

# **To use or not to use Open Source software: A comparison of strategies for development of robust geoprocessing services.**

Lars Aksel Opsahl and Knut Bjørkelo

The Norwegian Forest and Landscape Institute  
(Skog+Landskap)  
Lars.Aksel.Opsahl@skogoglandskap.no  
Knut.Bjorkelo@skogoglandskap.no

The Norwegian Forest and Landscape Institute (Skog+Landskap) have made experiences using open source java packages, like Geotools, for some applications. While proprietary software has evolved "unique" concepts which can be traced in performance and functionality, the open source is based on accepted, and fairly simple, concepts that are well known from standards and text-books.

Ideally we will combine open source and commercial products, tied together with standards and simple programming. We would use the software modules as a black box; put data and instructions into it and wait for a successful response.

In our context the commercial software is a black box, we can not change that. If we have a problem, the bug may be in the software itself, the database, input file or the system environment. We have to post a bug report, with detailed description of all aspects. The main problem is that the software vendor does not have the same database and environment as we have.

Using Geotools gives us a much better possibility to identify problems, because we have an environment where we can trace the source code line by line. If you have theoretical background, necessary software skills and a test environment, you may be able to fix the problem.

This presentation describes some aspects of our development process, based on recent experiences using open source software for heavy geoprocessing.

## **1 Introduction**

The primary goal of Skog+Landskap is to make resource information available to the public, typically as services on the internet. Internally we also need to handle large geographical datasets for analysis etc. We have started to use open source for a variety of applications. Our organization has broad GIS experience, which makes it possible to evaluate the results by comparison with other methods. Skog+Landskap is a public service institute, and the Governmental policy encourage use of open source software.

We did have problems with an Internet application running on Windows offering data, extracts from our databases, for municipalities or map tiles. The Java code was quite old and not very well structured. We cleaned up the Java code and made separate modules using Java interfaces for the Python and ArcGIS parts of the code. To test the code we made JUNIT tests, and the system failed as before. We were no closer to find

the reason for the problems, for some geographical structures the system just stopped inside the code, for others it worked OK.

It is not our goal to develop neither basic nor complete geographical software packages, but to use modules that handle geographical data efficiently. Ideally we will combine open source (like Geotools [1]) and commercial products (like ArcGIS), tied together with standards and simple programming. As long as the modules fill our requirements for functionality and efficiency, there is no need for us to go under the hood of the software we use. We could use the software as a black box because we don't really care what happens inside the box as long it behaves as expected; put data and instructions into the box and wait for a successful response.

## 2 Our current services on the Internet

Skog+Landskap's geodatabases are running on SunOS and we use Oracle 10 and ArcGIS 9.2. We have a master and slave database. The slave database is updated every week and is used by our Internet services. The databases are used by both Linux and Windows application servers running with ESRI products, Geotools and Mapserver. Our web server [www.skogoglandskap.no](http://www.skogoglandskap.no) is running on Linux and we are using ZTM for content management on the web server. Our applications also depend on external systems to provide property borders, background maps, orthophoto, etc for selected area.

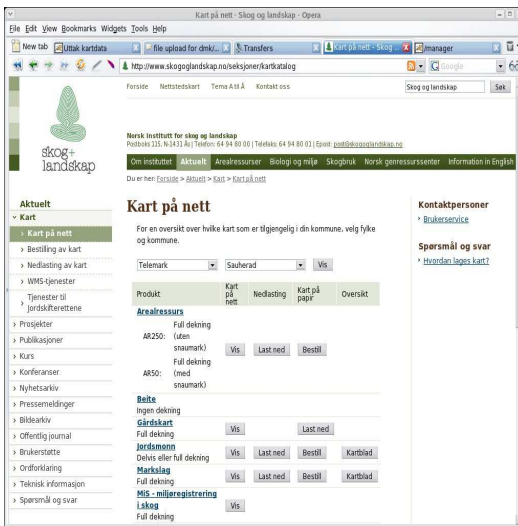
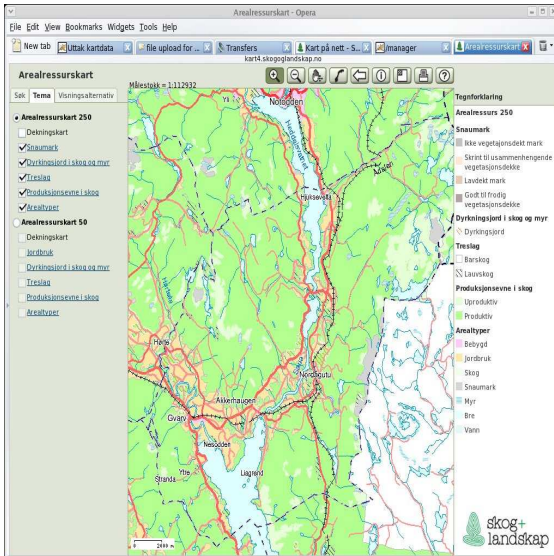


Fig. 1. Different types of maps on Internet available for the municipality Sauherad in Telemark.

We have applications for map download (figure 3) and map views (figure 2). The main portal to these services is a catalog web page, listing available maps for each

municipality (figure 1). The services are mainly intended for farmers and public services, but most are open to the public. A password protected service “farm maps” (figure 4) provides important documentation when farmers apply for governmental support.



**Fig. 2.** “Arealressurskart 250” for the municipality Sauherad in Telemark.

## 2.1 Typical tasks

In general the purpose of our map applications is to extract spatial data from different sources and put them together for the customer. The customer uses different kinds of criteria for data selection. We offer county borders, map tiles, attribute conditions or shape file [6] upload for selection of the area. Shape files can be dissolved and/or buffered before use. The result may be presented in a web browser, or returned as shape or SOSI-format files. The customer selects what coordinate system to use.

For external customers the requirements are mostly handled by our web based applications. For internal usage we have more special requirements which we handle with ad-hoc scripts or development versions of web based tools.

The majority of tasks can be solved using the data types and methods specified in the OGC specification for Simple feature access (SFA) [3], which has also become an ISO standard, and is implemented in a large number of software packages.



used both Python scripts and ArcGIS commands, and we tied it all together using Java.

One problem with the bugs was lack of error messages: the system just stopped and we had to restart the Java application to get it up running again. For the ESRI products we don't have any source code, thus it is difficult to find the exact cause of problems and file a precise bug report.

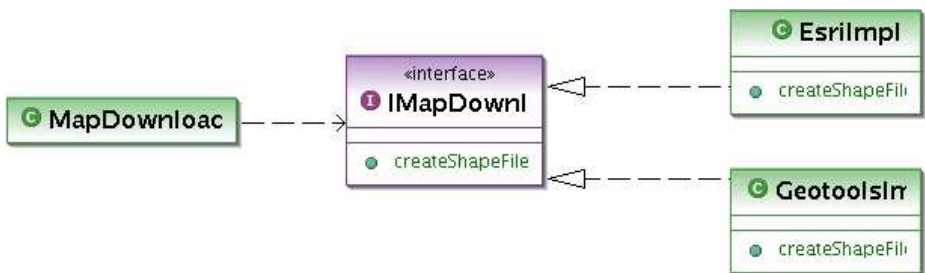
In our context the ArcGIS software is a black box, we can not change that. If we have a problem, the bug may be in the software itself, the database, input file or the system environment. We have to post a bug report, with detailed description of all aspects. The main problem is that the software vendor does not have the same database and environment as we have. The bug may not be reproduced in the vendor lab, thus the vendor is unable to verify if the bug is fixed. Further, the time elapsed before a new release is available to us may be weeks, months, or more.

We could specify our requirements and invite vendors to bid. We found that approach impractical, mainly because our need is a wide range of general GIS functionality. Any vendor would claim to satisfy the spec, and we might have to install and benchmark in excess, possibly under various versions of OS, web server, database, etc.

### 2.3 A possible new solution

From one application (developed by an external consultant) we had good experience with Geotools, and decided to try it for our “un-specified” geoprocessing demands. Our cost to test Geotools was very low since this is open source, and we had some knowledge of the software from before.

After the code clean up the map download application was structured using interfaces (figure 5) for the geoprocessing part, and it was easy to replace this part with Geotools components instead of ESRI components.



**Fig 5.** UML-diagram showing two classes that implement the interface **IMapDownload**.

We made the new class and rerun all JUNIT tests with success! Then we moved the code to Linux. The SOSI-format file creation program could only run on Windows, so we left it on a Windows server and used http for file transfer between the modules.

Using Geotools gives us a much better possibility to identify and solve problems, by source code debugging using production spatial data. To do this detailed source code debugging we must have the theoretical background, necessary software skills and a full test environment.

When working with application development on large geographical data sets, that are only available in a production environment, we believe the possibility to a full debug inside the house is very important for finding and fixing bugs. For instance, if we have a problem which is caused by polygon number 200, but don't appear before processing polygon number 891666, then the importance of production data is pretty clear.

Another aspect with open source is the community. For most of the problems we have been working with, we have usually found discussions around it on the Internet. If not a direct solution is offered, there are comments that guide us towards a solution.

Adherence to the fundamental concepts of geometry and topology, as expressed in relevant standards, is the glue for community and applications. All participants, not only the programmers, should know these standard concepts. Unexpected results may originate from data, methods or people not conforming to the standard. However, the outcome from debugging such cases is valuable improvements to data sources, software modules, and persistent knowledge for the people involved.

One thing about Geotools we like, is the purity of it. For instance, there is no check on ring ordering, polygon validity or duplicate point removal. However, there are plugins for this kind of functionality, that can be added when need arise. For example, when it seemed like ESRI had changed the default polygon ring ordering, then we had to include a plugin to check and correct ring ordering.

Using open source may also be a good opportunity to learn smart programming techniques from remote contributors, but of course the quality of the code may vary a bit.

### **3 Applications developed using Geotools**

To show how we use Geotools for development we will present three in-house cases.

#### **3.1 System setup**

The application server is a Dell with two Intel(R) Xeon(TM) CPU 3.60GHz dual core CPU, and 8 GB memory and Scsi disk. The operating system is CentOS 4.4 and use Tomcat web server and Java 1.5.

We have two database servers: Frigg which is a Sun Fire 280R (UltraSPARC-111+) with 3 GB memory, and Balder which is a Sun Fire V240 with 2 GB memory. The application server and Frigg is connected by a gigabit network, while application server and Balder is connected through firewall which means max 100 Mbit network connection.

The dataset used is DMK, a detailed polygon coverage describing land cover for the productive parts of Norway. It is stored as an ArcSDE layer in decimal degrees (Euref89), and contains approximately 9 million polygons with a total of 1 billion points. The minimum mapping unit is 0.05 hectares.

To this was added a development team of one experienced Java programmer and one experienced GIS user with detailed knowledge of relevant data and standards. Additional expertise on Oracle, web and ESRI software was easily available within the department.

### 3.2 Case 1: Selection by attribute

This test is a simple extract of all agricultural land which covers about 5 % of mapped area, based on attributes (SQL where-clause: ATIL in (21,22,23)). The result set will contain 867603 features, and is projected to UTM33, converted to a Shape file, which is zipped and stored on web server. User receives a mail with link to the result. The \*.dbf file size is 1200 MB, the \*.shp file size is 934 MB, and the zip size about 720 MB.

Using Balder as a database server the execution time is 2222 seconds, but when using Frigg the execution time is 3674 seconds. CPU usage (table 1) on the application server is low, around 5%. It seems to be waiting for the database server, but we have not focused much on finding bottlenecks yet.

**Table 1.** System load on application server, using Frigg as database server, sampled every 10 seconds.

```

[root@appl5 ~]# vmstat -n 300 10 -S M
procs -----memory----- ---swap-- ----io---- --system-- ----cpu----
 r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa
 0 0   1637  2147   1   14   0  0   62  75  133  16  2  0  98  1
 0 0   1637  2138   2   15   0  0   29   5 1183  72  5  1  94  0
 0 0   1630  2111   2   16   0  0   14   4 1224  71  7  1  93  0
 0 0   1630  2067   2   16   0  0    9   4 1203  70  5  0  94  0
 1 0   1630  1190   3   16   0  0    4   3 1187  68  6  1  93  0
 0 0   1630   589   2   16   0  0   44   4 1154  70  2  0  96  0
    
```

### 3.3 Case 2: Clipping to areas

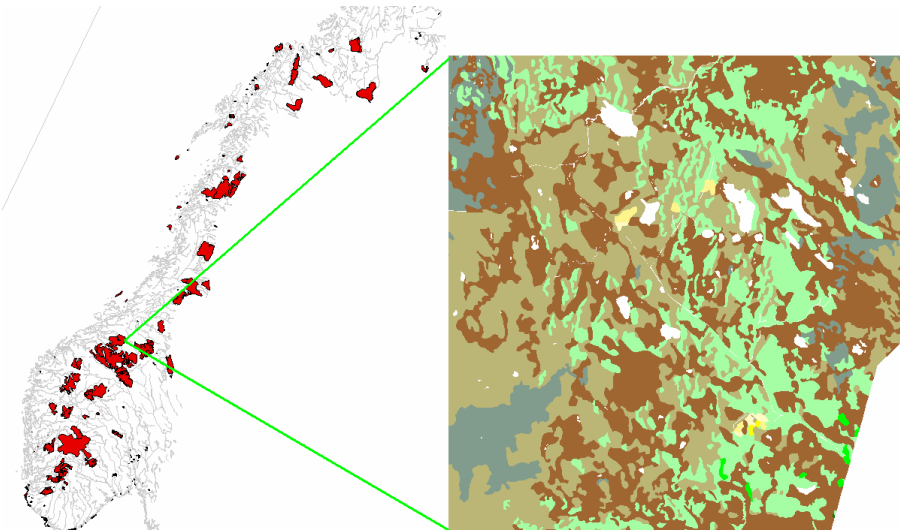
In another example we extract all DMK inside 520 protected areas (figure 6) which are uploaded as a shape file. The job takes 1665 seconds using Balder. The resulting \*.dbf file is 622 MB and the \*.shp file is 291 MB.

**Table 2.** System load on application server, using Frigg as database server, sampled every 10 seconds.

```

vmstat -n 300 10 -S M
procs -----memory----- ---swap-- ----io---- --system-- ----cpu----
 r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa
 0 0   20  2942   7   181  0  0   64  85  26  17  5  0  94  1
 5 0   20  2752   8   186  0  0    0  28 1558 106 47  1  52  0
15 0   20  2247   8   188  0  0    0  16 1522 106 74  1  25  0
 1 0   20  2157   8  193  0  0    0  27 1424 110 49  1  50  0
 0 0   20  1869  10  416  0  0    1 3912 1287  75 24  1  74  1
    
```

This case requires topological testing and intersection, which gives higher CPU load (around 50%) on the application server (table 2).



**Fig 6.** The left hand map shows all protected areas in the study. The right hand detail map shows all DMK polygons for a fraction of one area.

### 3.4 Case 3: Buffering

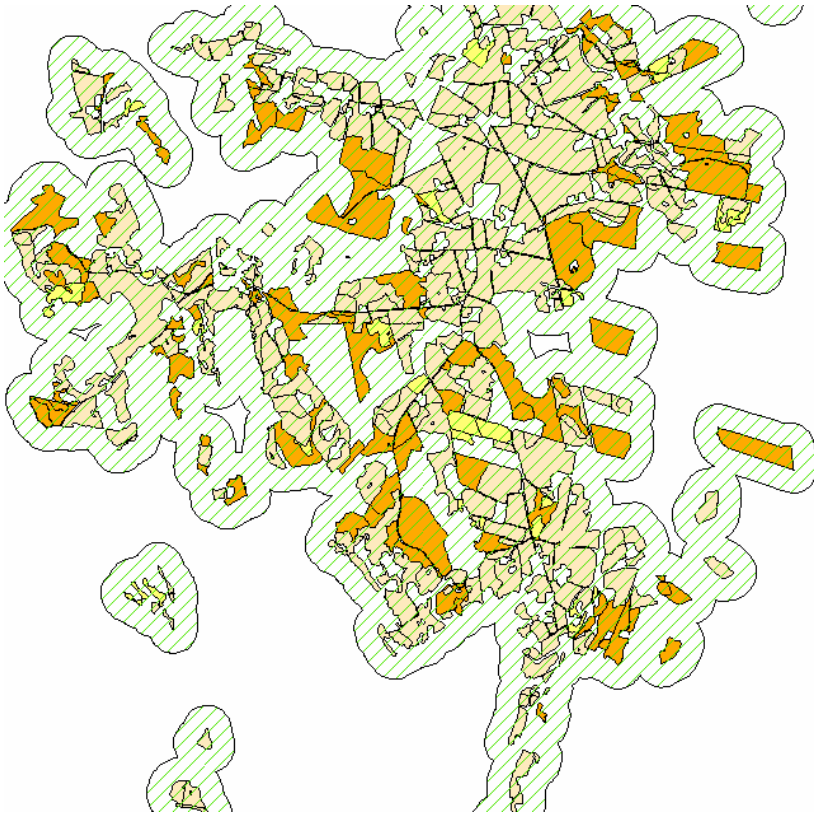
Geotools does not have a direct method to dissolve polygons. In our case we needed to buffer and dissolve a set of polygons (starting with the result data set from case 1). To implement dissolve we combined basic methods like intersection, difference and union that are implemented in the JTS Topology suite [2]. We had to resolve all geometry collections because some of the methods were only implemented for single polygon instances.

These operations encountered problems with invalid polygons and topology exceptions. We exported the objects, at various stages in the process, to shape files and inspected with ESRI tools. ArcView3 revealed some “inverted polygons”. The ArcGIS REPAIR function reported a lot of “self intersections”, but these seemed to be generated within the REPAIR function itself. Further, we used the debugger to trace some exceptions deep down into the Geotools and JTS source code and found double points, most likely a result of round off in some method used. To ensure correct ring ordering we used the `JTSUtilities.makeGoodShapePolygon`. We also used the `JTS DouglasPeuckerSimplifier` to clean up polygons, with the microscopic value 0.000000000001 decimal degrees as distance tolerance. This removed duplicate points and ensured valid polygons. After doing this all exceptions were gone and the application produced the desired result.

The open source made it possible for us to export data from any stage in the process, and make them available for thorough inspection. Without this option the bugs could still have been inside a black box.

The latest version of JTS (1.8.0) has improved robustness. However, objects are generated in other packages that have been developed and tested with earlier versions, and might not satisfy these requirements.

In this case, 867000 polygons are buffered with 100 meter and then dissolved. The input and output data is shown in figure 7 for a 4 km by 4 km area, which represents 0.01 % of the study area. The result is a shape file of 205 MB and the dbf file at 628 K, since we do not store any attributes. This operation reduces the set of polygons to about 37793, without any exceptions or errors that we have found so far. It used about 26 hour on this operation. Because the application server has 2 dual core CPU's and we have not added any concurrency for this operation, we are only able use 25 % of the CPU.



**Fig 7.** Selected polygons with agricultural land (yellow solids) and the dissolved buffer polygons (green hatching).

## 4 Conclusions

Skog+Landskap's general plan is to implement robust applications that contribute to the spatial data infrastructure. Independent of tools chosen, these applications develop from idea and internal ad-hoc solutions, via prototype web applications, to reliable

published services. The now simple and widely used WMS service [5] was in the workshed 7 years ago.

We believe that the combined efforts of standardization, software vendors, open source communities, academic research, and organizations like ours, will make it possible to implement a lot of powerful GIS operations. Finally we can make the GIS textbook examples work for real life applications.

So far Geotools is handling geo operations very well, but from our experience we use time to plug things together so it works as an application. For instance we had to include tools for concurrency, file download, mail, external communication, performance issues and more. The problem with this is that the business code is infected with all kinds of dependencies which does not have anything to do with geo processing. We believe that the Spring framework [4] is the right framework for us. In the Spring framework we have tools for most of cases above and we have support for dependency injection, which means that it is up to the container to set up a runtime environment for our application. In Spring you use XML to specify/describe the container that your modules should run in. Geotools it self is ready for Spring.

One explicit advantage of the open source approach is that licensing issues does not prohibit optimal configuration of the modules. When we have overcome the functional obstacles, we will focus on performance issues. That is a necessity if our new services get popular on the Internet.

The overall amount of labor for these programming and debugging efforts slightly exceeds similar projects based on black box modules. However, when using packages with closed source, we have to adapt functionality to their limitations. Being able to fully implement the services as desired by the customer is a great professional satisfaction. Even tracing some of the 1 billion points through the Math functions is more fun than adapting applications to black-box limitations and “undocumented features”, or waiting for responses from vendors support. Further, we now have knowledge and a framework for development of similar applications, based on the SFA standard.

It is encouraging that geoprocessing of large datasets is possible. In fact, some of the difficult cases had not been revealed if we used small test data sets. And they would not yet have been solved if we did not use open source!

## References

1. Geotools, The Open Source Java GIS Toolkit, URL <http://geotools.codehaus.org/>
2. JTS Topology suite, URL <http://www.vividsolutions.com/jts/jtshome.htm>
3. Open Geospatial Consortium, Simple Feature Access - Part 1: Common Architecture, URL <http://www.opengeospatial.org/standards/sfa>
4. Spring Framework, URL <http://www.springframework.org>
5. Open Geospatial Consortium, Web Map Service, URL <http://www.opengeospatial.org/standards/wms>
6. ESRI Shapefile Technical Description, ESRI White Paper, July 1998.