

Value-added Services for 3D City Models using Cloud Computing

Javier HERRERUELA, Claus NAGEL, Thomas H. KOLBE
(javier.herreruela | claus.nagel | thomas.kolbe)@tu-berlin.de
Institute for Geodesy and Geoinformation Science, Technical University Berlin

Abstract. An increasing number of 3D city models is being offered for viewing purposes on different web platforms. Adding value to these models beyond visualization is a task cloud services offer optimal support for. By exporting the project relevant part of the model and uploading it to the cloud (possibly along some model-linked documents) resources like storage space and access control are outsourced to the cloud services while keeping the original data safe. The exported models can then be collaboratively modified and enriched by a work team without altering the original official 3D model administered by the city.

Keywords. 3D City Models, Cloud Computing, Collaborative Urban Planning, CityGML

1. Introduction

Many municipalities provide access to their 3D city models on the Internet today. Their usage is typically restricted to interactive viewing, because only a graphical 3D representation of the model is being provided from a dedicated website of the municipality or by an Earth Browser's warehouse (the best known being Google Earth).

However, many more applications would become feasible, if the 3D city models could be augmented with individual thematic information and additional 3D model content. For example, an architect could embed a newly planned building within the existing 3D city model. This would require a functionality to extend the 3D model by additional 3D contents. That use case may also include the requirement to virtually demolish existing buildings in the city model in order to show the new design without interference with the old buildings / constructions. The architect may then want to add thematic information to the newly planned 3D objects, but also for some of the surrounding buildings, which can be visualized and explored by the stakeholders. While using the publicly accessible 3D city model on the one side, the architect wants to be sure on the other side, that his designs are only visible (and discussed) by a selected group of people.

2. New possibilities offered by cloud services

This sketched support for architecture and urban planning using 3D city models has been already investigated by some other groups before. Typically this requires read/write access to the underlying GIS or spatial databases (see e.g. Ross et al. [1]),

but this is generally neither supported nor wanted by the municipalities or their cadastral departments. First, these bodies want to make sure that the original dataset remains unchanged and safe from unauthorized access. Second, it is not in their responsibility to run a server-based platform to maintain the designs and changes of an urban planner or architect.

Hence, using a 3D city model for such complex tasks poses several challenges to the bodies administering the 3D city models and their IT departments.

- Make parts of the model accessible to only a selected group of people.
- Enhance the model with information having a spatial context.
- Allow model changes for alternative content without altering the original.
- Implement all that without too much technical and without cost overhead.

Cloud Services offer a suitable solution for these problems.

- They allow instant access to those people given the URL.
- Fine-grained access control (read/write rights) is set by the uploader (document owner) and handled by the cloud service.
- The uploaded model can be linked on a per object basis with other documents or files also hosted in the cloud, thus setting (or increasing) the information related to a spatial context.
- No server has to be set up or run for hosting the changes of the participants. Hosting is done in the cloud. Committed changes stay in the cloud. The official 3D city model hosted by the municipality or similar organization remains untouched.

Most of these features are woven into the fabrics of what a cloud service is and how it works, so resorting to cloud services for virtual work with 3D city models seems the next natural step.

The concepts presented in this article aim at solving common issues that arise around collaborative urban planning within the general set-up of public, sometimes even freely available, 3D city models: these are usually restricted to viewing functionality with no feedback allowed. Enhancing the models without cloud support is an arduous task, doing it at no additional cost becomes impossible.

The focus is not on saving 3D city models (or parts) in the cloud or using cloud computing resources for outsourcing complex calculations needed for rendering or other high-profile purposes, but on applying already existing cloud technologies in order to add flexibility to otherwise static city models with user- or group specific information in a spatial context. This is especially appropriate for closed 3D city models that allow no modifications at all, since all intended changes will happen *virtually* in the cloud and only be visible (and feasible) within the client application.

3. Components and their interaction

For the combined usage of 3D city models with cloud services, the following components are required and linked together:

- A visualization model exported from the official 3D city model. This can be the entire city model or just a subset (e.g. redevelopment area).

The final integration of all information parts takes place only within the client, usually a browser, which connects to all components separately. The application logic needed to put them together is delivered by a web page containing a visualization plugin for the 3D model integrated with at least some basic scripting functionality in order to interact with the spatial context.

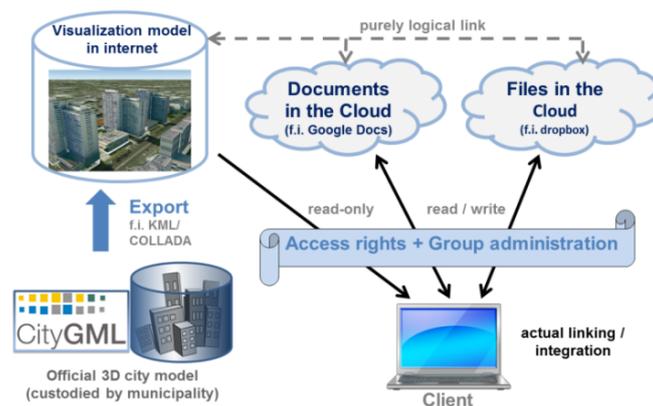


Figure 2. Interaction between 3D city model and cloud services

4. Putting collaborative work into practice

Accomplishing collaborative work on 3D city models and related documents and files in the cloud starts off with a plain two-step procedure:

- 1) Export of a visualization model from the official 3D city model and upload of all related documents and files to the cloud.
- 2) Interpretation of the spatial context, that is, logical linking of the visualization model with the corresponding documents, all represented by their individual URLs, within the web client application.

The whole spatial context including modifiers (for camera viewpoint or hidden or added buildings) is encoded within the web client URL. This is adapted from the concept of Cooperative Public Web Maps, which has been published earlier (Kolbe et al. [2]). As in there, the structure of a URL is defined by the internet standard RFC 2396 [3]:

```
<scheme>://<authority><path>?<query>
```

Where <scheme> specifies the underlying protocol (default: http), <authority> contains the authentication parameters username and password, <path> points to the server and website containing the 3D visualization plugin and integrated logic for handling, and <query> is the dynamic part used to store the spatial context plus modifiers as variable parameters that can be fed to the visualization plugin (e.g.: camera viewpoint) or to the interacting logic (e.g.: hidden buildings).

By storing all configuration data in the context within the URL the user can control the distribution of his individual set of integrated information assets. Different people may have different dynamic parameters (different camera viewpoints, different sets of

hidden buildings...), and thus different URLs all referring to the same 3D city model export in the cloud. These URLs containing specific settings can be bookmarked in the web browser or communicated to the working group by email, as a hyperlink on a web page or in a social network for a later display of the visualization model under the exact same rendering and semantic conditions created by that specific user.

Additionally, fine-grained access rights for individuals and groups are set by the document and file uploader (owner), deciding who is allowed to modify the cloud document or file contents and who is only allowed to read them. The cloud service handles the access control according to the policies set by the owner.

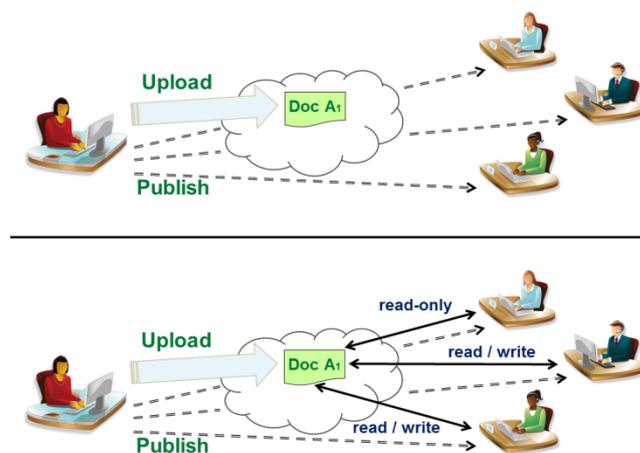


Figure 3. Other users can work with the model according to the rights set by the owner

Work will be committed in the cloud. This can happen in a synchronous (all modifications are immediately visible to all participants) or asynchronous way (document or file gets locked during modification and must be released again for all others to see). In this last case it is possible that a new document or file with a new URL and possibly new owner and access rights is created in the cloud. The new URL should be distributed again to the participants.

5. Usecase example

On the basis of a 3D city model accessible on a running server, the next steps must be followed in order to enhance and work with an augmented model in the cloud:

- 1) Choose an area of interest and export the 3D city model contents of that area in a suitable visualization format like KML/COLLADA, X3D, etc. Each exported city object should be individually accessible and unambiguously identifiable by its ID. Upload this export to a cloud service and keep the URL.
- 2) Export some thematic attributes from the city objects inside that same area of interest in the form of a spreadsheet or some other collaborative shareable document. The exported document must also include the unique IDs for each

- city object it contains. Upload this document to a cloud service (possibly a different one) and also keep the URL.
- 3) Set up an elementary web page including a plugin for the visualization format the 3D city model was exported in integrated with the URLs of the uploaded visualization model and document(s). This logical integration of the spatial context can take place by means of some scripting language supported by the browser, like for example JavaScript.
 - 4) Open the web page in a browser. From this moment on the browser becomes a client application interacting directly with the exports from the original model in the cloud and only there. There is no interaction with the server containing the original 3D city model. From this point of view all work will be accomplished serverless.

Images of the following example correspond to the Berlin 3D city model. Parts of this model as well as some open source tools (3D City Database, Importer / Exporter Tool, citygml4j...) can be downloaded from the official 3D City Database website [4].

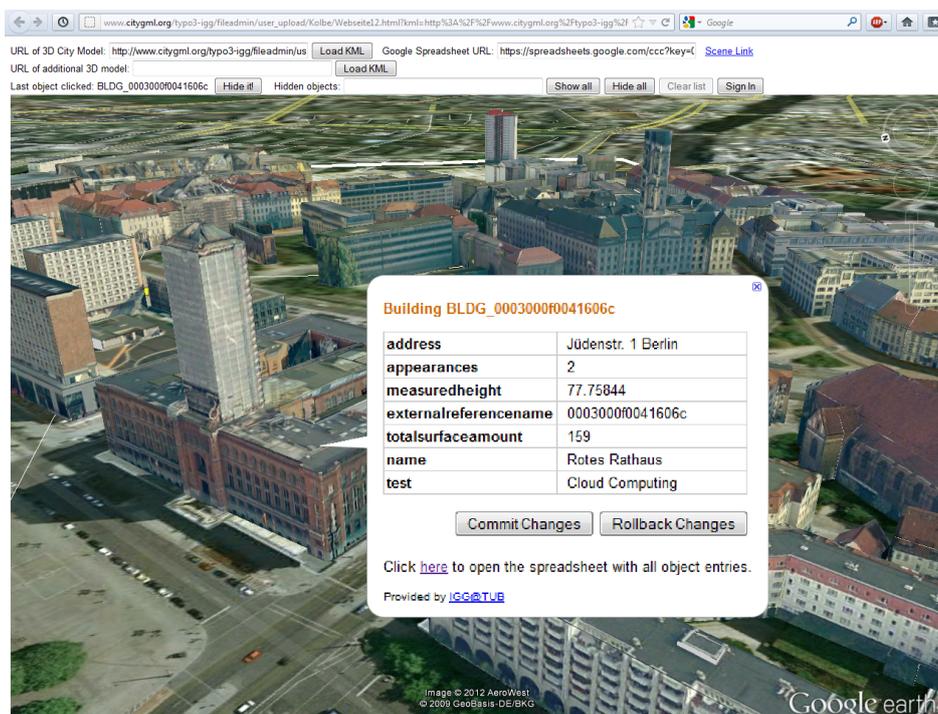


Figure 4. Dynamic content from the cloud attached to the visualization model

Thematic attributes of city objects updated in the document stored in the cloud can be linked to be displayed on the visualization model (also from the cloud) as shown in Figure 4. Their updates can be immediately reflected (aside from the cloud service own latency) when the client requests the contents of the thematic attributes linked to those specific city objects again.

Another interesting possibility of model modification is the ability to remove existing objects from the city model or replace them with alternatives and see how the alternatives look in their expected environment. The visualization plugin API must offer support for these tasks, with corresponding functions callable from the containing web page.



Figures 5a. Model virtually deleted from the context **5b.** Model from user repository added in the context

After the model is adapted as intended, the new resulting model resides exclusively in the cloud (no updates were done in the original 3D city model at all) and it can be made accessible to the rest of the participants in the collaborative planning by just publishing the new resulting URL to them by the usual means - email, hyperlink on a web page or in a social network -, and a new work iteration cycle can begin.

6. Summary and Perspective

Linking different information sources by using cloud computing is not new. However, the presented concept allows supporting collaboration on and augmentation of 3D city models that wouldn't normally support modification or augmentations by the providers. Especially in the case of 3D city models offered by municipalities in internet.

The presented approach allows the definition of contexts in which users can combine an arbitrary (portion of) 3D city model, additional information resources on the 3D objects virtually hidden or virtually added and a predefined camera perspective.

Since the entire context is represented within a URL it can be easily stored in a web browser, exchanged as a link via e-mail or published as a hyperlink on websites. The advantages of this method are that all modifications and additions do not affect the original model; all changes take place exclusively in the cloud. And for this same reason they do not even require permission by the provider

Coupling 3D city models with cloud services paves the way for a wide range of value-added distributed applications bound to 3D city models that can benefit from the

ease of use and low maintenance of cloud services, where lots of information can be exchanged via simple URLs.

These value-added applications have an unlimited flexibility since they are defined by the documents or files linked logically to the exported 3D city model excerpt. By using unique identifiers for the 3D city objects any kind of information or media (films, flash apps...) can be attached to each individual city object.

References

- [1] Ross, L., Döllner, J., Kleinschmit, B. & Schroth, O. (2009): E-Collaboration between the private and the civil Sector: Support of long-term Utilization and update of Official 3D City models. Geoweb 2009 Conference in Vancouver, Canada.
- [2] Kolbe, T. H. / Steinrücken, J. / Plümer, L. (2003): Cooperative Public Web Maps. International Cartographic Congress ICC 2003 in Durban, South Africa.
- [3] T. Berners-Lee, R. Fielding, U. C. Irvine, L. Masinter: Uniform Resource Identifiers (URI): Generic Syntax, Internet Society Standard RFC 2396 (1998)
- [4] 3DCityDB Home, Weblink (accessed January 2012): <http://www.3dcitydb.net>