xml"/>
          <ns:Format mimeType="text/xml" encoding="base64"/>
          <ns:Format mimeType="text/xml"
schema="http://schemas.opengis.net/gml/2.0.0/feature.xsd"/>
          <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
          <ns:Format mimeType="application/vnd.geo+json"/>
        </ns:ComplexData>
      </wps:Output>
    </wps:Process>
  </wps:ProcessOffering>
</wps:ProcessOfferings>
```

8.1.3. Results

This method was tested with the NHDPoints dataset, the attribute FType and a distance of 0.005 decimal degrees. Comparison at scale 1:64000:

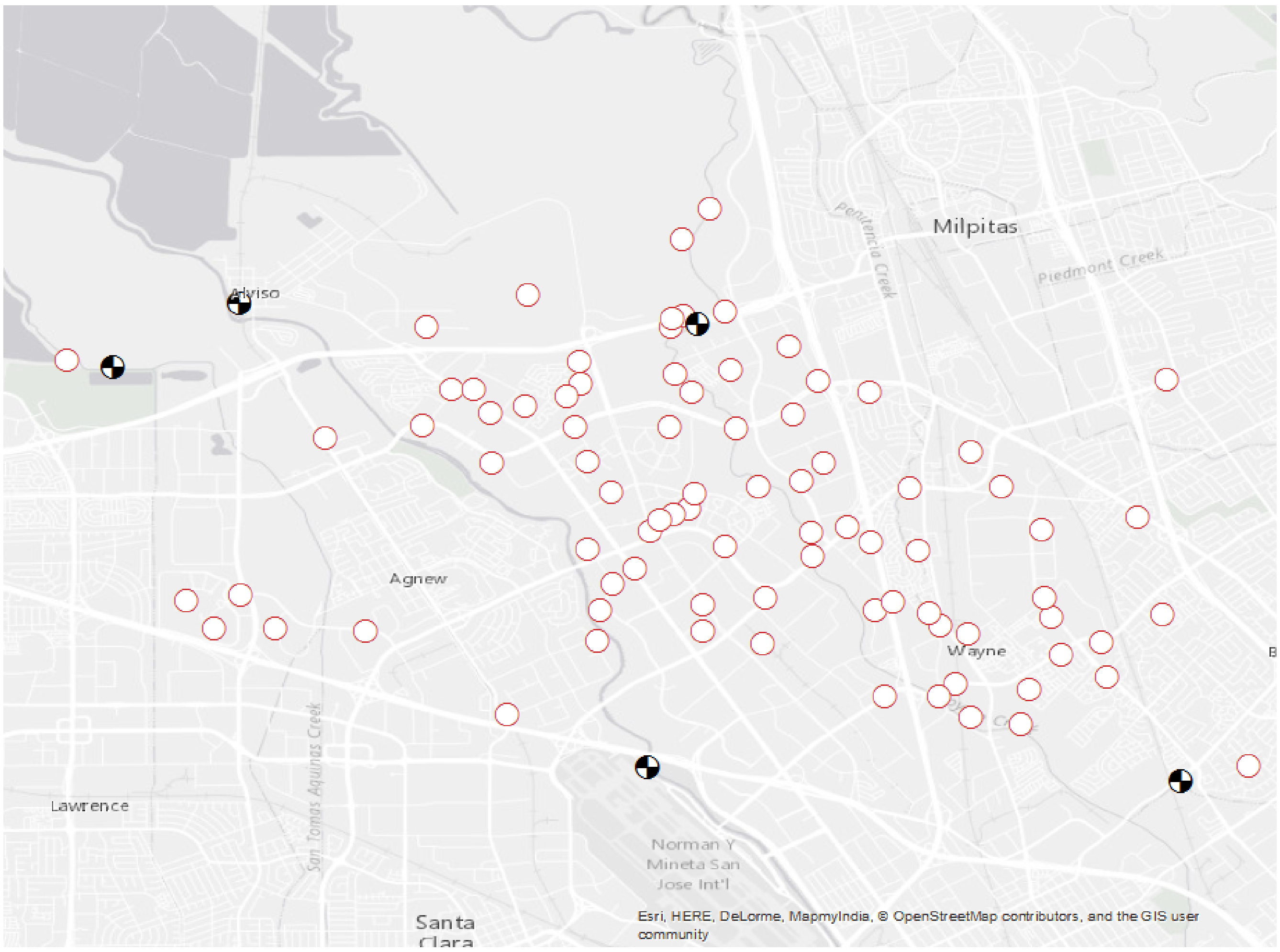

Milpitas
Penitencia Creek
Piedmont Creek
Alviso
Agnew
Wayne
Lawrence
San Tomas Aquinas Creek
Norman Y Mineta San Jose Int'l
Santa Clara
Esri, HERE, DeLorme, MapmyIndia, © OpenStreetMap contributors, and the GIS user community

Figure 15. All NHDPoints in the area of interest at scale 1:64000

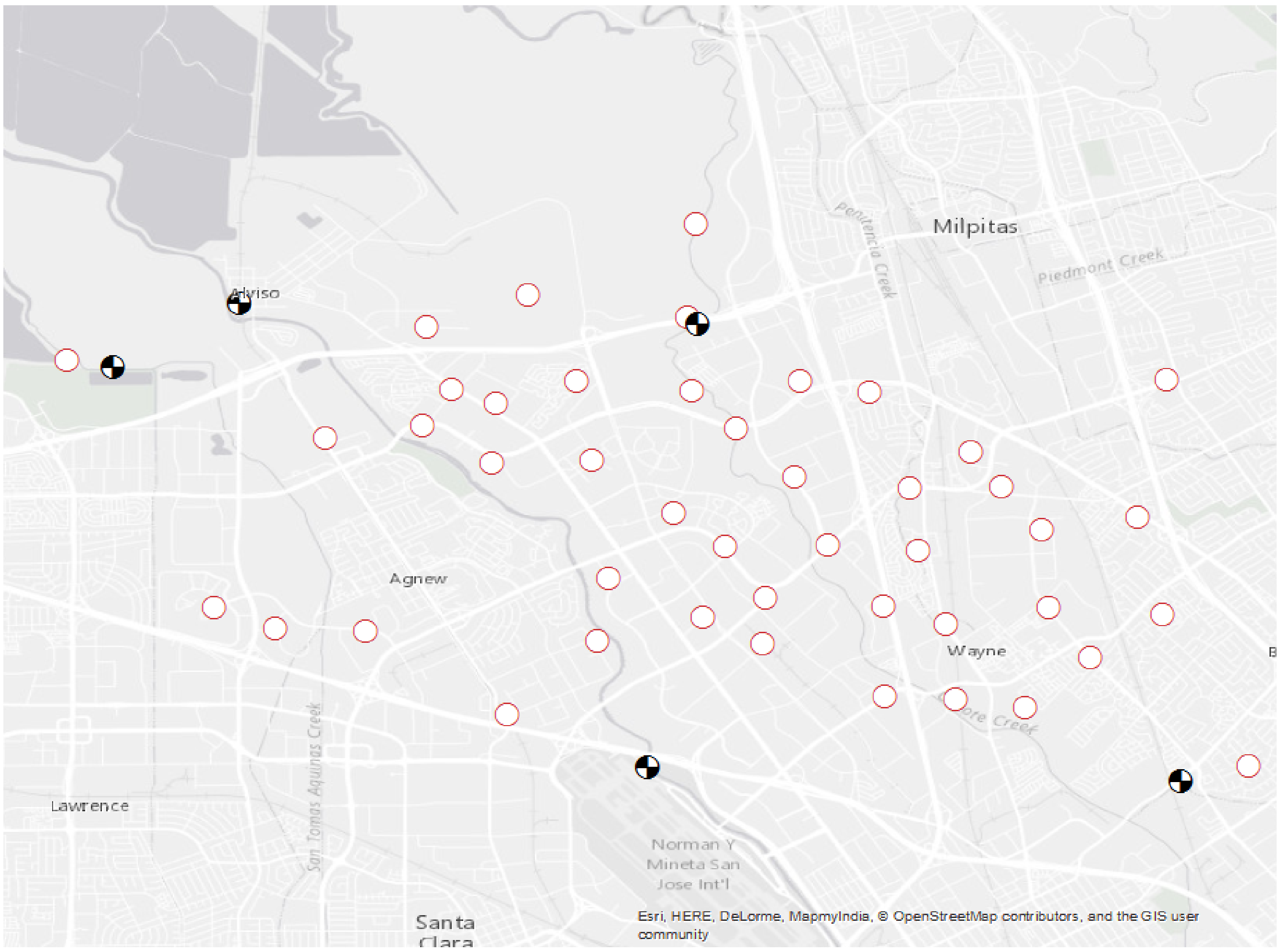

Milpitas
Piedmont Creek
Penitencia Creek
Alviso
Agnew
Wayne
Lawrence
San Tomas Aquinas Creek
Norman Y Mineta San Jose Int'l
Santa Clara
Esri, HERE, DeLorme, MapmyIndia, © OpenStreetMap contributors, and the GIS user community

Figure 16. Generalized NHDPoints in the aoi at scale 1:64000

Comparison at scale 1:150000:

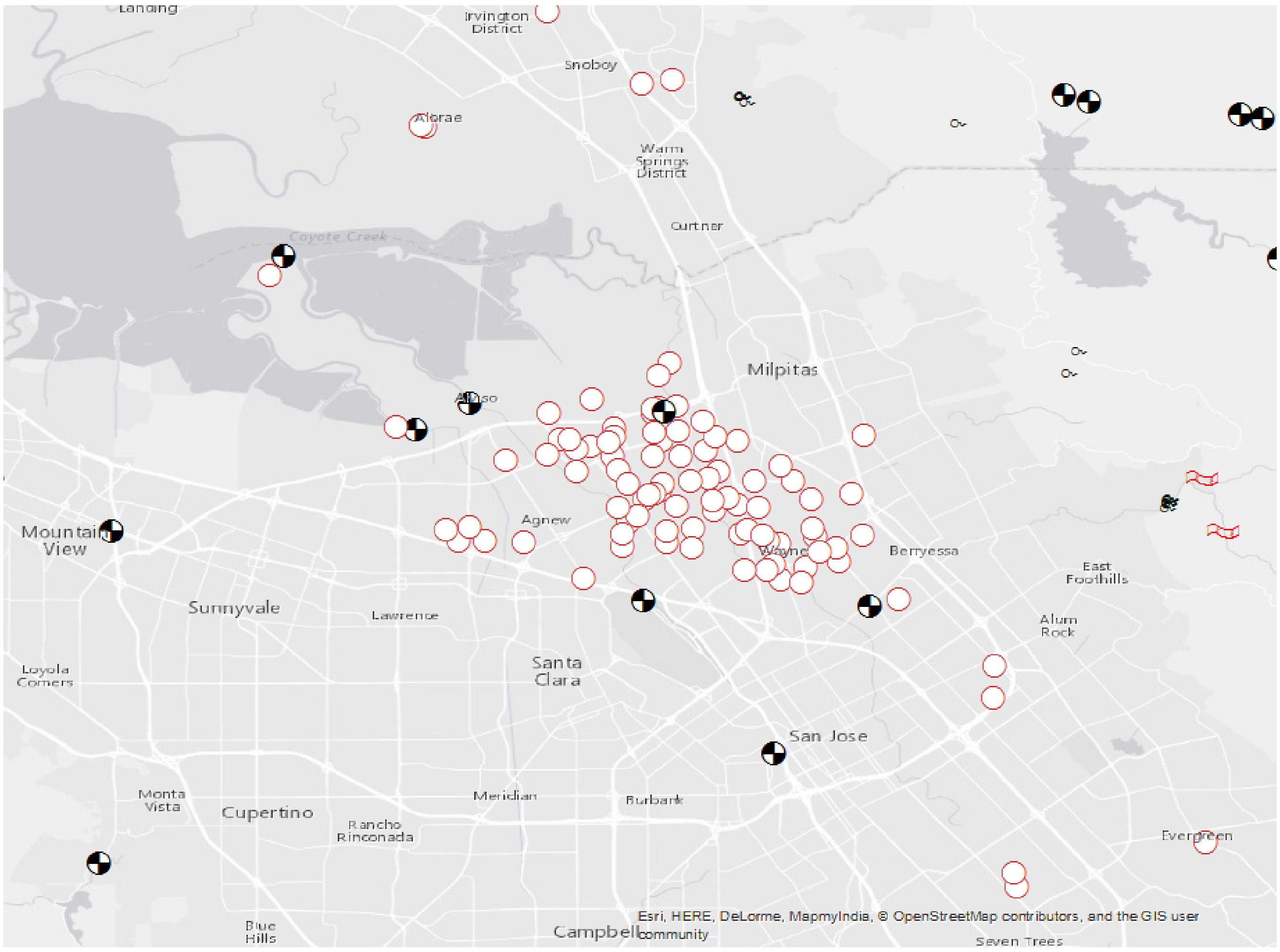


Figure 17. All NHDPoints in the aoi at scale 1:150000

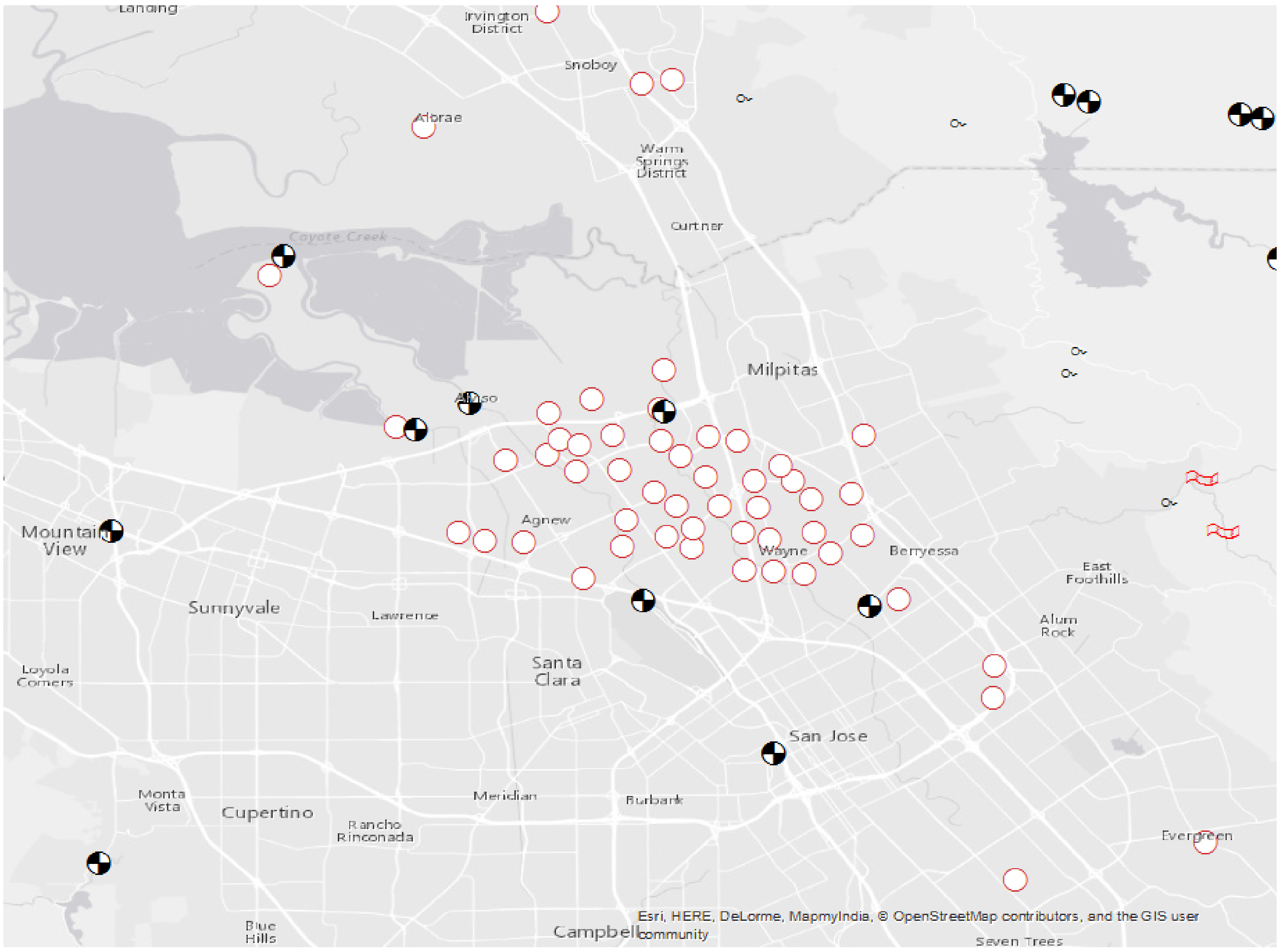


Figure 18. Generalized NHDPoints in the aoi at scale 1:150000

Table 4. Point generalization dataset statistics

Dataset	**Sum of all features**
Original	1819
Generalized, distance 0.005 decimal degrees	787
Generalized, distance 0.05 decimal degrees	176

The feature-count can be reduced greatly, up to 90 %. The following table shows a comparison of the sizes of the NHD_Point dataset before and after the generalization.

Table 5. Point generalization dataset size

Dataset	**Size zipped**	**Size unzipped**
Original	64.6 KB	454 KB
Generalized, distance 0.005 decimal degrees	38 KB	345 KB
Generalized, distance 0.05 decimal degrees	11.1 KB	78.4 KB

The size of the dataset could be significantly reduced. Using a distance of 0.005 decimal degrees, the size was reduced by approximately 40 % (zipped)/ 25 % (unzipped). A distance of 0.05 decimal degrees lead to a reduction of approximately 83 % for both zipped and unzipped dataset.

8.2. Line Generalization

8.2.1. Concept

For line generalization the Douglas Peucker algorithm was chosen. A boolean flag can be set to enforce topological preservation during the simplification process.

8.2.2. Implementation

Java-based DouglasPeuker line generalization algorithm *[5: source https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm]*

```
function DouglasPeucker(PointList[], epsilon)
    // Find the point with the maximum distance
    dmax = 0
    index = 0
    end = length(PointList)
    for i = 2 to ( end - 1) {
        d = perpendicularDistance(PointList[i], Line(PointList[1], PointList[end]))
        if ( d > dmax ) {
            index = i
            dmax = d
        }
    }
    // If max distance is greater than epsilon, recursively simplify
    if ( dmax > epsilon ) {
        // Recursive call
        recResults1[] = DouglasPeucker(PointList[1...index], epsilon)
        recResults2[] = DouglasPeucker(PointList[index...end], epsilon)

        // Build the result list
        ResultList[] = {recResults1[1...length(recResults1)-1],
recResults2[1...length(recResults2)]}
    } else {
        ResultList[] = {PointList[1], PointList[end]}
    }
    // Return the result
    return ResultList[]
end
```

The following listing shows the process description:

DouglasPeucker algorithm process description

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:ProcessOfferings xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xlin="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd">
  <wps:ProcessOffering processVersion="1.0.0" jobControlOptions="sync-execute async-
execute" outputTransmission="value reference">
    <wps:Process>
      <ows:Title>testbed12.fo.DouglasPeuckerAlgorithm</ows:Title>
      <ows:Identifier>testbed12.fo.DouglasPeuckerAlgorithm</ows:Identifier>
      <ows:Metadata xlin:role="http://www.opengis.net/spec/wps/2.0/def/process-
profile/concept" xlin:href="http://52north.github.io/wps-
profileregistry/concept/generalization.html"/>
      <ows:Metadata xlin:role="http://www.opengis.net/spec/wps/2.0/def/process-
profile/generic" xlin:href="http://52north.github.io/wps-profileregistry/generic/dp-
line-generalization.html"/>
      <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html"/>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>tolerance</ows:Title>
        <ows:Identifier>tolerance</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html#tolerance"/>
        <ns:LiteralData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="text/plain"/>
          <ns:Format mimeType="text/xml"/>
          <LiteralDataDomain>
            <ows:AnyValue/>
            <ows:DataType ows:reference="xs:double"/>
          </LiteralDataDomain>
        </ns:LiteralData>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>preserve_topology</ows:Title>
        <ows:Identifier>preserve_topology</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html#preserve_topology"/>
        <ns:LiteralData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="text/plain"/>
          <ns:Format mimeType="text/xml"/>
          <LiteralDataDomain>
            <ows:AnyValue/>
            <ows:DataType ows:reference="xs:boolean"/>
          </LiteralDataDomain>
        </ns:LiteralData>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
```

```
        <ows:Title>data</ows:Title>
        <ows:Identifier>data</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html#data"/>
        <ns:ComplexData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="application/vnd.google-earth.kml+xml"
schema="http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd"/>
          <ns:Format mimeType="application/x-zipped-shp"/>
          <ns:Format mimeType="application/x-zipped-shp" encoding="base64"/>
          <ns:Format mimeType="application/x-zipped-wkt" encoding="Base64"/>
          <ns:Format mimeType="application/vnd.geo+json"/>
          <ns:Format mimeType="text/json"/>
          <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
        </ns:ComplexData>
      </wps:Input>
      <wps:Output>
        <ows:Title>result</ows:Title>
        <ows:Identifier>result</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html#result"/>
        <ns:ComplexData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="application/vnd.google-earth.kml+xml"
schema="http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd"/>
          <ns:Format mimeType="application/x-zipped-shp"/>
          <ns:Format mimeType="application/x-zipped-shp" encoding="base64"/>
          <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
          <ns:Format mimeType="application/vnd.geo+json"/>
        </ns:ComplexData>
      </wps:Output>
    </wps:Process>
  </wps:ProcessOffering>
</wps:ProcessOfferings>
```

This process implements the DouglasPeucker process implementation profile described in section 9. The profile is extended by additional formats like GeoJSON.

8.2.3. Results

The algorithm was tested on the NHDFlowLine dataset, using a tolerance of 0.01 decimal degrees. The following images show a comparison of the input and output datasets:

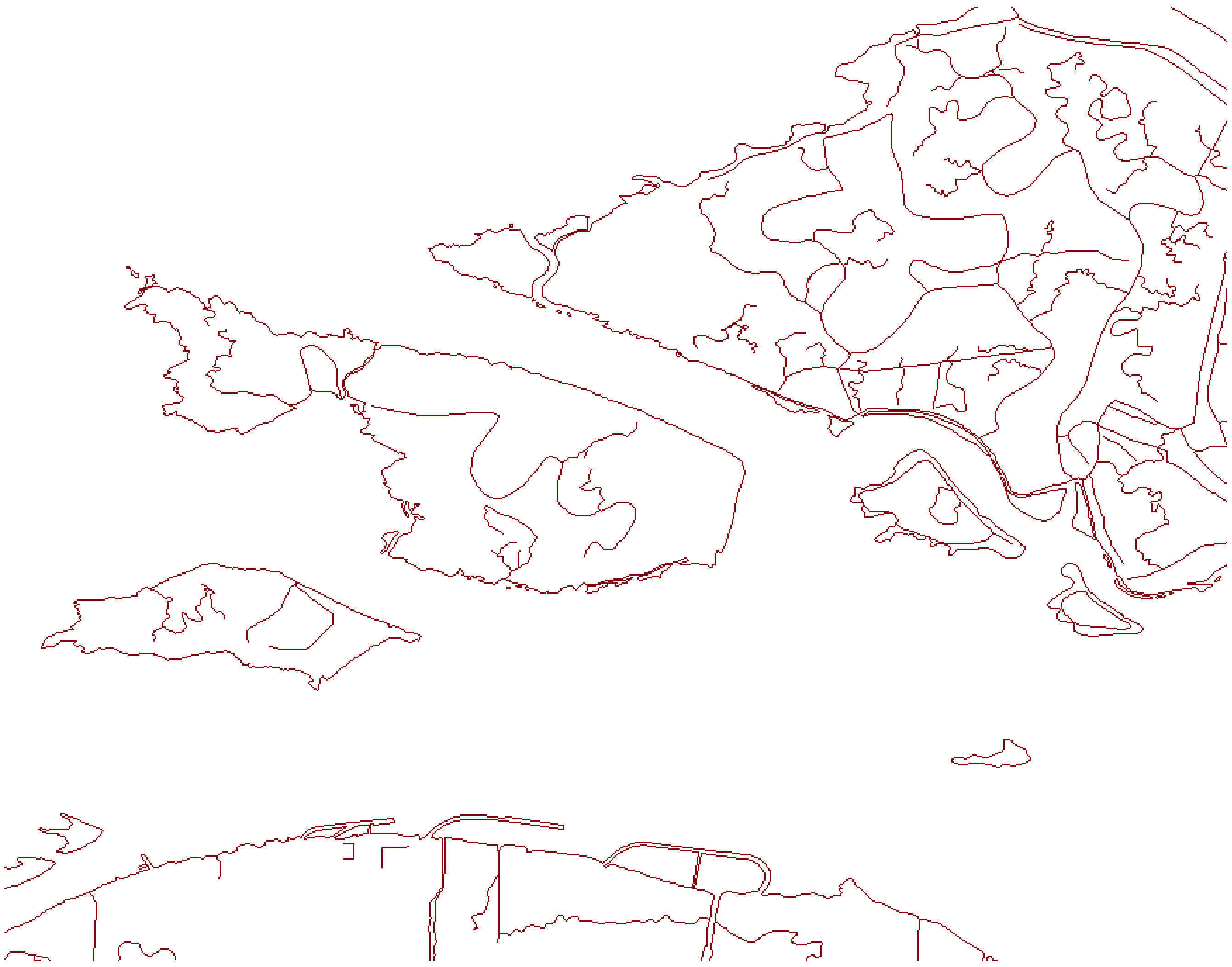


Figure 19. Original NHDFlowline in the aoi at scale 1:40000

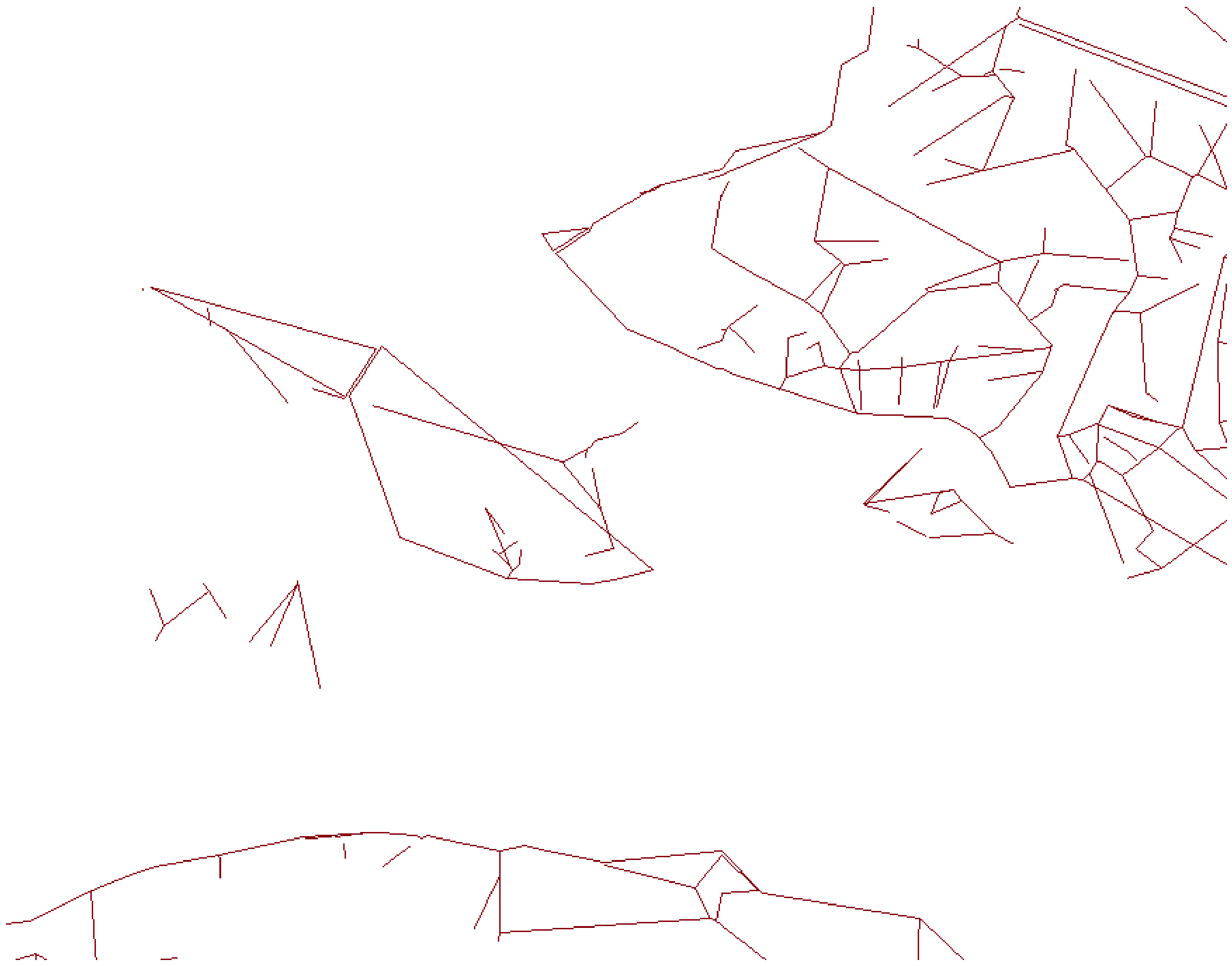


Figure 20. Generalized NHDFlowline in the aoi at scale 1:40000

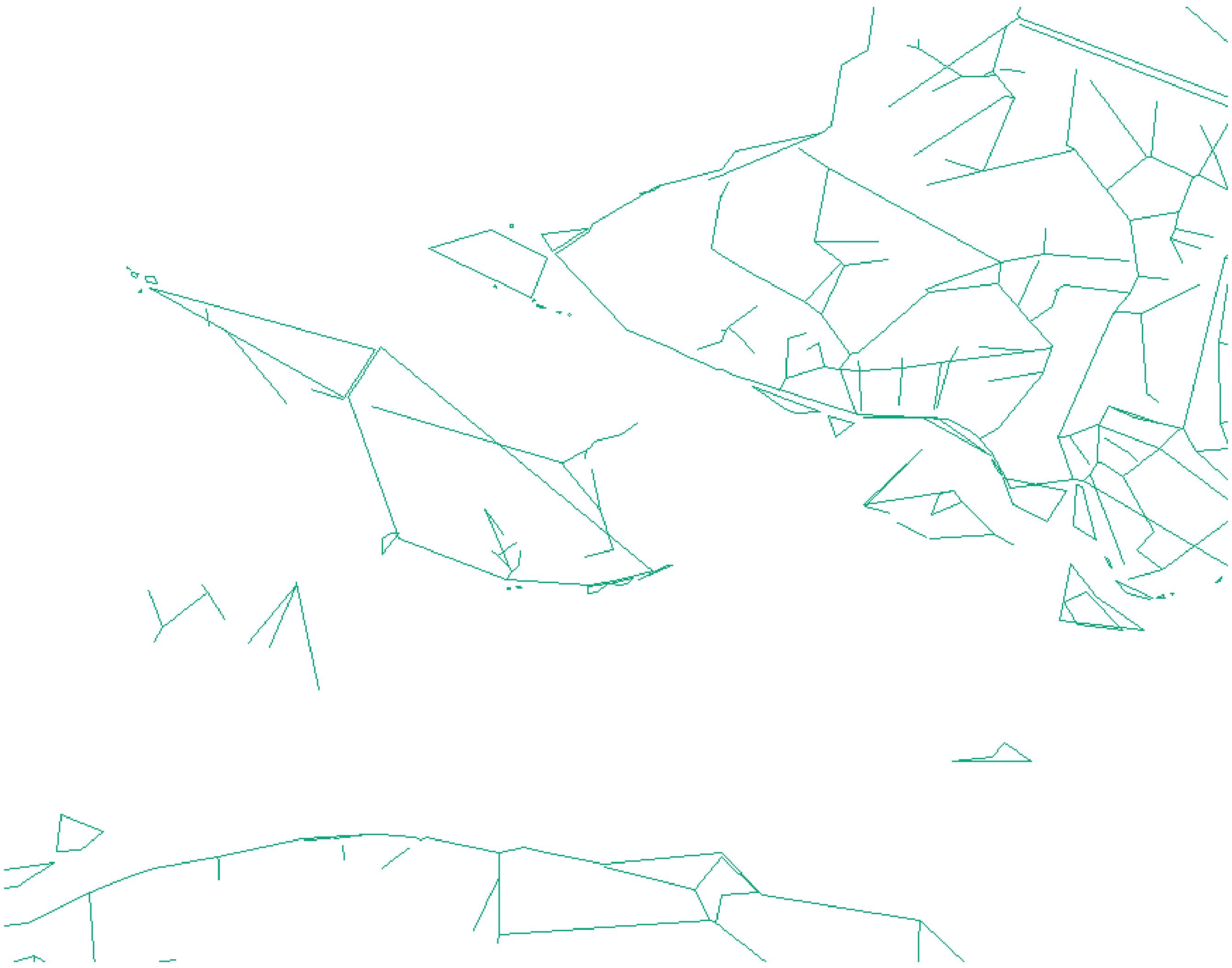


Figure 21. Generalized NHDFlowline with preserved topology in the aoi at scale 1:40000

The following table shows statistics for the original and the generalized datasets:

Table 6. Line generalization dataset statistics

Dataset	Minimum length of a feature	Maximum length of a feature	Sum of length of all features
Original	1.96 meters	32561.15 meters	16875660.43
Generalized, topology not preserved	0 meters	30499.58 meters	14432542.2 meters
Generalized, topology preserved	1.96 meters	30499.58 meter	14630853.69 meters

The overall length of the features was reduced by approximately 15 % either preserving the topology or not. The following table shows a comparison of the sizes of the NHD_Flowline dataset before and after the generalization. Again 0.01 decimal degrees were used for the tolerance. The number of features stayed the same for all three datasets.

Table 7. Line generalization dataset size

Dataset	Size zipped	Size unzipped
Original	12.6 MB	30.3 MB
Generalized, topology not preserved	1.77 MB	15.1 MB
Generalized, topology preserved	1.79 MB	15.2 MB

The size of the dataset could be significantly reduced. The size of the zipped dataset was reduced by nearly 85 %. After unzipping the reduction was still approximately 50 %. Whether the topology was preserved or not did not make a significant difference.

8.3. Area Generalization

8.3.1. Concept

As an example for area generalization we chose an algorithm, that removes polygon features based on a minimum area threshold. The following listing shows the code of the algorithm written in Java:

8.3.2. Implementation

The following listing shows the code of the algorithm written in Java:

Java-based AreaFeatureRemoval algorithm.

```
        for (FeatureIterator<?> iter = features.features(); iter.hasNext();) {
            SimpleFeature f = (SimpleFeature) iter.next();
            Geometry geom = (Geometry) f.getDefaultGeometry();
            Polygon polygon = null;
            if (geom instanceof Polygon) {
                polygon = (Polygon)geom;
            }else if(geom instanceof MultiPolygon) {
                polygon = (Polygon) ((MultiPolygon)geom).getGeometryN(0);
            }
            if (polygon != null && polygon.getArea() >= minSize) {
                featuresToRemain.add(f);
            }
        }
```

The process description of the process looks like the following:

AreaFeatureRemoval algorithm process description

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:ProcessOfferings xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xlin="http://www.w3.org/1999/xlink"
```

```
xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd">
  <wps:ProcessOffering processVersion="0.0.1" jobControlOptions="sync-execute async-
execute" outputTransmission="value reference">
    <wps:Process>
      <ows:Title>testbed12.fo.AreaFeatureRemoval</ows:Title>
      <ows:Identifier>testbed12.fo.AreaFeatureRemoval</ows:Identifier>
      <ows:Metadata xlin:role="http://www.opengis.net/spec/wps/2.0/def/process-
profile/concept" xlin:href="http://52north.github.io/wps-
profileregistry/concept/generalization.html"/>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>minSize</ows:Title>
        <ows:Identifier>minSize</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/merge-features-
by-attribute.html#distance"/>
        <ns:LiteralData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="text/plain"/>
          <ns:Format mimeType="text/xml"/>
          <LiteralDataDomain>
            <ows:AnyValue/>
            <ows:DataType ows:reference="xs:double"/>
          </LiteralDataDomain>
        </ns:LiteralData>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>data</ows:Title>
        <ows:Identifier>data</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/merge-features-
by-attribute.html#data"/>
        <ns:ComplexData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="application/vnd.google-earth.kml+xml"
schema="http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd"/>
          <ns:Format mimeType="application/x-zipped-shp"/>
          <ns:Format mimeType="application/x-zipped-shp" encoding="base64"/>
          <ns:Format mimeType="application/x-zipped-wkt" encoding="Base64"/>
          <ns:Format mimeType="application/vnd.geo+json"/>
          <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
        </ns:ComplexData>
      </wps:Input>
      <wps:Output>
        <ows:Title>result</ows:Title>
        <ows:Identifier>result</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/merge-features-
by-attribute.html#result"/>
```

```
          <ns:ComplexData xmlns:ns="http://www.opengis.net/wps/2.0">
            <ns:Format default="true" mimeType="application/vnd.google-earth.kml+xml"
schema="http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd"/>
            <ns:Format mimeType="application/x-zipped-shp"/>
            <ns:Format mimeType="application/x-zipped-shp" encoding="base64"/>
            <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
            <ns:Format mimeType="application/vnd.geo+json"/>
          </ns:ComplexData>
        </wps:Output>
      </wps:Process>
    </wps:ProcessOffering>
  </wps:ProcessOfferings>
```

8.3.3. Results

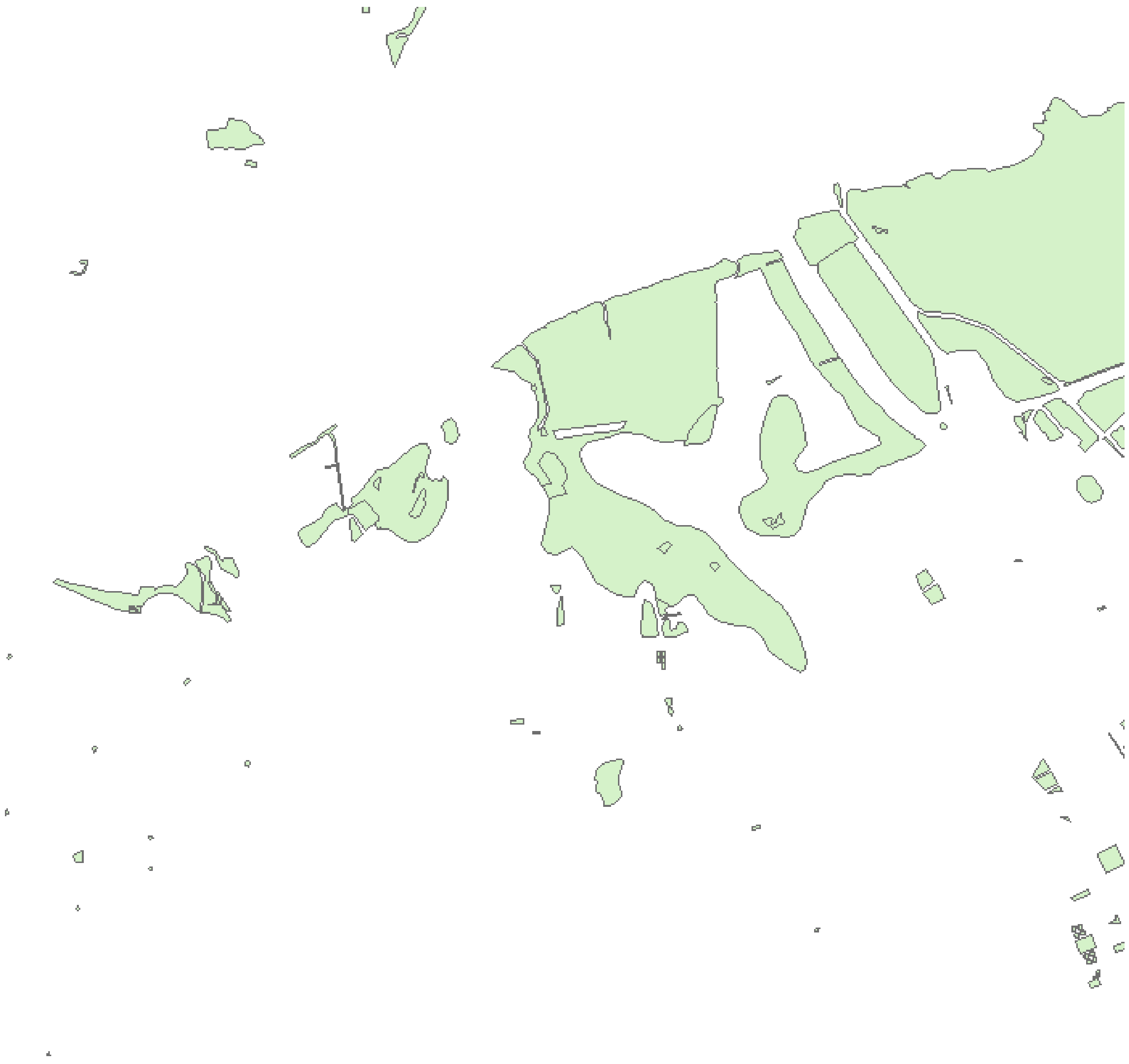


Figure 22. Original NHDWaterbody in the aoi at scale 1:43000

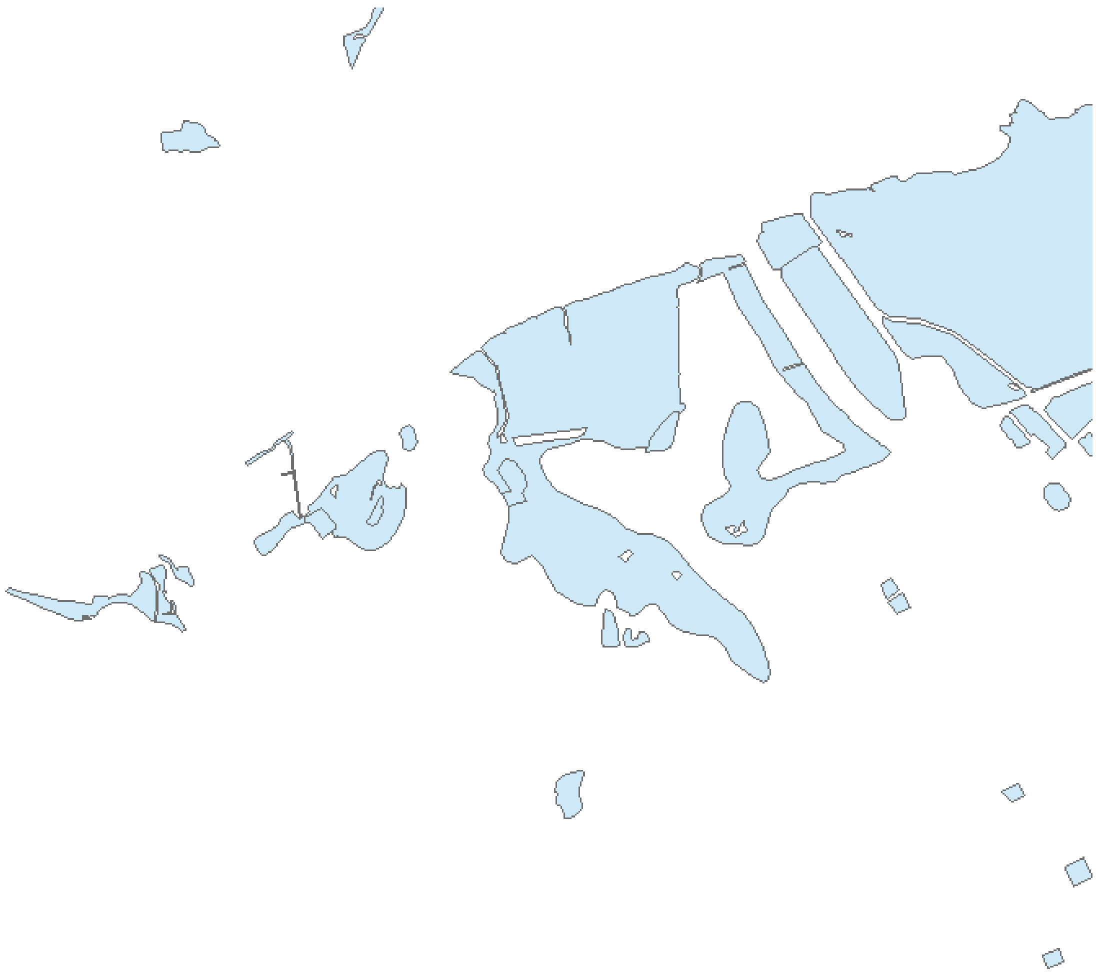


Figure 23. Generalized NHDWaterbody in the aoi at scale 1:43000

The following table shows statistics for generalized NHDWaterbody features:

Table 8. Area generalization dataset statistics

Dataset	**Minimum area of a feature**	**Sum of area of all features**	**Feature count**
Original	59.67 m^2	533170369.04 m^2	7536
Generalized, min. area 100 m^2	102.31 m^2	533170126.77 m^2	7533
Generalized, min. area 1000 m^2	1000.53 m^2	531998402.48 m^2	5437

The overall length of the features was reduced by approximately 1 %. The following table shows a comparison of the sizes of the NHD_Waterbody dataset before and after the generalization.

Table 9. Area generalization dataset size

Dataset	**Size zipped**	**Size unzipped**
Original	3.72 MB	12.2 MB

Dataset	Size zipped	Size unzipped
Generalized, min. area 100 m^2	3.66 MB	13.2 MB
Generalized, min. area 1000 m^2	3.25 MB	11.3 MB

After removing all features smaller than 100 m^2, the size of the unzipped dataset increases. This is due to the relatively small number of removed features and the increased field lengths of the resulting shapefile. When removing all features smaller the 1000 m^2, the size is reduced by approximately 7 %.

Chapter 9. WPS 2.0 Generalization Profile

The WPS 2.0 standard defines a hierarchical profiling approach (see OGC 14-065, section 7.5). On a high level, process concepts are defined. These concepts don’t follow a specific format and can be e.g. HTML pages describing the concept in human-readable form. The second level of the profile hierarchy are generic profiles. These profiles are described in an abstract way similar to a process description. The mechanics of the processes implementing this profile should be explained in detail. Also the inputs and outputs are described in a generic way, excluding specific data formats. The third level of profiling are implementation profiles. Technically, they are process descriptions, i.e. they describe inputs and outputs including data formats.

Based this approach, we defined different profiles for generalization:

- Generalization process concept
- Generic line generalization process
- DouglasPeucker process implementation profile

Profiles can be extended and themselves extend multiple profiles. In the following the profiles for generalization are described.

9.1. Generalization process concept

A process concept describes the functionality of a process or process group on a high level. The WPS 2.0 standard recommends that this could be done using a HTML page. The high level process concept for generalization can be found here: http://52north.github.io/wps-profileregistry/concept/generalization.html

9.2. Generic line generalization process

A generic process profile consists of (1) a XML document similar to a process description but without data formats specified for inputs and outputs and (2) a HTML page with descriptions for the process itself and for the inputs and outputs. The generic generalization process profile can be found here: http://52north.github.io/wps-profileregistry/generic/dp-line-generalization.html

The generic process XML description:

DouglasPeucker line generalization process description

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:GenericProcess
    xmlns:ows="http://www.opengis.net/ows/2.0"
    xmlns:wps="http://www.opengis.net/wps/2.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd">
      <ows:Title>DouglasPeucker line Generalization</ows:Title>
      <ows:Identifier>http://52north.github.io/wps-profileregistry/generic/dp-line-
generalization.xml</ows:Identifier>
      <ows:Metadata xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-
profile/concept" xlink:href="http://52north.github.io/wps-
profileregistry/concept/generalization.html"/>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>data</ows:Title>
        <ows:Identifier>data</ows:Identifier>
        <ows:Metadata
xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlink:href="http://52north.github.io/wps-profileregistry/generic/dp-line-
generalization.html#_data"/>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>tolerance</ows:Title>
        <ows:Identifier>tolerance</ows:Identifier>
        <ows:Metadata
xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlink:href="http://52north.github.io/wps-profileregistry/generic/dp-line-
generalization.html#_tolerance"/>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>preserve_topology</ows:Title>
        <ows:Identifier>preserve_topology</ows:Identifier>
        <ows:Metadata
xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlink:href="http://52north.github.io/wps-profileregistry/generic/dp-line-
generalization.html#_preserve_topology"/>
      </wps:Input>
      <wps:Output>
        <ows:Title>result</ows:Title>
        <ows:Identifier>result</ows:Identifier>
        <ows:Metadata
xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlink:href="http://52north.github.io/wps-profileregistry/generic/dp-line-
generalization.html#_result"/>
      </wps:Output>
</wps:GenericProcess>
```

The following table describes the inputs of the profile:

Table 10. DouglasPeucker algorihm inputs

Input Name	**Description**	**Value**
data	The input line features	Line features
tolerance	The tolerance for the DouglasPeucker algorithm	Double
preserve_topology	Indicates whether or not to preserve the topology of the features	Boolean

Table 11. DouglasPeucker algorihm outputs

Output Name	**Description**	**Value**
result	The simplified line features	Line features

9.3. DouglasPeucker line generalization process implementation profile

The lowest, most detailed hierarchy level are the implementation profiles. Their structure is that of a process description. The following listing shows the implementation profile for the DouglasPeucker process:

DouglasPeucker algorithm process description

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:ProcessOfferings xmlns:wps="http://www.opengis.net/wps/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xlin="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd">
  <wps:ProcessOffering processVersion="1.0.0" jobControlOptions="sync-execute async-
execute" outputTransmission="value reference">
    <wps:Process>
      <ows:Title>DouglasPeucker Algorithm Implementation Profile</ows:Title>
      <ows:Identifier>http://52north.github.io/wps-
profileregistry/implementing/douglas-peucker</ows:Identifier>
      <ows:Metadata xlin:role="http://www.opengis.net/spec/wps/2.0/def/process-
profile/concept" xlin:href="http://52north.github.io/wps-
profileregistry/concept/generalization.html"/>
      <ows:Metadata xlin:role="http://www.opengis.net/spec/wps/2.0/def/process-
profile/generic" xlin:href="http://52north.github.io/wps-profileregistry/generic/line-
generalization.html"/>
      <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html"/>
```

```
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>preserve_topology</ows:Title>
        <ows:Identifier>preserve_topology</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html#preserve_topology"/>
        <ns:LiteralData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="text/plain"/>
          <ns:Format mimeType="text/xml"/>
          <LiteralDataDomain>
            <ows:AnyValue/>
            <ows:DataType ows:reference="xs:boolean"/>
          </LiteralDataDomain>
        </ns:LiteralData>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>tolerance</ows:Title>
        <ows:Identifier>tolerance</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html#tolerance"/>
        <ns:LiteralData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="text/plain"/>
          <ns:Format mimeType="text/xml"/>
          <LiteralDataDomain>
            <ows:AnyValue/>
            <ows:DataType ows:reference="xs:double"/>
          </LiteralDataDomain>
        </ns:LiteralData>
      </wps:Input>
      <wps:Input minOccurs="1" maxOccurs="1">
        <ows:Title>data</ows:Title>
        <ows:Identifier>data</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html#data"/>
        <ns:ComplexData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="application/x-zipped-shp"/>
          <ns:Format mimeType="application/x-zipped-shp" encoding="base64"/>
          <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
        </ns:ComplexData>
      </wps:Input>
      <wps:Output>
        <ows:Title>result</ows:Title>
        <ows:Identifier>result</ows:Identifier>
        <ows:Metadata
xlin:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
```

```
xlin:href="http://52north.github.io/wps-profileregistry/implementing/douglas-
peucker.html#result"/>
        <ns:ComplexData xmlns:ns="http://www.opengis.net/wps/2.0">
          <ns:Format default="true" mimeType="application/x-zipped-shp"/>
          <ns:Format mimeType="application/x-zipped-shp" encoding="base64"/>
          <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
        </ns:ComplexData>
      </wps:Output>
    </wps:Process>
  </wps:ProcessOffering>
</wps:ProcessOfferings>
```

It inherits the input and output names of the generic profile and in addition specifies the formats. This can be seen as blueprint for DouglasPeucker line generalization processes among different vendors.

Chapter 10. Recommendations

10.1. Low Bandwidth WFS

Given the results of Testbed 12 it may be reasonable to consider advancing an 'Compression Profile for WFS'. This profile would describe Best Practices for a WFS using compression out formats such as GZIP and EXI. Specifically, it would discuss

- Requesting compressed output (Accept-Encoding header vs OutputFormat methods)
- Client use. Specifically, if the schema is used on the server during the encoding, it is required during the decoding process on the client.
- Impact of situations where the schema for each FeatureType on a WFS must be preprocessed on the server. Guidance the schemas declaring 'float' data types. For example, the Compression Profile for WFS may be restricted to GML 3.
- Other topics as identified.

However, if a profile is considered it must be remembered that there may be operational implications of requiring the schema during the decoding process on the client. This means that for each FeatureType on a WFS the GML schema needs to be present on the server and each client that wishes to use EXI. implication of This may limit ad hoc client-server connections unless clients read the schema for each feature type encoded in GML, which is possible, but adds complexity to client development.

10.2. Generalization WPS

The profiles developed in Testbed-12 belong to the first WPS 2.0 profiles ever created. They serve as proof-of-concept and blue-prints for future profiling efforts. However, the clients used in Testbed-12 did not support profiling, i.e. they could not handle the metadata information regarding the profiles. This reduces the usefulness of the profiles that should foster interoperability between clients and servers developed by different vendors. For future testbed initiatives, clients should be able to understand profile metadata and possibly servers from different vendors, implementing the same profile, should be developed. Then the usefulness of the profiles can be tested accordingly. Two additional recommendations for future work:

- Generalization ontology: The ICA Commission on Generalisation and Multiple Representation developed a ontology, called *GeneProcessOnto* [6: http://generalisation.icaci.org/index.php/generalisation-ontologies]. This could be utilized to perform automated generalization tasks.
- Spatio-temporal aggregation algorithms: The extend generalization processes with a temporal aspect.

Appendix A: Revision History

Table 12. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
April 12, 2016	B. Pross	.1	all	initial version
April 15, 2016	B. Pross	.1	all	Add outline and relevance section
May 4, 2016	B. Pross	.1	all	Complete IER
May 12, 2016	B. Pross	.1	8	Added section about point generalization
June 27, 2016	B. Pross	.1	8	Added section about line generalization
June 30, 2016	B. Pross	.1	all	Preliminary draft ER
July 26, 2016	B. Pross	.1	8	Added section about area generalization
August 5, 2016	B. Pross	.1	8	Updated UML diagram
August 8, 2016	B. Pross	.1	all	Merged remote changes
August 17, 2016	B. Pross	.1	8	Added section about topology preservation
September 27, 2016	B. Pross	.1	8	Updated WPS section
September 29, 2016	B. Pross	.1	all	Updated content
September 30, 2016	B. Pross, J. Harrison	.1	all	Updated various sections
October 10, 2016	B. Pross	.1	6	Updated background section, editorial changes

Date	Editor	Release	Primary clauses modified	Descriptions
October 11, 2016	B. Pross	.1	6	Updated background section, editorial changes
October 17, 2016	B. Pross	.1	8	Addressed comments from review

Appendix B: Bibliography

[1] Shea, S. K., and McMaster, R. B.: Cartographic generalization in a digital environment: When and how to generalize, Proceedings of AutoCarto, Vol. 9. (1989)

[2] McMaster, R. B.: Automated Line Generalization, Cartographica, 24(2), pp. 74-lll (1987)

[3] Keates, J.S.: Cartographic Design and Production (1973)

[4] Swiss Society of Cartography: Cartographic Generalization, Cartographic Publication Series, No. 2. (1977) English translation by Allan Brown and Arie Kers, ITC Cartography Department, Enschede, Netherlands)