Computers & Geosciences 37 (2011) 65–74

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Geo-visualization Fortran library

Gen-Tao Chiang [a,*], Toby O.H. White [a], Martin T. Dove [a], C. Isabella Bovolo [b], John Ewen [b]

[a] Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ, UK
[b] School of Civil Engineering and Geosciences, Newcastle University, Newcastle upon Tyne NE1 7RU, UK

ARTICLE INFO

Article history:
Received 12 June 2009
Received in revised form
21 April 2010
Accepted 29 April 2010

Keywords:
Fortran
Google Earth
KML
Geobrowser

ABSTRACT

Geobrowser tools offer easy access to geographical and map images over which geospatial data can be overlaid, a process that provides a powerful new visualization resource for scientists. Many of these tools make use of the well-documented KML/XML data formats, and the challenge for the scientist is to generate KML files from their simulation and analysis programs. Since many of these programs are written in the Fortran language, which does not have native tools to support XML files, we have developed a new library – WKML – that enables KML files to be produced directly and automatically. This paper describes the WKML library, gives a number of different examples to illustrate the breadth of its functionality, and describes in more detail an example of its use for hydrology.

1. Introduction

1.1. Geobrowsers and XML

The recent development of geobrowser tools that provide free and easy access to high-quality maps and high-resolution aerial photographs of land have revolutionized the way in which researchers can visualize geospatial data. Not only are researchers able to use these tools in their own data analysis tasks, but they can create representations that allow them to present information to their collaborators or the wider public.

Many geobrowser tools (e.g. Google Earth, ArcGIS Explorer, NASA World Wind) will accept geospatial data represented using the Keyhole Markup Language (KML) (Wernecke, 2009), an XML-based language. KML is able to describe various primitive elements, such as points, lines and polygons, with specific geographical information. For example, a point can be represented using the following relatively simple piece of code:

Listing 1: Example of a simple KML file showing information about a single point.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
  <Placemark>
    <Point>
      <coordinates>-122.0822,37.4222
      </coordinates>
    </Point>
  </Placemark>
</kml>
```

Although this code looks straightforward, a file with many data points (and with more complex primitives) might normally be expected to be generated automatically as an output from a simulation or data analysis program. Many (or even most) numerical and simulation computer programs used in several branches of science – including climate, atmosphere, ocean, and hydrology models – are written using the Fortran language, and there are a number of reasons why it would be useful to have a library to support writing XML in general, and KML specifically, from Fortran programs. Briefly, some of these reasons are:

1. To make it easy to write XML/KML documents from a Fortran program, reducing the burden on the programmer in terms of the amount of coding and familiarity with XML and KML required. This is not to imply that programmers should not need to know anything at all about KML/XML, but essentially all they need to know is the broad general ideas. Moreover, there is no reason why a programmer should need to know about issues such as namespaces (as represented by the line `<kml xmlns="...">` in Listing 1).
2. To enable additional content, such as contour maps, which are not included in KML, to be automatically generated from programs.
3. For the generation of KML documents to be automatic from within a simulation program. The reader might question whether it would instead have been better to focus on writing a program that would parse a general output file into KML using

* Corresponding author. Tel.: +44 1223 764918; fax: +44 1223 333450.
E-mail address: gtc25@cam.ac.uk (G.-T. Chiang).

0098-3004/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2010.04.012

a more natural programming language. Whilst this is possible, it would put an additional burden on the code developer (not least because he/she would need to generate this tool, and parsing tools require some effort for them to be made robust), and would require the end user to run an additional program.

For these reasons we have created a library of subroutines to enable easy generation of KML files from within Fortran programs; the purpose of this paper is to provide a wide-ranging introduction to its use.

1.2. A taster of our XML Fortran library

We have previously developed the Fortran library "FoX" – Fortran XML (White et al., 2006a, 2006b, 2009) – to support writing XML from Fortran 95 programs.[1] For particular XML languages, we have developed specific libraries that provide more powerful language-specific subroutine calls. For example, much of our earlier work was concerned with the Chemical Markup Language (CML; White et al., 2006a, 2006b, 2009), and we developed a set of subroutines within the FoX library that automatically impose the appropriate syntax required by CML (WCML, "Writing CML"). This has been demonstrated to be extremely useful in the chemical simulation sciences to quickly add CML output capabilities to existing codes (Salje et al., 2009). The FoX library (Dove, 2009) is available as a download package, free of charge for anyone.

Based on our experience with FoX and WCML we have developed another FoX library called WKML ("Writing KML") to enable KML output to be produced by Fortran codes, and the purpose of this paper is to describe this library and to give a case study to illustrate its value. In anticipation of the description that follows later in this paper, and to set the scene, the small KML file shown above (Listing 1) would be written using the following WKML library calls within a Fortran program:

Listing 2: Fortran code to generate a KML file shown in Listing 1.

```
program write_kml_points

! Set up the XML file output channel
type(xmlf_t) :: xfile
real :: lat_long_positions(2,1)

! create the file
  call kmlBeginFile(xfile," output.kml" )
! Set up data
  lat_long_positions(1,1) = -122.0822
  lat_long_positions(2,1) = 37.4222
! Do the hard work with one subroutine call
  call kmlAddPoints(xfile, lat_long_positions)
! close the KML file neatly
  call kmlFinishFile(xfile)

end program write_kml_points
```

What should be clear from this example is:

1. No part of the program needed an awareness of the syntax of KML.
2. The writing of the latitude and longitude points required one simple subroutine call (`kmlAddPoints`), which clearly makes for condensed code when this process is repeated often in a program.
3. The subroutine calls `kmlBeginFile` and `kmlFinishFile` handle the correct beginning and ending of the KML file, again without requiring the programmer to know what XML/KML expects.

1.3. XML and eScience

Grid computing, as a general framework, is being adopted more and more by the Earth and Environmental Sciences communities worldwide (Renard and Badoux, 2009). Examples include the use of eScience approaches in Earth System modeling (Lenton et al., 2009), oceanographic modeling (Holt et al., 2009), geology (Gahegana et al., 2009), mineral physics (Salje et al., 2009), climate modeling (Frame et al., 2009), and simulations of natural hazards (Bovolo et al., 2009). Grid computing for hydrological applications, however, is still a relatively new field.

Grid computing can be a valuable resource for hydrology, particularly since it is well-suited for computationally intensive applications such as large-scale parameter sweeps, sensitivity studies and uncertainty analysis, and for multiple scenario applications such as looking at the impacts of land-use change or future climate change on the hydrology of a catchment. Such virtual experiments, by their very nature, generate large number of simulations and hence vast amounts of data, which in turn require efficient data handling and metadata management tools, together with ways to visualize and capture relevant outputs. Simple visualization tools such as Google Earth, are particularly useful, as they allow geospatial, graphical and animated outputs to be viewed in a freely available, easy to use, dynamic, interactive environment on the user's desktop.

Our work on WKML has arisen from realising that the use of XML for data representation can play an important role in aiding data interoperability (White et al., 2006a, 2006b). The *e* Minerals project (Salje et al., 2009) developed the FoX toolkit for the use of XML in grid computing, enabling automatic collaborative exchange of information, metadata collection through using Xpath expressions coupled with the *e* Minerals RCommands system (Tyer et al., 2006, 2007) and data analysis. In the development of the new WKML library for writing KML, we envisage that this will facilitate, in particular, the ability of collaborating researchers to share results with each other and with a wider public. This may include communities involved in setting policies and those who are subsequently affected.

1.4. Outline of this paper

In the next section of this paper we will build upon this introduction to give a more detailed description of some of the key features of WKML together with some simple examples. Following this we will describe a case study from the field of hydrology, but the reader should appreciate that nothing within WKML prohibits application in any other field of science that uses geospatial data or information. In the use case described here we actually focus on the visualizations of outputs using WKML and its benefits, and the associated eScience infrastructure, rather than on the scientific aims and results.

2. The new WKML Fortran library

2.1. General principles

The introduction described the main design requirement behind WKML, namely is that it should make creation of KML files as easy and intuitive as possible from within Fortran programs. The target audience is the Fortran programmer who wants to be able to generate KML files but who is not necessarily an expert in KML. Thus our main aim has been to create a set of

[1] Fortran 77 has persisted into the 21st Century for many applications, but incorporation of Fortran 77 syntax within a Fortran 95 program is straightforward for well-written codes.

subroutines that can be called in a style that likely best matches the thought processes of the scientist programmer, with usability being a key requirement. As a result, we have attempted to identify the sort of task that the programmer will want to tackle based on analyzing the actual work requirements of a small group of users and programmers, and the subroutine calls have been designed accordingly. We have not set out to provide a complete coverage of KML functionality, although missing features can often easily be added.[2] Some of the general principles have been described in a preliminary conference paper (Chiang et al., 2007) and a brief report (Chiang et al., 2009).

In this section we describe the main subroutines that are each accessed by programmers as a standard Application Programming Interface (API). These are summarized in a handy form in Table 1. WKML also provides a set of lower-level functions which are designed for developers to build a WKML API rather than for application programmers, and we will only mention some of these in passing here.

One point should be noted at the outset is that WKML does not provide a Document Object Model (DOM) type of mechanism for writing the XML file. One of the implications of this is that programmers are required to write their programs in a style in which data are added to the KML file in a sequential manner. On the other hand, although not discussed here, FoX does provide the capability for reading XML using a DOM (Dove, 2009). It should also be noted that programmers need to open the tag and close all tags manually. If any program fails to perform a close operation, the FoX library will generate an error message to report an invalid XML file.

WKML has been developed through a period of time that saw several updates to the KML schema. The version described in this paper is fully compatible with the current OGC standard KML 2.2 schema and therefore is designed to work with all tools that properly follow this standard. On the other hand, although the KML schema has recently been extended to include parts of the GML schema, these are not yet implemented within WKML.

Table 1
Summary of main WKML subroutines that form basic API for the programmer.

Subroutine name	Purpose
General API	*General functions, such as opening and closing files and other book-keeping tasks*
`kmlBeginFile`	Open a new KML file
`kmlFinishFile`	Close an open KML file and ensure that all tags are closed
Geometry API	*Functions for creating geometrical objects, such as points, lines, and polygons*
`kmlCreatePoints`	Create a set of points from an array of data
`kmlCreateLine`	Create a set of lines from an array of data, with an option to close the polygon
`kmlStartRegion`	Create an outer boundary from an array of data values
`kmlAddInnerBoundary`	Create an inner region from an array of data values
`kmlEndRegion`	Close the creation of a region defined by the previous two calls
Two-dimensional field API	*Functions for visualizing two-dimensional fields of data*
`kmlCreateCells`	Create a grid of cells on a grid colored according to the third dimension of an array of data
`kmlCreateContours`	Create a contour plot from a grid of cells with a third dimension of an array of data
Color-handling API	*Functions for manipulating the colors of objects*
`kmlGetCustomColor`	Select a color from the X11 color palette
`kmlSetCustomColor`	Select a color using the standard 8-digit hexadecimal palette
Styling API	*Functions for manipulating the styles of objects, such as widths*
`kmlCreatePointStyle`	Style individual points
`kmlCreateLineStyle`	Style individual lines
`kmlCreatePolygonStyle`	Style individual polygons
Chart API	*Functions for adding charts*
`kmlAddChart`	Create a chart using the Google Earth API
`kmlAddChart_gnuplot`	Create a chart using the gnuplot API

2.2. File structure

XML files, as illustrated in Listing 1, have a well-defined structure, particularly with regard to the syntax of the file header and the end of the file. WKML provides one subroutine call, `kmlBeginFile`, to open an output KML file and write the appropriate header information, and another subroutine call, `kmlFinishFile`, to close the file correctly. It is possible for a program to write to several KML files at any point, and WKML provides a data type that specifies a specific Fortran channel number associated with each file. This is associated with the `xmlf_t` type in Listing 1, with the associated variable (`xfile` in Listing 1) defining the write channel.

We note that WKML can support nested structures such as folders, but through the lower-level functions rather than through the public API described here.

2.3. WKML primitives

The simplest tasks that can be imagine are to create some basic primitive elements within the KML document, such as to place a marker (point) at a specific geographical location, to connect two points with a line, or to encompass a geographical area with a polygon. The first two can be accomplished with the `kmlCreatePoints` subroutine shown in Listing 1, and the `kmlCreateLine` subroutine. An example of using `kmlCreatePoints` for a geophysical application is shown in Fig. 1. The `kmlCreateLine` subroutine call has the form

```
call kmlCreateLine(xfile, lat_long_positions, &
    color="red", linewidth=5)
```

The features of points and lines, such as color, width and style can be defined with additional and optional subroutine call arguments, as in this example, or with a prior call to a style subroutine such as `kmlCreatePointStyle` or `kmlCreateLinetyle`. It is also possible to add text description to a point or line.

Creating a polygon can be achieved using the subroutine `kmlCreateLine` with an argument that tells the subroutine to close the loop around a series of points. However, users may frequently want to draw polygons in a more sophisticated manner, particularly to create multiple inner boundaries. An example can be seen in Fig. 2. The following snippet of code shows how this can be implemented within a Fortran program:

```
call kmlStartRegion(xfile, longitude_out, &
  latitude_out, altitude=zed_out)
call kmlAddInnerBoundary(xfile, longitude_in, &
  latitude_in, altitude=zed_in)
call kmlEndRegion(xfile)
```

The effect of creating filled regions with inner boundaries is achieved using the three subroutine calls shown in this listing.

[2] The authors of WKML are happy to collaborate with users to implement new feature requests, and also are happy to include new features added independently in subsequent version updates.

These subroutines are used in WKML's contour algorithms discussed next.

2.4. Handling two-dimensional data

One common use case is the need to overlay a map with a representation of an array of gridded data, such as surface temperature, depth of the water table, or a density of specific flora or fauna. This is not a feature that is native to KML, and thus we have created two approaches (this being part of the added value of WKML pointed out in the introduction). The first, using the subroutine `kmlCreateCells`, is to create a grid of cells with an associated value, which can be represented either as a color or a vertical elevation. This is illustrated in the following snippet of code:

```
myci(:) = kmlGetCustomColor(colr)
call kmlCreateCells(xfile, xygrid, field, myci)
```

where the two-dimensional array `xygrid` contains the x–y grid values (but note that the longitudinal and latitude coordinates can be passed across as two one-dimensional arrays), `field` contains the array of values that will color the cells, and `myci` is an array that contains the color map.

The problem with gridded cells is that file sizes can become unreasonably large, particularly with fine grids. One solution is to be able to create polygons where neighboring cells have the same color or height, and to achieve this we have developed a contour routine within WKML. This only requires a simple call to a subroutine called `kmlCreateContours`, with options to control the look of the contour map:

```
call kmlCreateContours(xfile, grid, field, myci)
```

To accomplish the creation of a contour map in a robust manner (for example, avoiding problems with boundaries) we used the algorithm of van Snyder et al. (1978). The resultant contour map uses the polygonal region routines described in the previous subsection.

2.5. Graphs and charts

Geobrowser tools may enable graphs and charts to be incorporated within the map plot. WKML uses two approaches to implement this capability, namely the use of the Google Chart API and the use of gnuplot as an external tool. An example using the first approach is shown in Fig. 3.

The Google Chart API (Google, 2010) was released at the beginning of 2008 (renamed to Google Chart Tools in 2010), and this can be integrated within KML. In short, the Google Chart API merely needs a server URL including values for several variables such as chart size, chart data, and chart type. The WKML library wraps the interface to the Google Chart API within the `kmlCreatePoints` subroutine we met previously, allowing information such as chart type, chart data, chart scale to be passed as variables as in this example:

```
call kmlCreatePoints(xfile, grid, &
charttype=type, chartsize=size, chartdata=data)
```

There are two issues about this approach that merit comment. First is the fact the Google Chart API only supports just over 2000 characters, which limits the ability of KML to represent data for plotting. In practice, therefore, users would need to decide on the granularity of the chart. However, a hydrological simulation, by way of example, may generate many more data records than can be supported by the character limit imposed by the API. The additional granularity parameter would complicate the routines. The second issue is that not all versions of Google Earth properly support the Google Chart API. For example, our own tests showed that it worked fine on version 4.2 for the Microsoft Windows and Mac OS X operating systems, but since version 4.3 (including 5.0) the Windows version has not been working properly (the OS X version works fine).

An alternative approach is to use gnuplot, which has the advantage of being a package that is widely in most Linux/Unix (including Mac OS X) systems and can be run from the command shell (Crawford et al., 2010). The subroutine `kmlAddChart_gnuplot` has been provided within WKML, and is called as in this

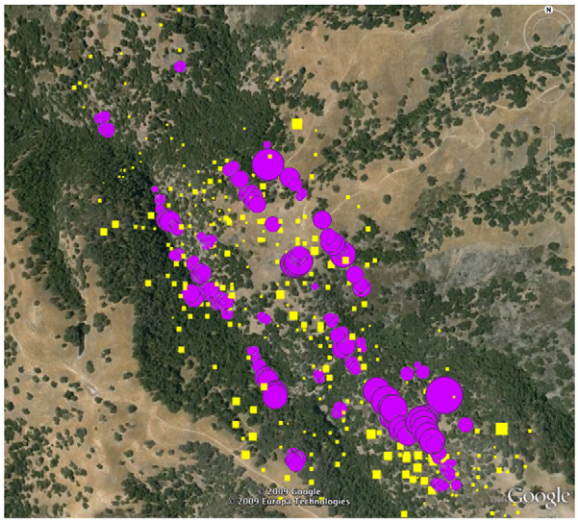

Fig. 1. Use of points to highlight earthquake activity in an area of Northern California as generated by HypoDD program, showing earthquake hypocenters as initial estimates (circles) and computed (squares) with test data provided by program web site.

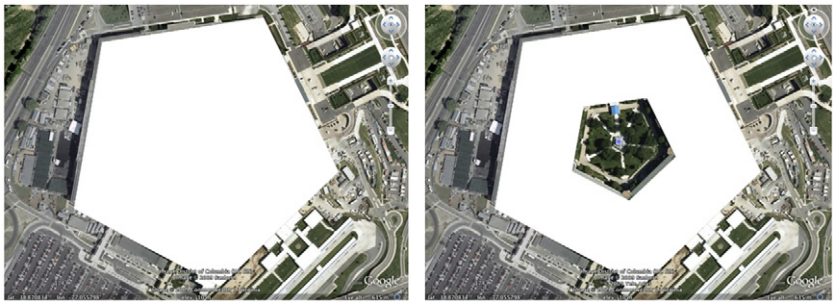

Fig. 2. Demonstration of using polygon fills with inner boundaries. Left image shows how we have used WKML to create an image mask over a polygonal-shaped building, and right image shows use of an inner boundary to reveal building's courtyard.

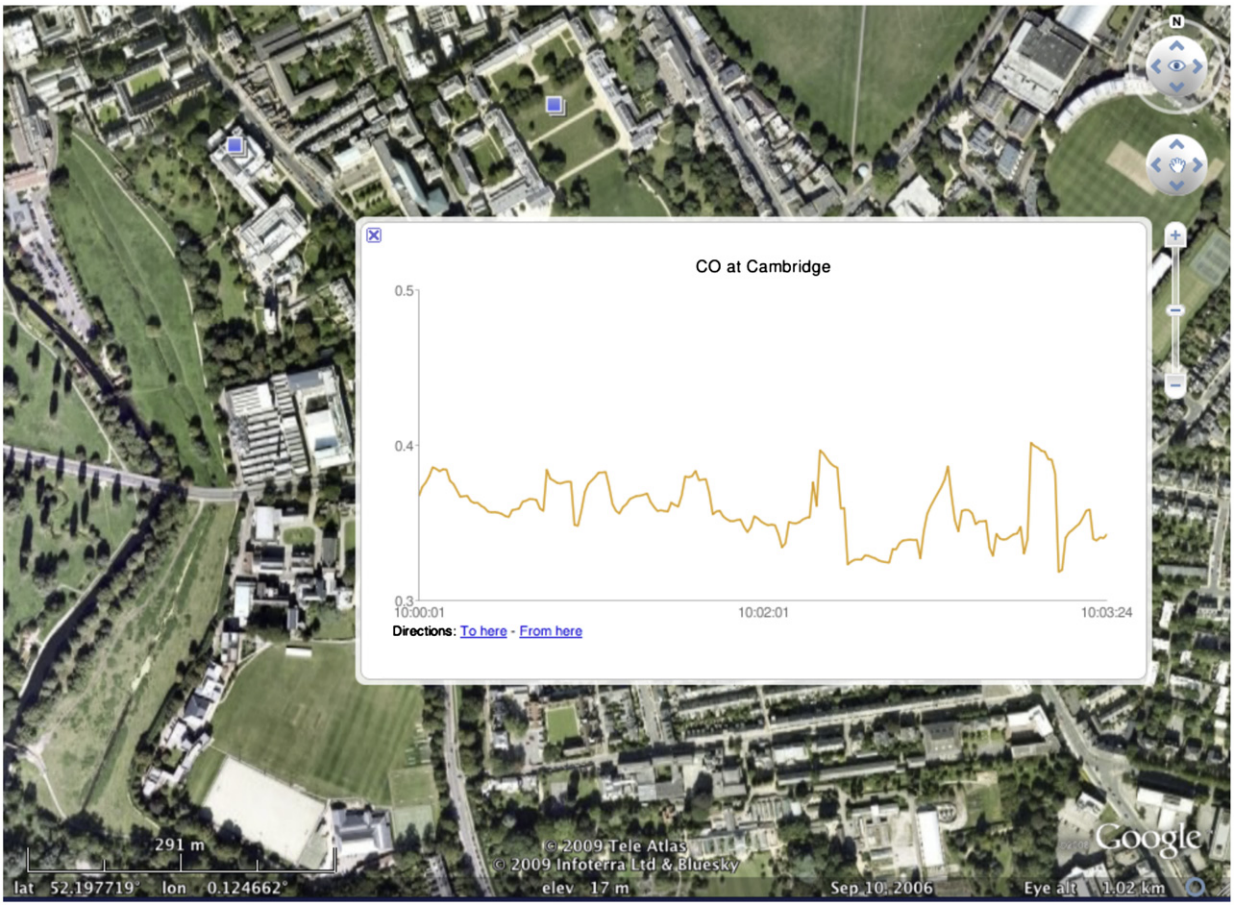

Fig. 3. Example of use of a Google chart. This plot shows temporal variation of an urban pollutant associated with a point in the city of Cambridge, UK. Data courtesy of Dr Mark Calleja, Cambridge eScience Centre.

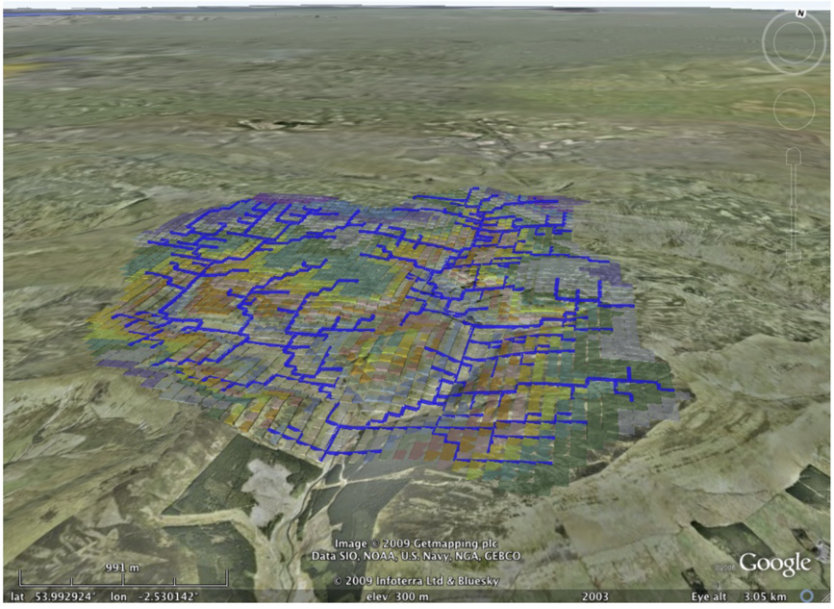

Fig. 4. Plot of river links (dark blue) within Dunsop Catchment overlain onto digital terrain model. Each grid cell measures $100 \times 100\,m^2$. Colors reflect surface height, with white corresponding to lower heights and green to higher land. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

example snippet of code:

```
call kmlAddChart_gnuplot(xfile, longitude, latitude, &
  commands_filename = "plotfile.com")
```

where the last argument points to a gnuplot command file that has to be created in advance. The WKML subroutine calls gnuplot via a system call (as enabled within Fortran 2003, and incorporated into many Fortran 95 compilers currently being used), generating a graphics image (in a standard format such as PNG, EPS etc.) of the chart and writing the file URL to the KML file. An example of the gnuplot command file is

```
set terminal png
set output" discharge.png"
plot" ./output/dunsop-discharge-sim.txt"
exit
```

with commands that request creation of a PNG file, providing the output file name, and thirdly requesting plotting of data held within another file.

3. Case study: using WKML for hydrological modeling

3.1. The SHETRAN hydrological modeling program

SHETRAN (Ewen et al., 2000, 2002) is a three-dimensional (ie including the surface and subsurface structures) physically based hydrology simulation program, which uses a spatially distributed finite difference model to simulate coupled water flow, multifraction sediment transport and multiple reactive solute transport in river basins. It is capable of modeling all phases of the hydrological cycle. Water flow can be simulated following temporal and spatial varying rainfall patterns and other meteorological inputs, accounting for flow on the surface, through stream channels, and in saturated or unsaturated zones. It is also able to model processes such as canopy interception, evapotranspiration, soil erosion and sediment yield arising from rain drop

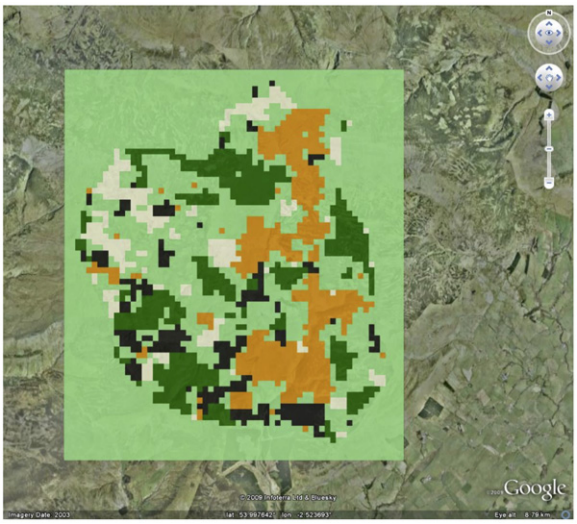

Fig. 6. This KML shows vegetation types generated by using `kmlCreateCells`. Color scheme is gray for urban, orange for deciduous trees, bright green for conifer trees, yellow for crops and light green for grass (the simulations place grass at grid points not included in catchment area). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

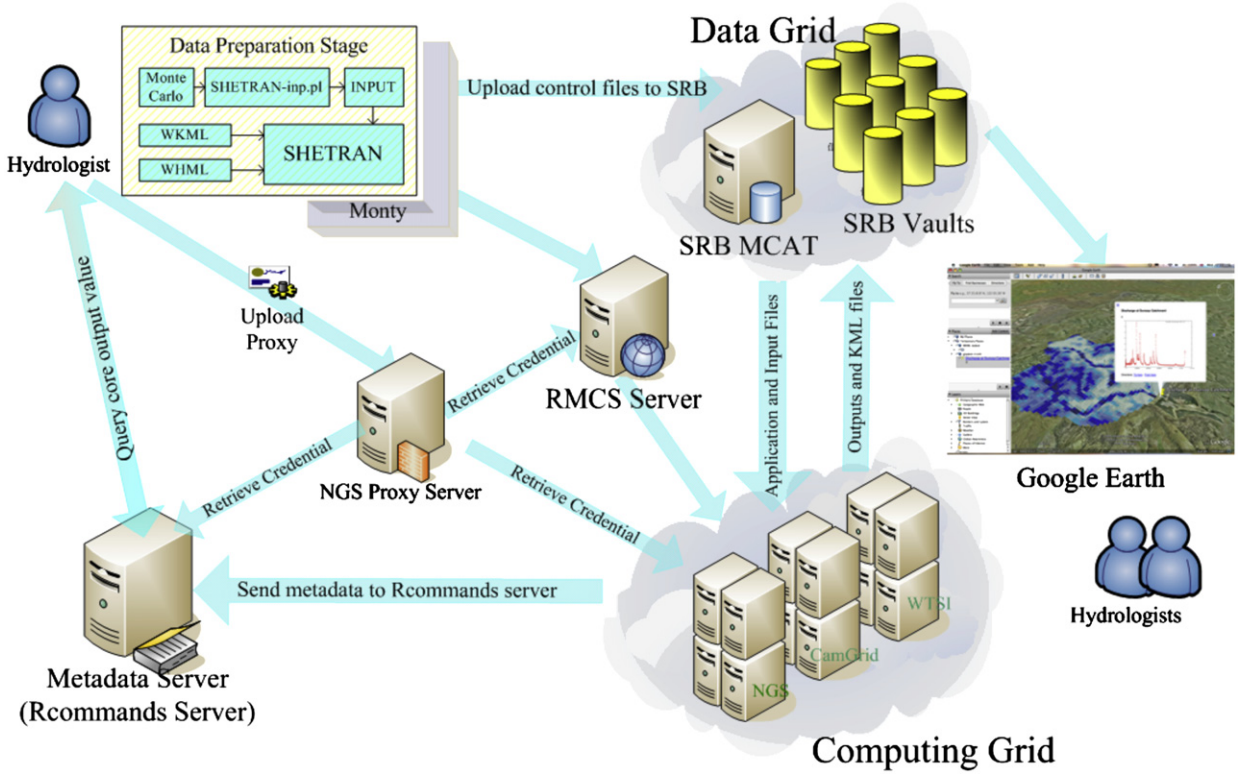

Fig. 5. Representation of workflow associated with running SHETRAN within a grid infrastructure and using the WKML library to analyze outputs. KML files are generated at data preparation stage and as outputs from the simulations. SRB is used for storing all files so that users can view the simulation inputs and outputs using a geobrowser client tool.

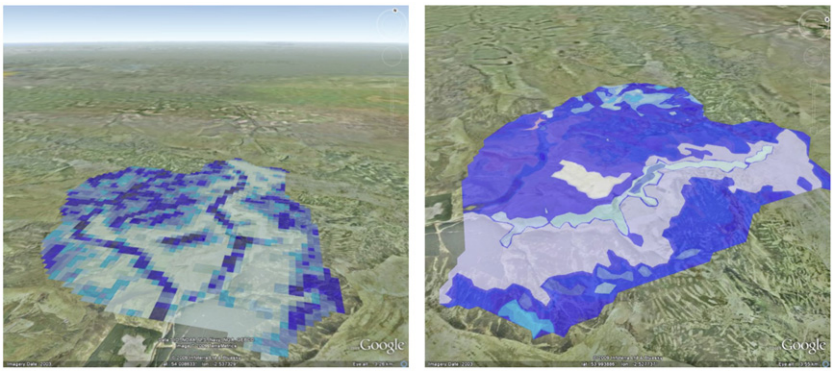

Fig. 7. Comparison of representations of phreatic surface in Dunsop catchment generated at an instantaneous moment during a SHETRAN simulation. Left image shows cell representation (generated using `kmlCreateCells`) and right image shows contour representation (generated using `kmlCreateContours`) Scale runs from dark blue representing water table being at the ground surface to white representing water table lying 1 m below ground surface. Those two images are viewed from different directions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

impact and overland flow. SHETRAN is thus a powerful, high-end research tool. SHETRAN takes as input several parameters that describe the physical characteristics of the catchment, including soil and vegetation types and properties, and gives as output a detailed description in time and space of the flow and transport in the basin. Within SHETRAN the spatial distribution of catchment properties, rainfall input and hydrological response is achieved in the horizontal direction through the representation of the catchment by an orthogonal grid network and in the vertical direction by a column of horizontal layers at each grid square, thereby allowing a representation of the sub-surface. Rivers, or "links", are located along the edges of the grids and fluxes are calculated across all links and grid boundaries. SHETRAN has the capability of simulating a vast amount of output data, including for example, transpiration, evaporation from the soil surface, evaporation from intercepted storage, drainage from intercepted storage, canopy storage, vertical flow, snow pack depth, phreatic (i.e. groundwater) depth below surface, overland flow, surface water depth, the soil water potential and soil water content for one or each river link and/or grid cell for each time step in a simulation.

Initially within the MEDIGRID project, and subsequently as a new project based on the *e* Minerals toolkit, SHETRAN has been incorporated into a grid infrastructure to enable a range of computationally intensive and multiple scenario hydrological experiments to be carried. An example is the use of this infrastructure to carry out the investigation of flood-peak response to changes in land-use and land management practices (O'Connell et al., 2007).

Prior to our work, SHETRAN used text-based and HDF5 (The HDF Group, 2009) formats as output. We have incorporated our WKML library into SHETRAN to produce KML output capability. We do not attempt to replace the HDF5 output file with an XML file because the resultant file would be too large to parse and maintain. Instead we use the KML file for the key data for visualization. We remark here that KML is incomplete for hydrological data description, because it is designed primarily for visualization and is therefore limited in practice in the extent to which it can represent all the information hydrologists require for a complete data description (such as the full state of the catchment at each model time-step). Therefore, we augment the XML file with the use of HydroXC (Piasecki, 2007) through the use of an analogous WHML library. HydroXC handles geo-reference data based on the Geographical Markup Language (GML), but HydroXC is not officially a GML application schema; a description of WHML is beyond the scope of this paper.

3.2. The Dunsop catchment

The Dunsop catchment is an area of size 26 km^2 located in the Forest of Bowland in Lancashire, UK (geographic coordinates 53.99768°, −2.527106°). The Dunsop River is a tributary of the Hodder and Ribble Rivers. Large-scale changes are being made to the land-use and its management, giving a special opportunity to analyze the effects on local and downstream flood risk. The changes being made include woodland planting, blocking of moorland grips, changes in stocking density, and changes in bracken burning policy, all with the aim of preventing further deterioration of the raw water quality and improvement of the condition of the area (which has been designated as a UK Site of Special Scientific Interest).

The Dunsop catchment is represented in Fig. 4, where we have plotted the digital terrain model (DTM) using our WKML tool, drawing rivers and adding colors to indicate the variation in surface height.

In the virtual experiments we are performing using SHETRAN, we are investigating the way that flood-peaks respond to changes in land use and land management practices. A range of physically realistic parameters are being changed in a systematic way to identify which values and combinations of parameters lead to a significant increase in peak river-discharge during storms (Ewen et al., 2000). For example, one will be able to query which sets of parameters lead to a given percentage increase in peak discharge and these physically based parameters will then be interpreted in a hydrological context to determine the possible land use management scenarios or farming practices which might result in these high peak river discharges and therefore high flood-risk scenarios.

3.3. Computing and data grids for simulation jobs

In order to run the proposed range of hydrological experiments we used a grid infrastructure that had previously been the basis of the *e* Minerals project. The computing grid includes the resources of the University of Cambridge Campus Grid,[3] and the UK National Grid Service.[4] The data grid component is based on the Storage Resources Broker,[5] developed by the San Diego Supercomputer

[3] CamGrid http://www.escience.cam.ac.uk/projects/camgrid/, accessed April 21, 2010.
[4] NGS http://www.ngs.ac.uk/, accessed April 21, 2010.
[5] SRB, http://www.sdsc.edu/srb/, accessed April 21, 2010.

Center. In operation, the files created by a job, including the KML files, are stored together with all input files on the SRB directly as part of the job workflow. One important component concerns metadata. Large collections of data, such as can easily be accumulated with grid computing methods, should ideally have a rich set of metadata in order to allow the data to be unambiguously linked with their source, and to enable data to be easily discovered. The *e* Minerals project (Salje et al., 2009) developed the Rcommands system for metadata management, and the grid computing tools used in this work were developed to allow automatic collection of metadata from simulation jobs (Tyer et al., 2006, 2007). One metadata item is the location of the output files on the SRB. In this project the *e* Minerals metadata system has been adapted and exploited for hydrological research.

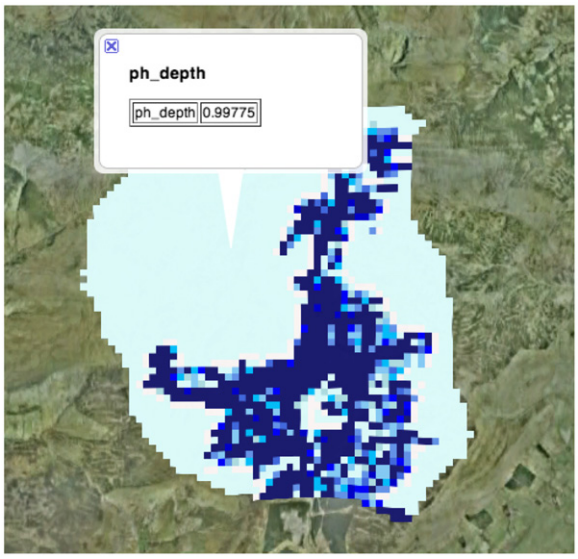

Fig. 8. A screen shot showing a single frame of animation of phreatic surface, with a value for a specific grid cell shown following a mouse click on that cell.

Fig. 5 illustrates the grid infrastructure and the workflow. In the data preparation stage, a large number of different land use scenarios are generated using a Monte Carlo program we developed for this work. We then use an automatic system (called Monty) to compress the input files and application binary into a single control file that is uploaded to the SRB. The job submission is handled using the *e* Minerals RMCS system (Dove et al., 2007; Walker et al., 2009), which allocates the jobs to the computing grid, and creates scripts to manage the data transfer from the SRB at the start of the job and to the SRB at the end of the job. The RMCS system also harvested metadata from the output XML files (Tyer et al., 2007; Salje et al., 2009). Collaborating hydrologists can access the KML files for automatic visualization of the simulation results.

3.4. Incorporating WKML into SHETRAN

SHETRAN requires visualizing of several types of gridded data, such as the DTM of the catchment area, the land use and vegetation types, the level of water table, or the net rainfall across the catchment area. Passing these data to the KML file generated directly by SHETRAN can be achieved using the WKML `kmlCreateCells` subroutine. An example plot of one of the vegetation models is shown in Fig. 6. It is also required to plot the river links, which we represent as lying along edges of the grid cells. These can be plotted using the `kmlCreateLine` subroutine, but modified in one critical way. The standard `kmlCreateLine` subroutine is given a one-dimensional array containing

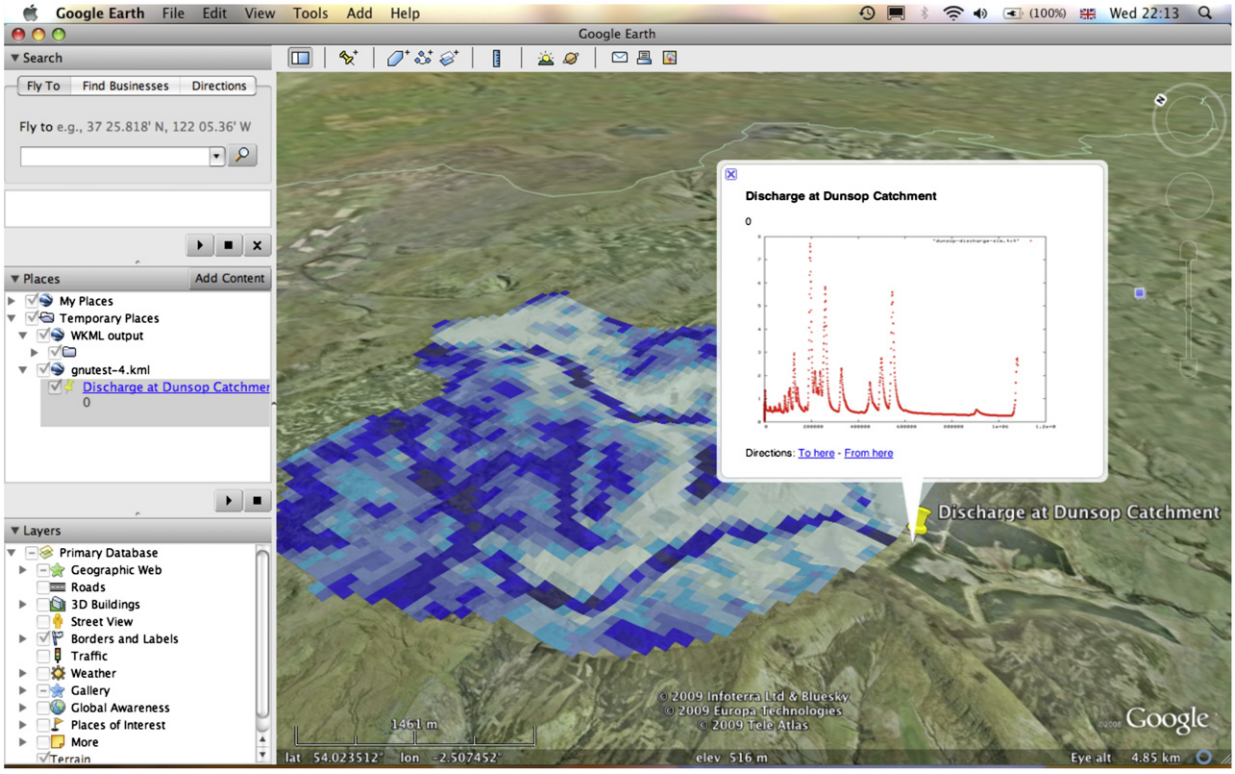

Fig. 9. Hydrograph generated using `kmlAddChart_gnuplot` subroutine from a SHETRAN simulation of Dunsop catchment. Colors of the grid cells represent the phreatic surface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the coordinates of each node of the line. Moreover, SHETRAN has to plot a number of river links, for which kmlCreateLine is essential. An example was previously shown in Fig. 4. The river links as shown as the blue lines overlaying the DTM of Dunsop catchment.

The two subroutines `kmlCreateCells` and `kmlCreateContours` can be used to generate visualizations of the simulation's outputs. Fig. 7 compares both representations of a snapshot of the instantaneous phreatic surface (defined as the depth below ground to the water table) generated by both functions. In practice images such as those shown in Fig. 7 need to be generated as a time sequence in order to produce animations. The two subroutines `kmlCreateCells` and `kmlCreateContours` can take an argument that defines the time according to ISO8601 standard.

The KML 2.2 schema (Wernecke, 2009) now provides the new tag `<extendeddata>` to represent the actual data value represented by a color instead of just the hexadecimal color code. Exploiting this, it is possible to allow a mouse click on a cell placemark to see the table of a data value. Fig. 8 shows a single frame of an animation based on the phreatic surface and a value associated with a specific grid cell.

Finally, it is necessary to plot the hydrograph at the outlet of the catchment. Hydrologists use hydrographs to display the hydrological variables over time. One of the most interesting hydrographs in this project is the river discharge hydrograph. Fig. 9 shows a hydrograph of water discharge data at the outlet of the Dunsop River, generated using the `kmlAddChart_gnuplot` subroutine as discussed in Section 2.5.

4. Discussion

In this paper we have described the WKML library for producing KML files as outputs from Fortran programs suitable for viewing using a geobrowser client tool. Our motivation has been to make this process as easy as possible for the Fortran programmer, reducing the need for the programmer to know about the syntax of KML or XML. Instead all the programmer needs to know is the format of some subroutine calls, which have been designed to be as intuitive as possible. We have illustrated some of the functionality of WKML using some simple examples and a more detailed case study based on the SHETRAN hydrology code operating within a grid computing infrastructure. This use case is interesting because KML representations of data are required for all stages in the job workflow, from the definition of the scientific virtual experiment, to the preparation of data, and for the simulation outputs. We have stressed that the approach is not tied to any particular application area (the range of examples demonstrates this point).

The coordinate system of WKML simply uses longitude/latitude, it does not support any map projection. If trying to rendering a large data set (500×500 grids), it takes about 3 min to display. Thus, users need to consider the performance of using KML when trying to plot large data set.

WKML is freely available from http://web.esc.cam.ac.uk/xml/kml.html together with on-line documentation. It is released as an open-source product, in anticipation that if it proves to be useful to the scientific community additional contributions from other people will follow.

Acknowledgement

We are pleased to acknowledge funding from NERC (UK) for this work.

References

Bovolo, C.I., Abelea, S.J., Bathursta, J.C., Caballerob, D., Ciglanc, M., Eftichidisd, G., Simo, B., 2009. A distributed framework for multi-risk assessment of natural hazards used to model the effects of forest fire on hydrology and sediment yield. Computers & Geosciences 35 (5), 924–945. doi:10.1016/j.cageo.2007.10.010.

Chiang, G.-T., White, T.O.H., Dove, M.T., 2007. Driving Google Earth from Fortran. In: Proceedings of the UK e-Science All Hands Meeting 2007, Nottingham, UK, pp. 236–243. URL: 〈http://www.allhands.org.uk/2007/proceedings/papers/825.pdf〉, accessed April 21, 2010.

Chiang, G.-T., White, T.O.H., Dove, M.T., 2009. Geospatial visualization tool kit for scientists using Fortran. Eos 90 (29), 249–250.

Crawford, D., 2010. Gnuplot 4.4—an interactive plotting program 〈http://www.gnuplot.info/docs/gnuplot.html, accessed April 21, 2010.

Dove, M.T., Walker, A.M., White, T.O.H., Bruin, R.P., Austen, K.F., Frame, I., Chiang, G.-T., Murray-Rust, P., Tyer, R.P., Couch, P.A., Kleese van Dam, K., Parker, S.C., Marmier, A., Arrouvel, C., 2007. Usable grid infrastructures: practical experiences from the *e* Minerals project. In: Proceedings of the UK e-Science All Hands Meeting 2007, Nottingham, UK, pp. 48–55. URL: 〈http://www.allhands.org.uk/2007/proceedings/papers/831.pdf〉, accessed April 21, 2010.

Dove, M.T., FoX and Fortran 〈http://web.esc.cam.ac.uk/xml/fox.html〉, accessed April 21, 2010.

Ewen, J., Bathurst, J.C., Parkin, G., O'Connell, P.E., Birkinshaw, S.J., Adams, R., Hiley, R., Kilsby, C., Burton, A., 2002. SHETRAN: physically-based distributed river basin modelling system. In: Singh, V.P., Frevert, D.K. (Eds.), Mathematical Models of Small Watershed Hydrology and Applications. Water Resources Publications, LLC, Highlands Ranch, Colorado, USA, pp. 43–68.

Ewen, J., Parkin, G., O'Connell, P.E., 2000. SHETRAN: distributed river basin flow and transport modeling system. American Society of Civil Engineers Journal of Hydrologic Engineering 5 (3), 250–258. doi:10.1061/(ASCE)1084–0699(2000)5:3(250).

Frame, D.J., Aina, T., Christensen, C.M., Faull, N.E., Knight, S.H.E., Piani, C., Rosier, S.M., Yamazaki, K., Yamazaki, Y., Allen, M.R., 2009. The climateprediction.net BBC climate change experiment: design of the coupled model ensemble. Philosophical Transactions of the Royal Society A 367 (1890), 1041–1046. doi:10.1098/rsta.2008.0240.

Gahegana, M., Luob, J., Weaver, S.D., Pike, W., Banchuenb, T., 2009. Connecting GEON: making sense of the myriad resources, researchers and concepts that comprise a geoscience cyberinfrastructure. Computers & Geosciences 35, 836–854.

Google, 2010. Google Chart Tool 〈http://code.google.com/apis/charttools/〉, accessed April 21, 2010.

Holt, J., Harle, J., Proctor, R., Michel, S., Ashworth, M., Batstone, C., Allen, I., Holmes, R., Smyth, T., Haines, K., Bretherton, D., Smith, G., 2009. Modelling the global coastal ocean. Philosophical Transactions of the Royal Society A 367 (1890), 939–951. doi:10.1098/rsta.2008.0210.

Lenton, T.M., Myerscough, R.J., Marsh, R., Livina, V.N., Price, A.R., Cox, S.J., 2009. Using GENIE to study a tipping point in the climate system. Philosophical Transactions of the Royal Society A 367 (1890), 871–884. doi:10.1098/rsta.2008.0171.

O'Connell, E., Ewen, J., O'Donnell, G., Quinn, P., 2007. Is there a link between agricultural land-use management and flooding? Hydrology and Earth System Sciences 11 (1) 96–107.

Piasecki, M., 2007. HYDROXC, 〈http://www.weather.gov/oh/hydroxc/, accessed April 21, 2010.

Renard, P., Badoux, V., 2009. Grid computing for Earth science. Eos 90 (14), 117–119.

Salje, E.K.H., Artacho, E., Austen, K.F., Bruin, R.P., Calleja, M., Chappell, H.F., Chiang, G.-T., Dove, M.T., Frame, I., Goodwin, A.L., Kleese van Dam, K., Marmier, A., Parker, S.C., Pruneda, J.M., Todorov, I.T., Trachenko, K., Tyer, R.P., Walker, A.M., White, T.O.H., 2009. eScience for molecular-scale simulations and the *e* Minerals project. Philosophical Transactions of the Royal Society A 367 (1890), 967–985. doi:10.1098/rsta.2008.0195.

The HDF Group, 2009. HDF5, 〈http://www.hdfgroup.org/HDF5/〉, accessed April 21, 2010.

Tyer, R.P., Couch, P.A., Kleese van Dam, K., Todorov, I.T., Bruin, R.P., White, T.O.H., Walker, A.M., Austen, K.F., Dove, M.T., Blanchard, M.O., 2006. Automatic metadata capture and grid computing. In: Proceedings of the UK e-Science All Hands Meeting 2006, Nottingham, UK, pp. 381–384. URL: 〈http://www.allhands.org.uk/2006/proceedings/papers/650.pdf〉, accessed April 21, 2010.

Tyer, R.P., Couch, P.A., Mortimer-Jones, T.V., Kleese van Dam, K., Todorov, I.T., Bruin, R.P., White, T.O.H., Walker, A.M., Austen, K.F., Dove, M.T., 2007. Metadata management and grid computing within the eMinerals project. In: Proceedings of the UK e-Science All Hands Meeting 2007, Nottingham, UK, pp. 415–422. URL: 〈http://www.allhands.org.uk/2007/proceedings/papers/832.pdf, accessed April 21, 2010.

van Snyder, W., 1978. Algorithm 531: contour plotting [J6]. ACM Transactions on Mathematical Software 4, 290–294 . doi:10.1145/355791.355800.

Walker, A.M., Bruin, R.P., Dove, M.T., White, T.O.H., Kleese van Dam, K., Tyer, R.P., 2009. Integrating computing, data and collaboration grids: the RMCS tool. Philosophical Transactions of the Royal Society A 367, 1047–1050. doi:10.1098/rsta.2008.015.

Wernecke, J., 2009. The KML Handbook: Geographic Visualization for the Web, first ed. Addison-Wesley Professional, Pearson Education, Inc, Boston, MA 368pp.

White, T.O.H., Bruin, R.P., Chiang, G.-T., Dove, M.T., Tyer, R.P., Walker, A.M., 2009. Lessons in scientific data interoperability: XML and the eMinerals project. Philosophical Transactions of the Royal Society A 367, 1041–1046. doi:10.1098/rsta.2008.0175.

White, T.O.H., Dove, M.T., Bruin, R.P., Austen, K.F., Walker, A.M., Artacho, E., Murray-Rust, P., Walkingshaw, A.D., Marmier, A., Parker, S.C., Couch, P.A., Tyer, R.P., Todorov, I.T., Wilson, D.J., 2006a. Information delivery in computational mineral science: the eMinerals data handling system. In: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing 2006. Amsterdam, Netherlands, pp. 59. doi:10.1109/E-SCIENCE.2006.261143.

White, T.O.H., Murray-Rust, P., Couch, P.A., Tyer, R.P., Bruin, R.P., Todorov. I.T., Wilson, D.J., Dove, M.T., Austen, K.F., Parker, S.C., 2006b. Application and uses of CML within the *e* Minerals project. In: Proceedings of the UK e-Science All Hands Meeting 2006, Nottingham, UK, pp. 606–613. URL: 〈http://www.allhands.org.uk/2006/proceedings/papers/666.pdf〉, accessed April 21, 2010.