Service-oriented, standards-based 3D geovisualization: Potential and challenges

Dieter Hildebrandt*, Jürgen Döllner

Hasso-Plattner-Institut, University of Potsdam, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

ARTICLE INFO

Keywords:
Service-oriented architecture
Spatial data infrastructure
Geovisualization
3D geovirtual environment
Virtual 3D city model
3D computer graphics

ABSTRACT

The application of the architectural concept of service-oriented architectures (SOA) in combination with open standards when building distributed, 3D geovisualization systems offers the potential to cover and take advantage of the opportunities and demands created by the rise of ubiquitous computer networks and the Internet as well as to overcome prevalent interoperability barriers. In this paper, based on a literature study and our own experiences, we discuss the potential and challenges that arise when building standards-based, distributed systems according to the SOA paradigm for 3D geovisualization, with a particular focus on 3D geovirtual environments and virtual 3D city models. First, we briefly introduce fundamentals of the SOA paradigm, identify requirements for service-oriented 3D geovisualization systems, and present an architectural framework that relates SOA concepts, geovisualization concepts, and standardization proposals by the Open Geospatial Consortium in a common frame of reference. Next, we discuss the potential and challenges driven by the SOA paradigm on four different levels of abstraction, namely service fundamentals, service composition, interaction services, performance, and overarching aspects, and we discuss those driven by standardization. We further exemplify and substantiate the discussion in the scope of a case study and the image-based provisioning of and interaction with visual representations of remote virtual 3D city models.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

Within the domain of geovisualization, the rise of the Internet and ubiquitous computer networks in general creates new opportunities and new demands for visualizing and interacting with geospatial information. The amount of geospatial data (geodata) that is routinely collected is increasing dramatically (Muntz et al., 2003; Nöllenburg, 2007). The Internet has become the prominent medium through which geodata and maps are disseminated (MacEachren & Kraak, 2001). Besides geodata, massive amounts of computing functionality and processing power are increasingly available as distributed resources that can be accessed through the Internet (Armbrust et al., 2009; Foster & Kesselman, 2003). In many application domains and for many tasks, large amounts of geodata from different sources and a substantial amount of processing power and computing functionality operating on the geodata must be accessible for many, potentially collaborating users in different places (Brodie, Blechschmidt, Fairbairn, Kemp, & Schroeder, 2005). This applies in particular to digital cities, where networks of data and computers represent networks of people and buildings. In this setting, requiring each user to have available all resources locally is infeasible most of the time. An established solution is to design the required system as a distributed system that consists of a collection of autonomous computers, connected through a network, which enables computers to share the resources of the system, so that users perceive the system as a single, integrated computing facility (Coulouris, Dollimore, & Kindberg, 2005).

At the same time, the insufficient interoperability between geovisualization systems and components has been identified as a major barrier for progress in the domain. In the geospatial domain, this barrier was identified more than a decade ago (Bisht, 1998) and has lead to extensive work on standardization proposals by the Open Geospatial Consortium (OGC, www.opengeospatial.org). In the geovisualization domain, overcoming the existing limits in interoperability between systems and components was identified as a next major challenge (Andrienko et al., 2005; Gahegan, 2005; MacEachren & Kraak, 2001). The goal of interoperability is supported from the perspective of the domain of 3D computer graphics as a provider of enabling technology for 3D geovirtual environments (3DGeoVEs). Because of the implementation complexity of advanced 3D computer graphics functionality, providing functionality as interoperable, reusable, ready-to-use geovisualization components supports bridging the technology gap between geovisualization systems and 3D computer graphics and raises the level of abstraction at which geovisualization systems can be built (Döllner, 2005).
For the design and implementation of interoperable, distributed systems in the geospatial domain, the application of the service-oriented computing (SOC) and service-oriented architecture (SOA) paradigms and standardization proposals by the OGC are commonly proposed. SOC and SOA promote the idea of assembling application components into a network of services that can be loosely coupled to create flexible, dynamic business processes and agile applications that span organizations and computing platforms (Erl, 2005; Hildebrandt, Holschke, Offermann, & Steffens, 2009; Papazoglou, Traverso, Dустdar, & Leymann, 2007). Basing a SOA on open standards to improve interoperability between services can improve its effectiveness considerably.

In the literature, proposals already exist for designing visualization systems as distributed systems (e.g., Brodlie, Duce, Gallop, Walton, & Wood, 2004; Duce et al., 1998), distributed systems based on SOA (e.g., Charters, Holliman, & Munro, 2004; Shalf & Bethel, 2003; Wang, Brodlie, Handley, & Wood, 2008), or distributed systems based on SOA and OGC standards (e.g., Hildebrandt & Döllner, 2009; Schilling, Neubauer, & Zipf, 2009). In this paper, we intend to contribute to this line of work by presenting a discussion on the application of SOA and standards to the life cycle of 3D visualization systems based on a unified architectural framework, a case study, and a discussion of the potentials and challenges of the approach. The discussion is based on a literature study and our own experiences. We place a particular focus on 3DGeoVEs and virtual 3D city models as elements for digital cities. Furthermore, we focus on 3D rendering of primarily static CAD models with real-time navigation using six degrees of freedom.

The remainder of this paper is structured as follows. In Section 2, we briefly introduce fundamentals of the SOA paradigm, identify generic requirements of 3D visualization systems, and present an architectural framework that relates SOA concepts, geovisualization concepts, and standardization proposals by the OGC in a common frame of reference. In Section 3, as a case study, we present an example SOA for exploring massive, static, virtual 3D city models through the Internet. The service-based, image-based provisioning of and interaction with 3DGeoVEs represent a specific section of the case study that is presented in more detail. In Section 4, we summarize the potential and challenges driven by SOA and standardization, and the more detailed potential and challenges of the case study. Finally, Section 5 concludes the paper.

2. Fundamentals

2.1. Service-oriented architecture

The SOC paradigm promotes the idea of assembling application components into a network of services that can be loosely coupled to create flexible, dynamic business processes and agile applications that span organizations and computing platforms (Papazoglou et al., 2007). The term service-oriented architecture (SOA) denotes both an architectural concept and style that adheres to the SOC paradigm and architectures that are designed following that architectural concept. A service can be defined as a distinct part of the functionality that is provided by an entity through interfaces (ISO, 2005). An interface is a named set of operations that characterize the behavior of an entity (ISO, 2005). When interacting with other services, a service assumes one of three fundamental roles (Erl, 2005). A service provider offers a specific service and publishes meta data about the service to the service registry. A service registry maintains a directory of all registered services. A service consumer performs service discovery operations on the service registry to find a required service and if successful accesses the service provider for the desired service. The service registry facilitates location transparent invocation of services over a network, loose coupling between the service provider and consumer, and is essential for governance and portfolio management activities. Services can be aggregated into a single composite service to collectively automate a particular task or process (Erl, 2005; Papazoglou et al., 2007). In the geospatial domain, a service chain is a specific composite service defined as a sequence of services where for each adjacent pair of services, occurrence of the first action is necessary for the occurrence of the second action (ISO, 2005). To facilitate flexibility and maintainability of a SOA, the fundamental design principle of separation of concerns (Dijkstra, 1982) is applied to categorize services. A common set of categories consists of the categories data, functionality, process, and interaction (Erl, 2005; Hildebrandt et al., 2009) each defining a layer in a distributed layered software architecture (Fig. 1). Services must comply with specific design principles and guidelines to support the strategic goals associated with a SOA (Erl, 2005; Hildebrandt et al., 2009; ISO, 2005; Krafzig, Banke, & Slama, 2004). Design principles found in the literature postulate that services should be, e.g., loosely coupled, coarse-grained, stateless, autonomous, discoverable, composable, and represent domain concepts, activities, and processes. Though SOA is a technology independent concept, web service standards proposed by the W3C (e.g., SOAP, WSDL, UDDI, WS-BPEL, WS-.--) currently constitute the most promising and common base technology (Papazoglou et al., 2007).

The SOA paradigm originated from the domain of enterprise information systems and there it has been researched and applied to a large extent (Erl, 2005; Krafzig et al., 2004). One major goal when applying SOA to the design of distributed systems is not different in the geovisualization domain: supporting the effective and efficient life cycle of flexible, evolvable, interoperable distributed systems. For this, SOA facilitates the creation of a unified architectural view of the distributed system landscape at a high level of abstraction, i.e., the level of the application domain, to bridge the gap between application domain concepts and information technology (Hildebrandt & Döllner, 2009; Krafzig et al., 2004).

2.2. Generic requirements for 3D geovisualization systems

In this section, we identify generic requirements for 3D geovisualization systems. Subsequently, we investigate how applying the SOA paradigm and standards affects developing, operating, disseminating, maintaining, and evolving (system life cycle) 3D geovisualization systems. We examine the potential and the challenges for meeting a series of basic requirements when applying SOA and standards. Whilst derived from existing literature the generic list of requirements affected by the decision to design a system using SOA and associated standards is based upon our own perspective and is thus inherently debatable. We do not claim that the requirements are exhaustive and neither are they fully independent.

The first group of requirements is relevant but not specific to 3D geovisualization systems. Support for integration (R1) is required to connect computer systems effectively and efficiently on different levels of abstraction such as data, functionality, process, visualization, interaction, and system. On these levels, it should handle heterogeneity and is a precondition for reuse (Brodlie et al., 2007; Rhyme & MacEachren, 2004; Roberts, 2007). Interoperability (R2) increases the effectiveness and efficiency of the integration on the different levels and can be improved by applying standards (Andrienko et al., 2005; Gahegan, 2005; MacEachren & Kraak, 2001; Roberts, 2007; Thomas & Cook, 2005). Support for reducing and controlling complexity (R3) of the system life cycle, its artifacts and interactions with its artifacts is required to reduce costs and to raise the level of what is achievable with given resources (Brodlie et al., 2007; Roberts, 2007; Shalf & Bethel, 2003). Support for platform independency (R4) comprises the relative independency
of a system solution from software and hardware platforms on different levels of abstraction and adaptive and moderate use of platform resources. It can improve dissemination and reduces costs (Laramee & Kosara, 2007; MacEachren et al., 2004; Roberts, 2007; Shalf & Bethel, 2003). Support for dissemination (R5) of a system improves the ease and scope of its distribution, its use, success, and impact (Andrienko et al., 2005; Gahegan, 2005; Laramee & Kosara, 2007; Thomas & Cook, 2005). Support for further non-functional features (R6) particularly comprises support for performance, scalability, extensibility, and reuse, but also for security, dependability, robustness, availability, transparency, and so forth (Brodie et al., 2007; Roberts, 2007; Shalf & Bethel, 2003). Economic requirements (R7) refer to the economy of the system life cycle and the commercial exploitation of its products on the market. Cost-effectiveness and market opportunities can improve the feasibility of system developments, dissemination, and progress in general (Krafitz et al., 2004; Muntz et al., 2003; Nöllenburg, 2007) and massive amounts of geodata (R18) (Hildebrandt & Döllner, 2009). Support for effective, high quality visual representations and interaction techniques (Dykes, MacEachren, & Kraak, 2005; Gahegan, 2005; Roberts, 2007). Support for the specifics of geovisualization processes (R12) (e.g., dynamics of the process, relation to science process) are required to improve the effectiveness and efficiency of systems (Andrienko et al., 2005; Dykes, 2005; Gahegan, 2005). A high degree of interactivity (R13) is a key defining characteristic and critical requirement for geovisualization systems and should be efficient and effective (Dykes et al., 2005; MacEachren & Kraak, 2001; Nöllenburg, 2007; Roberts, 2007). Support for coordinated multiple views (CMV, R14) is a key requirement for exploratory visualization and, thus, for respective geovisualization systems (Boukhelifa, Roberts, & Rodgers, 2003; Roberts, 2007). Providing effective, high quality visual representations (R15) improves the effectiveness of a geovisualization system and is facilitated by advanced, complex, innovative visualization algorithms and in certain cases massive amounts of data (e.g., for virtual 3D city models), for both realistic and abstract views (Döllner, 2005; Hildebrandt & Döllner, 2009). Support for styling visual representations (R16) allows control over what (e.g., filtering of features) and how to portray (e.g., mapping of features to geometries and visual attributes) and is essential for interaction and generating different visualizations from the same base data (Lapp, 2007; Neubauer & Zifp, 2009; Yi, Kang, Stasko, & Jacko, 2007). Support for computational analysis and processing (R17) complements visualization and interaction and is required for different uses such as data preprocessing, cleaning, generalization, aggregation, and derivation of statistics (MacEachren & Kraak, 2001; Roberts, 2007; Shalf & Bethel, 2003). Typically, in real world applications, systems are required to facilitate processing, visualizing and interacting with massive amounts of geodata (R18) (Hildebrandt & Döllner, 2009; Muntz et al., 2003; Nöllenburg, 2007) and dynamic geodata (R19) (MacEachren & Kraak, 2001; Thomas & Cook, 2005).

In the following sections, we make explicit reference to each introduced requirement via its respective code when the discussion touches the requirement. If feasible, we estimate whether when applying SOA or standards the discussed aspects have the potential (“+”) or pose a challenge (“−”) to support meeting the requirements.

2.3. Architectural framework and standards

In this section, we present an architectural framework that organizes and relates SOA concepts, geovisualization concepts, and OGC standards in a common conceptual frame of reference to allow for a differentiated discussion of the potential and challenges in the following sections (Figs. 1–3).

In Fig. 1 we relate the SOA layers, the visualization pipeline extended with user controlled interaction, and the model-view-control pattern. From these concepts, we derive a decomposition of a basic, generic 3D geovisualization system into services and the data flow between the services. The visualization pipeline (Haber & McNab, 1990) is a well-established concept for separating the concerns of the process of generating visual representations from data in three stages. The data is filtered into enhanced data, then mapped into data representing geometry and visual attributes, and finally rendered into a digital 2D image. The display service extends the pipeline and displays the image typically in a GUI window to a user. It accepts input from the user through input devices. The controller service transforms input events (e.g., mouse click coordinates) into commands for updating the visualization. The view process acts as a facade and mediator (Gamm, Helm, Johnson, & Vlissides, 1995) to the visualization pipeline and transforms these commands into a sequence of service calls and corresponding parameters for executing the pipeline. The
The presented framework can effectively be used to describe, design, and analyze the separation of concerns and data flow when visualization functionality is decomposed into services. It represents an ideal-typical decomposition. When implementing systems, several adjacent services can be combined into a single service and vice versa. Services can span multiple layers (Erl, 2005). It should even be possible to design working systems in which the presented separation of concerns is not recognizable. However, since the framework is based on proven concepts, we assume that following the framework and its concepts is advantageous.

In Fig. 3, we supplement the framework with proposals by the OGC for services and data models that are directly relevant in the context of geovisualization. We assign the proposals to the respective layers and previously identified services to the proposals if applicable. Fundamental standards for data models and encodings are the Geography Markup Language (GML) for general geospatial features, CityGML (Gröger, Kolbe, Czerwinski, & Nagel, 2008) as an application schema of GML for 3D city models, and the Styled Layer Descriptor (SLD) (Lupp, 2007; Neubauer & Zipf, 2009) for user-defined styling of 2D and 3D visual representations. The data layer contains services that manage and grant access to geodata such as the Web Coverage Service (WCS), Web Map Service (WMS), and Web Feature Service (WFS). Moreover, it can contain data resources such as SLD resources, textures or other proprietary data that are not encapsulated by OGC services but by proprietary service interfaces. For the presentation of information to humans, the OGC proposes stateless portrayal services. For 2D portrayal, the WMS is proposed as an approved standard, whereas for 3D portrayal the Web 3D Service (W3DS) (Schilling & Kolbe, 2010) and the Web View Service (WVS) (Hagedorn, Hildebrandt, & Döllner, 2009, 2010) are proposed as different approaches that are both still in the early stages of the standardization process. The major difference in the current proposals for the 3D portrayal services is the representation that they generate and the extent to which they implement the three visualization stages. The W3DS delivers scene graphs that can be rendered by a client, whereas the WVS delivers rendered images of projected views that are ready for display. An analysis of the respective strengths and weaknesses of 3D portrayal services can be found in Hildebrandt and Döllner (2009).

Web Processing Services (WPS) (Schut, 2007) can encapsulate additional functionality that is required for a specific visualization process in the filtering, mapping, and rendering stages in a generic way. In the context of portrayal, services in the process layer are responsible for executing a portrayal or view process by orchestrating services on the functionality and data layer. Services on the interaction layer offer interfaces to human users or other software components. For both the process and the interaction layer, there are no dedicated service proposals by the OGC yet. Nevertheless, the generic WPS can be used to encapsulate process services.

2.4. Coordinated multiple views and collaboration

In this section, we show how in principle CMV and CV (R10, R14) can be designed architecturally in a SOA using the introduced framework.

CMV is a key technique for exploratory visualization where data is presented in multiple views to a user and operations on the views are coordinated (Roberts, 2007). We describe an adoption of an existing model for CMV (Boukhelifa et al., 2003) to SOA (Fig. 4). This existing model extends the interactive visualization pipeline and can be used to describe a wide range of instances of CMV (Boukhelifa et al., 2003). Each display passes user input events to its controller. A controller passes the input events directly to the view process service or to a coordination service (CS) if the event is relevant for coordination. A CS represents a number of coordinations. Each of them coordinates a set of entities (e.g., parameters of display, pipeline stages, data) that exist in the pipelines of different displays. Depending on the input event and the
originating controller, the CS identifies a specific coordination and transforms input events into modifications of abstract parameters for that coordination. For each pipeline, the CS transforms abstract to pipeline specific view parameters via a translation function and calls its view process. The view process transforms the view parameters into a sequence of service calls with corresponding parameters. The pipelines of different displays can share data and stages (fan-in/out) but this is not essential. A characteristic of CMV is that at least two displays are presented to one user and that at least one parameter change in one extended pipeline affects the other extended pipeline. We give an example of a service-oriented CMV system in Section 4.1.2.

The differences of this adoption to the original model are minor: The state of a pipeline is moved from its stages to the view process and data services (in case of the persistent passing pattern) to keep the stages stateless. The architecture is strictly layered to increase loose coupling: A stage does not know other stages, stages do not know their view process, and the view process does not know its CS (R1 +, R3 +, R13 −). The translation function is defined in the view process and not in the stages to keep the stages free from the knowledge of which processes they are integrated with and to improve separation of concerns and reusability (R3 +, R6 +). To make services more coarse-grained, multiple coordinations are combined within one CS and one view process encapsulates multiple different manipulations of the state of the pipeline (R3 +). The distributed system design can cause time lags between the generation of a user input and the display of the modified visualization that were considered negligible in the original model (R13 −).

Several characteristics of a service determine how suitable it is for use in a CMV system. A service can be used more flexibly in a CMV system, if the service offers explicit, extensive and preferably standardized means for controlling it (R1 +, R2 +, R3 +, R4 +). Offering functionality as one service instead of decomposing it into a number of services, can improve performance and interactivity, but degrades potentials for reuse, distribution between places, and possibilities when combining services as building blocks into new pipelines (R1, R6, R9, R13, R14). Coordination between different services is only possible, if a third service can map between the concepts of the services that are to be coordinated (e.g., spatial reference system, feature IDs). This mapping is eased if the services use common or standardized formalizations for concepts (R1 +, R2 +, R4 +). Support for CV is required to allow a group of people to work together on tasks involving geovisualization (R10). We describe an adoption of an existing model for distributed, synchronous (different place, same time) CV (Brodlie et al., 2004) to SOA. This existing model extends the interactive visualization pipeline and can be used to describe a wide range of instances of CV (Brodlie et al., 2004; Duce et al., 1998). Synchronous CV can be characterized by the selective sharing of data, functionality (e.g., stages), or control between users (Duce et al., 1998). A distributed CV system can be modeled as a collection of pipelines. The pipelines can be complete or partial and can share data and stages. Each participating user receives the output of at least one pipeline and can control the collection of pipelines at specific stages. The control information for a stage in one pipeline can be used for a corresponding stage in a different pipeline to synchronize the stages. The originally proposed models for CMV (Boukhelifa et al., 2003) and for CV (Brodlie et al., 2004) share many similarities. In both models at least two displays are present (for one user in CMV and two users in CV), and data, functionality and control can be shared. Sharing is not essential in CMV. However, at least one parameter change in one pipeline must result in a transformed parameter change in a different pipeline. Sharing of data, functionality, or control is essential in CV. When sharing control, parameter changes are typically shared untransformed between pipelines. Interestingly, because of the similarities, the adoption of the CMV to SAO (Fig. 4) can be used to model CV as well. In the case of CV, the sharing of control can be seen as just another instance of coordination. The user input events from different users all pass through the central CS. The CS encapsulates the knowledge of what event from what user affects what view process in what way.

3. Case study

To illustrate the introduced concepts and the discussion of the potential and challenges in the following sections, we present a case study and example of a fundamental SOA of a distributed system that implements a specific configurable, service-based geovisualization pipeline (Hildebrandt, 2009) for 3DGeoVE (Fig. 5). The main purpose of the example system is to enable a user with a lightweight, web-based client to explore interactively a massive, static, virtual 3D city model through the Internet based on server-side 3D rendering. To facilitate this purpose, the system offers functionality for importing and preprocessing the input geodata that comprises the virtual 3D city model into a representation that is ready to be rendered.

3.1. Architecture, services, and user roles

In the filtering stage, the purpose of the WPS-based City Factory Importer service is to transform raw geodata into an inte-
grated, semantic representation of a virtual 3D city model. For this, the service first imports raw geodata (e.g., shape files from the cadastre, digital terrain models, and aerial photographs) from potentially different external organizations. Then, it transforms the raw data into a model adhering to the CityGML data model. The CityGML representation is then transferred to and stored by a WFS-based City Storage service. The texture images are stored separately on a file server. The City Factory Importer service is implemented in C#/C++ based on the Virtual Rendering System (VRS, www.vrs3d.org). The VRS is used for this and other services (see below) since it already provides considerable functionality for processing, rendering, importing and exporting geometry, texture and scene graphs in different data formats. For the importer, the rendering functionality of the VRS is not used and extensions are implemented for accepting WPS requests, reading shape files, and exporting CityGML. The services for Raw Data and textures are implemented as file servers. The City Storage service is implemented using the Java-based WFS provided by deegree (www.deegree.org) and a 3D geo database for CityGML (Stadler, Nagel, König, & Kolbe, 2008) based on Oracle 10G R2.

In the mapping stage, the purpose of the WPS-based City Preprocess service is to transform the standards-based CityGML and texture representations into proprietary, internal representations optimized for 3D rendering. For this, the CityGML and texture representations are retrieved from the respective services, processed and stored by a Proprietary Storage service in proprietary data models. The purpose of the 3D Renderer service is to generate 2D images of projected views of 3DGeoVE (e.g., virtual 3D city models). For this, the service retrieves data from the Proprietary Storage, accepts requests from service consumers containing virtual camera specifications and SLD styling specifications (R16 +), and returns 2D images. Additionally, the 3D Renderer service accesses the City Storage service to satisfy clients requesting additional properties of portrayed features. In this example, we annotated the employed 3D Renderer service with the capabilities code “WVS-I-SLD” (Hildebrandt & Döllner, 2009). This code denotes that the service implements the WVS interface, is integrated with the geodata that it can portray (“I”), and supports the user-defined styling for integrated and externally referenced geodata (“SLD”). Since the 3D Renderer service supports user-defined styling, it performs in part mapping functionality besides the rendering functionality. The City Preprocess and 3D Renderer services are implemented using the Java-based WFS provided by deegree (www.deegree.org) and a 3D geo database for CityGML (Stadler, Nagel, König, & Kolbe, 2008) based on Oracle 10G R2.

The architecture supports two human user roles and their tasks: Administrator and Explorer. An Administrator uses the Admin Web-Client for content management (CM) tasks. The example architecture only supports the basic task of importing raw geodata into the system and preprocessing it into a representation ready to be rendered. Based on inputs from the administrator, the Admin Web-Client invokes the WPS-based CMS Process service. This service uses the City Factory and City Preprocess services for the implementation of its process and is capable of handling occurrences of long download durations, network failures, invalid or corrupt data and so on (R8 +). The Admin Web-Client is implemented as a simple HTML page. The CMS Process is implemented using the Java-based WPS provided by deegree. An Explorer uses the 3D Viewer Web-Client for interactively exploring the virtual 3D city model. In a short, closed loop, the 3D Viewer Web-Client requests images from the 3D Renderer service, displays the images, and requests new images based on the users input. The 3D Viewer Web-Client is implemented as a Java applet embedded in a HTML page. For hardware-accelerated, client-side 3D rendering, a Java binding for OpenGL (JOGL) is used. 3.2. Service-based, image-based provisioning and interaction

In this section, we introduce an approach for the service-based, image-based provisioning of and interaction with visual representations of remote 3DGeoVE. We focus on a specific part of the example architecture presented in Section 3.1 – the 3D Renderer service and the 3D Viewer Web-Client – and discuss it in more detail. The introduced approach is the subject of our ongoing research and we present preliminary results from our working, prototypical implementation (www.webviewservice.org, see the Supplementary video for a demonstration). We present this approach to give an example for service-orientation applied to 3D portrayal and interaction with 3DGeoVE, to illustrate further the general discussion of potential and challenges, and to allow for a more detailed discussion in a narrowed area. Furthermore, the introduced approach offers specific characteristics and potential that situates it between approaches utilizing either the W3DS or the original, purely image-based WPVS respectively (Hildebrandt & Döllner, 2009).

We propose the WVS (Hagedorn et al., 2009, Hagedorn, Hildebrandt, & Döllner, 2010) as a 3D portrayal service for generating 2D images of projected views of 3DGeoVE. The WVS overcomes restricted visualization and interaction capabilities of preceding proposals such as the Web Terrain Service (WTS) and its successor the Web Perspective View Service (WPVS, OGC-internal draft specification) (R13 +, R15 +). The OGC has approved the WVS specification as discussion paper. As a major functional extension, the WVS provides additional thematic information layers for generated images. A WVS client can retrieve and exploit thematic information layers storing information such as color, depth, object ID, surface normal and mask encoded in standard 2D image formats for a specified virtual camera location, orientation and perspective or orthographic projection (Fig. 6). This concept is based on the G-buffer (Saito & Takahashi, 1990) concept from 3D computer graphics.

To enable a user to explore and interact with a remote, massive virtual 3D city model stored and rendered by the WVS through the Internet, we developed a first concept and prototype implementation for a lightweight, web-based client application that utilizes the WVS (Fig. 7) (R4 +, R5 +). The underlying concept of the WVS client uses ideas from the domain of image-based rendering (Shum, Chan, & Kang, 2007) and applies the NPW pattern (Section 2.3). The client requests cube maps for user specified virtual camera locations in several information dimensions (color, depth, object ID) from the WVS (Fig. 8). The cube maps are transferred as image sequences to the client. The client interprets the images as a description of the remote 3DGeoVE consisting of 3D surface patches with attributes including 3D position, color, and object ID. From the surface patches, 3D meshes are constructed. The meshes represent the local, partial reconstruction of the remote 3DGeoVE (nested mapping stage). To support navigation and interaction, novel views can be rendered of the local 3D reconstruction from arbitrary virtual camera viewpoints with low latency (nested rendering stage). The client application supports features including

![Fig. 6. Examples of image layers provided by WVS: (a) color layer, (b) depth layer, (c) object ID layer.](image-url)
common direct navigation techniques (e.g., rotate virtual camera, pan, zoom, and orbit), field of view-based zoom with selective refinement of the cube map, selection and highlighting of object features (e.g., buildings) within the 3DGeoVE, retrieving and displaying additional thematic information for object features from the WVS, and measuring the Euclidean distance between arbitrary spatial positions (R13 +).

4. Potential and challenges

In this section, based on a literature study and our own experiences, we summarize identified potential and challenges, benefits and disadvantages that arise when building standards-based, distributed systems according to the SOA paradigm. We focus on systems built for 3D geovisualization, with a particular focus on 3DGeoVE and virtual 3D city models. We further focus on 3D rendering of primarily static CAD models with real-time navigation using six degrees of freedom. We organize the summary of identified aspects into three groups. The first group discusses more applied aspects in relation to the application presented above.

The second group constitutes aspects driven by the application of SOA and is further subdivided into the subsections service fundamentals, service composition, interaction services, overarching aspects, and performance. The third group summarizes aspects driven by standardization.

4.1. Discussion of the case study

4.1.1. Architecture

The presented example architecture is an instance of distributed CV (Section 2.4) where two user roles share data and services. Since no control is shared, there is no need for a coordination service in this case (R10 +). The persistent passing pattern is applied between the City Factory Importer, City Preprocess, and 3D Renderer services. Because the data passed between these services is expected to be voluminous and its processing is expected to be time-consuming (Hildebrandt & Döllner, 2009), the pattern is used to improve the robustness, availability, performance, and reusability of the system (R6 +). In general, a data model is designed for a certain purpose and, thus, can only be applied successfully within certain limits. The primary purpose of the stored CityGML representation and textures is to function as a standardized interchange format that can be reused for a variety of different tasks even beyond visualization. In contrast, the purpose of the proprietary data derived from that data is to act specifically and efficiently as input for 3D rendering by a specific 3D Renderer service. In this context, important requirements for 3D rendering and its supporting data representations are (R13, R15, R16, R18, R19). For 3D rendering with these requirements, no “standardized” solutions exist. It is still the subject of ongoing research (Döllner, 2005). Hence, generally, standard data models and encodings (such as CityGML, X3D, KML or COLLADA) cannot be used as efficient input for specific 3D rendering solutions. For example, a wide range of different approaches exist for 3D rendering of massive models that do not fit into main memory (e.g., out-of-core techniques), consist of more geometry than can be interactively rendered (e.g., level-of-detail techniques), and are globally illuminated (e.g., ambient occlusion techniques). In the example architecture, the proprietary data is designed for specific out-of-core and global illumination techniques implemented by the 3D renderer. Similarly, the expressiveness of the current proposal for 3D styling by the OGC (SLD Neubauer & Zipf, 2009, R16) seems limited in comparison to what appears to be beneficial for advanced, effective visualizations (R15 +). For instance, it is not possible to control lighting, advanced material properties, or environmental effects.

The capability of storing and provisioning a virtual 3D city model in a semantically rich, integrated standardized representation such as CityGML represents a significant value. Reusing an existing solution for modeling virtual 3D cities allows developers to start building systems on a higher level of abstraction (R8 +). However, the effort for developing a customized model is replaced by the effort to incorporate the extensive CityGML standard (R7). The model provided by the City Storage server can be reused as a base model for the integration of various georeferenced data in various application areas (R1 +, R2 +, R6 +). The 3D Renderer service can be reused by a multitude of different clients that can differ in hardware platforms (e.g., mobile devices or desktop computers), software platforms (e.g., browser-based or stand-alone application, programming language, operating systems), degrees of provided interactivity (e.g., static images, interactive navigation in 3D space) and application areas and tasks (R1 +, R2 +). The 3D Viewer can re-use and integrate arbitrary WVS services that can differ in provided content, non-functional characteristics, and vendor-specific functionality (R1 +, R2 +). For the remaining services, the potential for integration, interoperability, and reuse is considerably lower if they are not based on standards (R1 –, R2 –, R6 –). Even if

Fig. 7. Screenshot of the web-based WVS client application running inside a web browser (background). An object feature that was selected by the user is highlighted and annotated with additional information. Screenshot of the WVS client application running on the Smartphone Apple iPhone (foreground).

Fig. 8. Cube maps as primitives for image-based modeling and rendering. (a) Six faces of a cube map as requested by the client from the WVS service. (b) The cube map assembled and rendered by the client. When the virtual camera is placed in the center of the cube and the virtual camera is rotated, the user experiences a perfect visual illusion of being in the 3DGeoVE.
service interfaces are not standardized, flexibility is improved if services interchange data in standardized models and encodings. For example, the same City Factory Importer service can be used in conjunction with different City Preprocess services and vice versa. The use of web services facilitates the integration of services implemented and operating on different platforms (i.e., programming languages and operating systems) \( (R_1 +, R_2 +, R_4 +) \).

As an intermediate conclusion, we argue that applying standards and the architectural framework presented in Section 2.3 promotes a system design with a clear separation of concerns and significant potential for flexibility and efficiency when building and evolving systems.

4.1.2. Image-based provisioning and interaction

We describe a thin web-based WVS client (Hagedorn et al., 2010, Figs. 9 and 10), to demonstrate how the image-based approach can be applied to implement mashups that realize CMV, and integrate geodata and services from different organizations on the interaction layer \( (R_1 +, R_5 +, R_8 +, R_9 +, R_12 +, R_14 +) \). The JavaScript-based client presents two displays to the user: The first displays 2D maps through the Google Maps JavaScript API (conceptually using the NPVP pattern internally, Section 2.3), the second displays 3D perspective images from a WVS. Both displays show models of the same spatial region of the world, but the underlying data and functionality for generating images are completely disjoint. As a simple navigation technique, either display can be used to first select the location of the virtual camera and then its target with two separate mouse clicks. The mouse clicks are passed from either display to the CS. The CS passes the information about a newly set camera location or target to both view processes. The view process for the 2D display sets markers accordingly. The process for the 3D display shows an arrow connecting the camera location and target and retrieves a new perspective image from the WVS as soon as a new camera is fully specified. Coordinating locations between both displays is feasible because both use the same standardized coordinate reference system. Other types of coordination may be limited. For instance, currently, synchronizing the highlighting of features is not feasible because the 2D display does not allow individual features to be referenced \( (R_14 -) \). To illustrate further and evaluate how a service-oriented 3D geovisualization system can provide interactivity, we evaluate how the WVS client presented in Section 3.2 supports specific interaction categories \( (R_13 +) \). The used categories are based on the notion of user intent and are taken from Yi et al. (2007). As an interaction technique in the category Select (mark something as interesting), clicking on a feature highlights the feature (implemented by coloring all pixels with the same object ID as the clicked on pixel). As Explore techniques (show me something else), the client allows rotating the camera around itself, moving the camera by panning, and moving it by selecting a feature of interest that is then brought into focus via a smooth camera animation (implemented by rendering the local 3D reconstruction of the 3DGeoVE in a local NPVP with low latency). As Reconfigure technique (show me a different arrangement), the client allows rotating the camera around a selected feature in the 3DGeoVE (implementation similar to previous category). For Encode (show me a different representation), the client can specify the styling in a SLD document and retrieve projected images with the styling applied from a WVS. As Abstract/Elaborate techniques (show me more or less detail), the client offers (geometric) zooming and tool-tips for displaying additional information about features (implemented by modifying the field-of-view of the virtual camera and by retrieving additional information encoded in GML for a feature identified by its object ID). To Filter (show me something conditionally) the data set being presented, the client can specify in a SLD document conditionally what features to include and retrieve images from WVS with the filtering applied. As a Connect technique (show me related items), the client can be combined with different displays in a CMV system as described in the previous paragraph.

The presented approach has the potential to efficiently generate visual representations of 3DGeoVE consisting of massive amounts of geodata \( (R_{18} +) \). The required geodata is stored in a proprietary representation optimized for rendering and is retrieved by the WVS only once. Since the implementation details are hidden, the service can make use of parallelization to improve the rendering performance. This service is an instance of the performance-improving pattern described in Section 4.5 that processes the data with fixed functionality in the same place where it is stored. The service consumer receives the processing results as compressed 2D images \( (R_6 +, R_{13} +) \). The service can provide high quality, advanced, and innovative visual representations on a high interoperability level (see Section 4.7) \( (R_2 +, R_{15} +) \). This is due to the fact that implementation details of the service are hidden, the service supports a standardized interface and standardized data models and encodings for input and output, the service controls a significant portion of the geovisualization pipeline (i.e., the rendering and a part of the mapping stage), and the rendering is operated in a controlled server environment with potentially significant hardware resources. Since the service transfers 2D images instead of complex 3D representations such as the W3DS, the assets and intellectual property of the service provider are better secured \( (R_6 +) \). The WVS client is a lightweight application running inside a web browser or on the Apple iPhone. The client application running inside the web browser is cross-platform and can be used in composite applications and mashups. Since remote services store and render geodata, the client puts low requirements on the end user resources \( (R_1 +, R_4 +, R_5 +) \). The WVS client offers a high degree of interactivity, since additional information besides color is available for the visual representations, the 3DGeoVE is locally reconstructed, and novel 3D
views can be generated from arbitrary virtual camera points (R13 +). As observed in Altmaier and Kolbe (2003), in general, the integration of geodata on the image level is not possible. However, since the WVS is capable of providing additional depth information per pixel, geodata can be integrated on the image level (R1 +).

The remaining challenges and areas of future work include providing facilities to manipulate (i.e., modify, delete) data that resides on the WVS (R19 –), providing facilities to control what features are to be portrayed in what way (styling) (R16 –), developing a client-side concept for efficient and effective navigation in virtual 3D city models (R13 –), improving the rendering performance and image compression effectiveness of the service (R6 –), and to explore incorporating further modalities to enable immersive 3DGeoVE.

4.2. Service fundamentals

In a SOA, potentially remote services provide resources such as data, functionality, processes, and interaction capabilities over a network.

Resources dispersed within and between organizations all over the globe can be uniformly provided and utilized as services (R1 +, R2 +, R3 +). Existing assets can be leveraged and encapsulated as services to protect former investments (R7 +). SOA promotes interface orientation, encapsulation, and hiding of implementation details (Hildebrandt et al., 2009). This effectively facilitates hiding of implementation complexities (R3 +) and loose coupling of a service provider from a service consumer, each of which can be implemented on different hardware and software platforms (e.g., programming language, operating system) (R4 +). Though SOAs can be implemented with object-oriented (e.g., CORBA) or component-based (e.g., JEE, .NET) paradigms and technologies, the web service technology represents the first widely adopted, open standards-based technology for distributed systems that offers implementation transparency (R4 +). Hiding implementation details allows services to be transparently implemented and to operate in a grid (Foster & Kesselman, 2003) or cloud computing (Armbust et al., 2009) infrastructure to improve the scalability or performance of a service (R6 +, R17 +, R18 +). Decentralized, shared resources offer the potential to support different-place collaboration (Brodlie et al., 2005), and up-to-dateness and consistency of functionality and data (R1 +, R9 +, R10 +, R19 +). The postulated coarse granularity and representation of geovisualization domain (instead of technical) concepts, activities and processes for service interfaces enhance their potential for reuse, the comprehensibility of a SOA, and the level of abstraction for building a system (R6 +, R3 +, R8 +). From a 3D computer graphics perspective, services can help bridge the technology gap between geovisualization and computer graphics (Döllner, 2005). Complex computer graphics and geovisualization concepts, techniques and metaphors can be encapsulated by reusable, competing services (R15 +). This raises the level of abstraction at which geovisualization systems can be built, touching a key requirement of the geovisualization community (Döllner, 2005; MacEachren & Kraak, 2001) (R8 +).

Designing a system as a distributed system possibly spanning multiple organizations over the Internet can create significant additional complexity regarding technical, operational, organizational, social, and geovisualization aspects (R3 –, R9 –). Distribution is better avoided if it does not add value and is not required to meet respective requirements. Introducing distribution can degrade a system’s non-functional characteristics such as dependability, transparency, and performance (see Section 4.5) (R6 –). Distribution exposes a system to additional security risks (R6 –). Services implemented with open standards such as the web service standards are much more open to other services and applications and, thus, their vulnerability increases.

4.3. Service composition

Services can be aggregated into composite services or chained to automate collectively particular tasks or processes in the geovisualization pipeline.

SOA-based systems have the potential to react to changing requirements in an agile and efficient way. New and modified processes can easily be created by combining existing and newly created services (R1 +, R12 +). In a SOA, resources are functionally decomposed into services using a common base technology (e.g., web services) and a mechanism for composing services into higher-level services is provided. This facilitates the integration of functionality and data on the technical and syntactic level across multiple independent organizations and systems, and the reuse of resources even in processes they were not explicitly designed for (R1 +, R6 +, R8 +). In the domain of enterprise systems, for modeling and executing processes composed of individual services specific textual and visual languages (e.g., BPEL, BPMN) and tools, e.g., Microsoft BizTalk, SAP NetWeaver Process Engine) have been developed. Specialized languages and tools can be used to model and execute geovisualization processes more effectively and efficiently than with general approaches. Enriching service meta data and service registries with semantic information offers the potential to automatically discover and orchestrate services for specific tasks (Yue, Di, Yang, Yu, & Zhao, 2007) (R1 +). Providing an infrastructure for discovering, utilizing, and composing services can open new market opportunities, and increase the practical impact of research results (Gahegan, 2005) (R7 +, R5 +). Presently, limiting factors for the effective and efficient use of current geovisualization systems include that current systems do not support geovisualization processes sufficiently. Typically, the processes are specific and dynamic in nature and require diversity on different levels necessitating a number of different tools and “glue” (Andrienko et al., 2005; Dykes, 2005; Gahegan, 2005). Offering, composing, integrating, and reusing services for specific tasks and users from an extensive repository of services can alleviate these limitations (R11 +, R12 +).

A key challenge is to decompose functionally a geovisualization system into composable services while balancing forces that are introduced by the requirements for 3D geovisualization systems as presented in Section 2.2. The proposed architectural framework and the concepts upon which it is based (Section 2.3), and the scientific and industrial processes in the geovisualization domain (Dykes, 2005; Gahegan, 2005) can act as guides in the decomposition. Additionally, there is need to investigate to what extent we are able “to borrow tools” (Andrienko et al., 2005), i.e., to reuse specific or generic services for specific or generic tasks (R1 –, R8 –, R11 –). Integrating services from different organizations poses challenges on the service management must take into account factors such as ownership, responsibility, quality of service, and accounting (R3 –, R7 –). Service compositions have to take into account uncertain and variable characteristics of distributed system resources (e.g., availability, dependability, quality of network, processing, data resources) and client resources (e.g., storage and processing capacity, display technology, user characteristics). Adaptive resource utilization and management are required (Brodlie et al., 2007; Laramee & Kosara, 2007; Shalf & Bethel, 2003) (R4 –, R6 –, R17 –). Regarding the composition of services, research challenges include determining which models and languages facilitate effective and efficient service orchestration in the geovisualization domain (also supporting CV and CMV), investigating whether proposals from the enterprise domain can be applied in the geovisualization domain, and the (semi-) automatic composition of services (R1 –, R10 –, R12 –, R14 –).
4.4. Interaction services

Services on the interaction layer integrate services from the same and lower layers, and offer interfaces to human users or other software components. The SOA paradigm facilitates creating composite applications and mashups that can combine functionality and data from every layer of the layered SOA architecture and that can easily be recomposed and reconfigured to meet different task and user requirements. The traditional application-orientation is replaced by user- and process-orientation facilitating better accounting for user, task, and process requirements (R5 +, R8 +, R11 +, R12 +). By using interaction services, local user devices are freed from the burden of providing all required processing and storage capacity locally, resulting in lightweight clients that can operate, for example, in web browsers and on mobile devices. Interaction services generally allow cross-platform environments such as web browsers and the Java software platform to be used in their implementation, which can ease and increase the scope of the distribution of systems, facilitate maintenance and reduce costs (R3 +, R4 +, R5 +, R7 +). Thus, the SOA paradigm supports platform independent visualization (Laramee & Kosara, 2007), sharing of work and research results (Gahegan, 2005), and collaboration (R4 +, R5 +, R10 +). Two users can share the same functionality and data without the need to provide the same hardware and software.

On the interaction level, there is a need for specific interaction concepts. These concepts must take into account that the decomposition and distribution of functionality and data introduces limitations on the overall performance of the system and the control and availability of specific functionality and data (R3 +, R6 +, R13 +, R12 +, R14 +, R16 +). From the perspective of human computer interaction and usability, it is a challenge to compose and present interaction services that are perceived as seamlessly, logically and visually integrated and consistent in concepts, use and appearance (Andrienko et al., 2005) (R1 –).

4.5. Performance

Requiring a distributed system to facilitate processing, visualizing and interacting with massive amounts of geodata poses significant challenges on the performance of the system (R6 –, R13 –, R18 –). The performance must be sufficient to enable a high degree of interactivity as a critical requirement. Since functionality and data are distributed over the system, data must be transferred over the network at some point in time. Typically, due to the massive size of geodata, it is not feasible to pass all data that is required to generate an output image through the complete pipeline for each frame. The challenge is on the one hand to design the system based on SOA and standards, and on the other hand to provide a system with satisfactory performance. General strategies to approach this challenge include reducing the amount of data transferred over the network, avoiding the execution of stages, and using more hardware resources (Sisneros et al., 2007). For data reduction the following techniques can be considered: efficient, spatial and other multi-dimensional indexes that provide random, localized access to data; compression and streaming techniques; processing large amounts of data with fewer service calls; asynchronous service communication; multiple representations and resolutions (e.g., the results of abstractions, generalizations, level-of-detail processing, aggregations) in order to not transfer more data than is required for a given scale, and processing the data in the same place where it is stored. The last-mentioned technique has two variants: (a) the service provides fixed functionality, (b) code implementing the requested functionality is moved (moving code, Brauner, Foerster, Schaeffer, & Baranski, 2009) to the data for processing, similar to the concept of stored procedures known from the domain of databases. To avoid the execution of stages the following techniques can be considered: caching the results of preceding stages (in memory, local hard disk, remote data service via the persistent passing pattern, maybe combined with pre-fetching), using latency hiding techniques to trade high latency with approximations (e.g., using the NPVP pattern), and limiting the execution of stages to stages with changed input data or parameters (in combination with caching). Finally, as already mentioned, services can be transparently implemented and operating in a grid (Foster & Kesselman, 2003) or cloud computing (Armbrust et al., 2009) infrastructure to improve the scalability or performance of a service (Shalf & Bethel, 2003) (R6 +, R17 +, R18 +). Generally, massive dynamic data are more challenging to visualize than massive static data since most optimization techniques are most effective with static data (R19 –). One approach can be to separate static and dynamic data as much as possible, construct separate, optimized pipelines, and integrate the data again at latter stages. For instance, the case study (Section 3) can be extended with a pipeline for dynamic data (e.g., for movement data of mobile users). Both pipelines can be integrated in the client’s local rendering stage.

4.6. Overarching aspects

In this section, we summarize overarching aspects when introducing a SOA. The application of the SOA paradigm supports building geovisualization systems that yield high structural quality and maintainability (R3 +). The SOA paradigm can be seen as an instrument to control the complexity and heterogeneity of distributed systems and application landscapes on the architectural level by providing a unified view (R3 +). The unified architecture, a unified infrastructure and the reuse of services can lead to reduced costs when developing and maintaining a SOA (R7 +). The principles of encapsulation, implementation hiding, and loose coupling allow for a gentle, evolutionary migration from an existing architecture to a SOA (Hildebrandt et al., 2009; Krafzig et al., 2004) (R3 +).

Introducing a system based on the SOA paradigm can take longer than introducing a centralized or non-SOA-based distributed system, it can require larger upfront investments, and the return on investments can take longer to materialize (R7 –). For developers, adopting a new paradigm and new technologies requires training and rethinking of established practices (R3 –). Within an organization, various political, organizational, financial, and technical forces might oppose the introduction of a SOA (Erl, 2005; Krafzig et al., 2004) (R3 +).

4.7. Standardization

To facilitate the interoperability between software systems, the formalization of a common, agreed on understanding of the relevant concepts is required. Such formalizations can be expressed as ontologies. In the context of distributed systems, ontologies manifest themselves on different levels of abstraction. These levels of abstraction include the network layer, the operating system, the computing platform (e.g., web service standards), data (further dividable into semantic, schema, and syntactic levels Bishr, 1998), interfaces, architectural paradigms (e.g., SOA), visualization techniques (e.g., pie charts, scatter plots), user interfaces (e.g., interaction techniques such as pan, zoom), and benchmarks (Andrienko et al., 2005; Laramee & Kosara, 2007; Schroeder, 2005). To achieve interoperability, approving shared conceptualizations on different levels as widely accepted standards is required. Appropriate standards on the lower levels (network, operating system, computing platform) are available. On the data, interface and architectural level, the OGC is in the progress of
approving standards in the geospatial domain that can be lever-
gaged to a certain extent in the geovisualization domain.

Standards do not exclude innovation and market opportunities. In a SOA, innovations primarily manifest in data models, service interfaces, and service compositions. Innovations can achieve the greatest benefit and impact, if they are interoperable with stan-
dardized data models, service interfaces, and composition mecha-
nisms (R5 +, R7 +, R11 +). In a standards-based SOA, innovations encapsulated within a service can be provided on different levels of interoperability: using a standardized interface and data model (e.g., improving quantitative, qualitative service characteristics, possibly offering proprietary extensions), using a proprietary inter-
face and a standardized data model, or using a proprietary inter-
face and data model. A challenge is to establish standards that allow the development and interoperable integration of innova-
tions. For instance, this can be approached by keeping standards flexible enough and by developing them when appropriate (R2 –).

Standards facilitate the comparability of data and implementa-
tions (Laramee & Kosara, 2007), allow competition, reduce the risk of vendor lock-in, simplify the outsourcing of services, platform independence, and allow for software systems of higher quality by profiting from the work of domain experts (R7 +, R4 +, R3 +). Standards are important for future widespread use and success (Laramee & Kosara, 2007) and interoperability generally “allows us to proceed at speed” (Andrienko et al., 2005) (R5 +, R8 +).

The OGC already approved relevant standards for the geospatial domain but many relevant standards are still missing especially on the higher layers of the layered SOA architecture. Moreover, there is a lack of formal standards specific to visualization and geovisual-
alization (Laramee & Kosara, 2007) (R2 –, R8 –). When applying standards, common possible disadvantages include that deciding standards takes time, adhering to standards can imply additional efforts, developed services can be ineffective if the standard does not include everything that is required or forces inappropriate behavior, and changing standards may invalidate legacy services (Andrienko et al., 2005) (R3 –, R7 –).

5. Summary and conclusions

In this paper, we establish the motivation for distributed sys-
tems and standards for 3D geovisualization. We introduced the fundamentals of the SOC and SOA paradigm, highlighted specific characteristics of the geovisualization domain relevant when building a SOA, and presented an architectural framework combinin-
g SOA concepts, geovisualization concepts, and OGC standards in a common conceptual frame of reference. We presented a case study including the service-based, image-based provisioning of and interaction with 3DGeoVE. Finally, we presented a summary of potential and challenges driven by SOA and standardization, and the more detailed potential and challenges of the case study.

SOA is an architectural paradigm and standards represent collect-
ively agreed concepts. In principle, functionality, every system that can be built based on SOA and standards can as well be built with different approaches. However, the resulting systems differ in primarily non-functional characteristics. In summary, we estimate that applying SOA and standards has its strengths in supporting integration, interoperability, platform independence, dissemination, distribution, and a raise of abstraction level. The approach seems well suited to support distributed CV, CMV, diversity, geovisualization processes, effective visual representations, and computa-
tional processing. We did not identify predominant advantages or disadvantages regarding complexity, non-functional features, economic requirements, styling, and dynamic data. We estimate that supporting high degrees of interactivity and massive data are most challenging.

Applying the SOA paradigm and open standards promises to move us closer to fulfilling the vision of connecting people, data and resources (Brodlie et al., 2005) and to providing geospatial information to everyone, everywhere, in appropriate and useful ways (Muntz et al., 2003). However, designing, implementing, and maintaining complex, interactive, data intensive, distributed systems possibly spanning multiple organizations over the Inter-
net introduces various complexities and challenges. Furthermore, standardization in the geospatial and geovisualization domain as well as research on and practice in applying the SOA paradigm in these domains is still in an early phase. To realize the identified po-
tential, a wide adoption of the approach and continuing work on standards and the open challenges is required.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.compenvurbys.2010.05.003.

References

Altmaier, A., & Kolbe, T. (2003). Applications and solutions for interoperable 3d geo-