CHALMERS

Rich Web Map Applications

An assessment of performance, functionality and implementation of Rich Internet Application techniques in web-based GIS

Master of Science Thesis in the Programme Software Engineering and Technology

HANNES JOHANSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, December 2010

Rich Web Map Applications
An assessment of performance, functionality and implementation of Rich Internet Application techniques in web-based GIS development

H. JOHANSSON

Examiner: Joachim von Hacht

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden December 2010

Abstract

Geographic information systems (GISs) are becoming more and more commonly deployed in a web environment. Web map clients are today heavily used by amateurs in a wide variety of applications. Web map application developers need to deal with the challenges of client-server transfer of sometimes great amounts of geographic data, diverse data formats, the limited application host environments that are the web browsers and the heterogeneous web environment.

In this thesis work, modern web technologies often referred to as Rich Internet Application (RIA) technologies have been studied to determined if and how they can be used to increase the user experience of web map applications. Relevant technologies have been discerned and their applicability to map applications has been assessed. The technologies have been defined in two main categories: plugin-based technologies (Flash and Silverlight) and HTML/JavaScript-based technologies (HTML5).

In the final stage of the thesis work, a prototype was developed to evaluate the applicability and performance some technologies that have been determined to be of most interest to this thesis. By utilizing HTML5 technologies such as SVG, Web Storage and video, it is deduced that the performance of HTML/JavaScript applications can be improved by using newer HTML5 technologies and that the user experience may indeed be improved.

Keywords: Geographic information systems, GIS, rich internet applications, RIA, HTML5, JavaScript, web map clients, user experience, caching, SVG

Table of contents

Table of figures

List of acronyms

RIA – Rich Internet applications
GIS – Geographic Information System
OGC – Open Geospatial Consortium
WFS – Web Feature Service
WMS – Web Map Service
GML – Geography Markup Language
XML – Extensive Markup Language
SVG – Scalable Vector Graphics
W3C – World Wide Web Consortium
HTML – Hypertext Markup Language
JS – JavaScript
DOM – Document Object Model
UI – User interface
GUI – Graphical User Interface
SGML – Standard Generalized Markup Language
DTD – Document Type Definition
XSL – Extensible Stylesheet Language
XSLT – Extensible Stylesheet Language Transformation
CSS – Cascading Style Sheets
XAML – Extensible Application Markup Language
HTTP – Hypertext Transfer Protocol
WPF – Windows Presentation Foundation
MXML – Magic Extensible Markup Language/Macromedia Extensible Markup Language
X3D – Extensible 3D Graphics
API – Application Programming Interface
E4X – ECMAScript for XML
RPC – Remote Procedure Call
SDK – Software Development Kit
URL – Uniform Resource Locator
URI – Uniform Resource Identifier
FXG – Flash XML Graphics
SWF – Small Web Format/ShockWave Flash
AS3 – ActionScript 3
VML – Vector Markup Language
XHR - XmlHttpRequest

1 Introduction

This chapter describes the background of the thesis project, defines its purpose and the methods that were used throughout the project.

1.1 Background

In recent years there has been a great proliferation of web-based mapping applications. While geographic information systems (GIS) were traditionally used mainly by a relatively small group of experts, the general public has now been introduced to mapping applications by the increasing number of free web-based mapping applications that use a set of different technologies popularly referred to as Rich Internet Application (RIA) technologies.

Traditionally, web mapping applications have used technologies and techniques such as JavaScript and Ajax to create advanced web mapping features. Asynchronous calling of remote procedures, intelligent caching, image tiling and advanced user interface construction has enabled developers to create rich, interactive mapping applications. Even though the field has seen great development, traditional technologies have still suffered from inherent limitations such as limited multimedia support, graphics rendering capabilities and cross-platform interoperability issues.

In recent years, other web technologies run applications on web browser plugins that have their own run-time environments have become popular, such as Adobe Flash and more recently Microsoft Silverlight. These new RIA platforms have introduced new opportunities for multimedia playback, vector graphics rendering and animation, extended caching capabilities and network protocols, as well as advanced development environments and tools. Web applications and desktop applications have thus started to converge and are continuing to converge.

In recent years, development of the next incarnation of the HTML standard has begun, and in the HTML5 draft specification there is support of video and audio playback, native vector support, as well as many other new features aimed at adapting HTML to web application development and bringing some of the capabilities of plugin-based technologies to HTML.

Some of the opportunities presented by these new technologies should be able to improve the user experience in web mapping applications. With older technology, the web mapping applications have mostly displayed tiled raster images and standard HTML user interface controls. Recent technology advancements should open up opportunities to create more dynamic and interactive maps.

Carmenta is a company in the geospatial information technology field that has developed a JavaScript/Ajax-based web map client that interacts with their Carmenta Server product, which is a web map server. Carmenta Server implements several standardized interfaces for building web mapping applications, such as the OGC WFS and OGC WMS standards. These services can be used when building web mapping RIAs.

1.2 Purpose

The purpose of the thesis is to survey and evaluate different modern web application technologies associated with Rich Internet Application concepts in the context of web mapping services in terms of interactivity and user experience. The goal is to increase user experience and interactivity of web mapping applications in relation to the current Carmenta JavaScript/Ajax web map client.

1.3 Method

The thesis work has been carried out in three parts. In the first part, a survey of traditional and modern web mapping application technologies as well as the most popular technologies associated with RIAs was performed. An analysis of the collected literature was then made to filter out the relevant technologies for increasing the user experience and interactivity of modern web mapping applications.

In the second part of the thesis work, a comparative study was performed to scrutinize the RIA technologies in relation to the web technologies currently in use by the Carmenta web map client. This was done by comparing a set of procedures and functions that were considered most important in the context of web mapping applications, such as remote procedure calls, caching, XML parsing and vector rendering, when implementing them with HTML5 technology and with plugin-based technology.

Finally, a prototype was implemented by using the technologies selected on basis of the comparative study. This prototype was then evaluated and compared with Carmenta's current JavaScript-based web map client in terms of user experience and interactivity. Conclusions about the eligibility of the selected technologies and future development potential were then made.

2 Geographic Information Systems (GIS)

This chapter describes several GIS-related concepts, technologies, definitions and standards for both traditional (desktop) GIS and web-based GIS. It also discerns some recent trends and future prospects of the field.

2.1 General GIS

A geographic information system can be described as a system of computer software and hardware components for capturing, storing, updating, manipulating, analyzing and displaying geographically referenced data and associated attribute data. [1] A GIS typically contains a user interface (UI), a spatial database containing geographic data and tools for analysis and visualization of these data and their relationships. Furthermore, a definition of GIS can also include the process of entering, managing and using the geographic data in addition to its software and hardware components. [4]

Traditionally, GIS software has been installed as desktop applications. Because GIS often contain large amounts of data and perform complex spatial analyses, such systems have been difficult to integrate and disseminate. [2] With the vast amount of data associated with geographic analysis, the data structures used for representation and communication are often very large. Many software systems and file formats are also proprietary and thus difficult to integrate with each other. Thus, GIS have typically had rather complex, centralized and tailored architecture.

GISs have a wide variety of uses, such as collection and management of remote sensing data, homeland security or emergency planning and other decision-making applications. Originally, GIS software was mostly used by experts because of their complexity and need for technical training. Today, the user-base has grown significantly mainly because of the advent of web-based easy-to-use mapping applications. [3] This increase of users has paved the way for an expansion of the market and new applications for mapping solutions. Even though this development presents many possibilities for GIS developers, it has also raised new challenges of creating more accessible user interfaces and of creating more interoperable and de-centralized systems.

2.1.1 Raster and vector representation

Geographic data is visualized by GIS mostly as either raster images or as vector graphics. Most GIS are capable of visualization in both formats, and the two formats have different strengths and weaknesses and can thus be optimally used for different purposes.

A raster image basically consists of a grid of cells, or pixels. Each cell has a certain set of attribute data associated with it, e.g. the type of feature it contains, such as a road,

forest, building and so on. Thus, a raster image can be described as a pixel-by-pixel representation of the real world. Raster images can have different resolutions. The higher the resolution the more data is needed and the more accurately the image can depict the real world. Cells in the raster image are referenced in a coordinate system, e.g. so that the top left cell has the coordinate (0, 0) and the cell to the right has coordinate (0, 1) and so on.

In a vector representation of geographic data, the map contains a set of features made up by points, lines and polygons. Each point in a feature has a specific coordinate in a coordinate system and each feature also has associated attribute data. Additionally, lines and curves can be described by different mathematical functions instead of a collection of connected points in a coordinate system. [4]

Raster images are typically used as backdrops with layers of vector graphics overlaid on top. This is because raster images are typically more visually detailed and less expensive in terms of data collection and structural complexity. The vector format, on the other hand, offers capabilities for topology representation and manipulation of geographical features. Which format to use, or whether to use a combination of the two, is mainly dependent on each specific application. [5]

2.2 Web-based GIS

Along with the development of Internet technology and world Internet access coverage, GIS have developed from being mainly centralized desktop systems to also providing services for web mapping applications. This allows users and applications to access geographic information and map visualizations from anywhere, without the need of local installations of complex systems. Through the development of advanced web technologies in the recent decades such as tile servers and Ajax in combination with improvement of Internet data transmission capabilities, desktop GIS and web mapping applications are in some aspects converging. [3] [6]

Web-based GIS need to deal with challenges such as interoperability, data communication, file formats and interface standardization as they are deployed in a highly heterogeneous environment with relatively low data transmission rates.

2.2.1 Interoperability and standards issues

Even though GIS have improved interoperability and become more standardized in recent years, there are still many challenges to be met. Collecting, storing and updating geographic data is usually very expensive on a larger-than-trivial scale and there are relatively few providers of geographic data. Most providers are government agencies or large enterprises. Few, if any, data providers have a complete set of data for general-purpose web mapping applications. Instead, web mapping applications often have to

collect and integrate geographic data from many different providers. No standards regarding service interfaces or data formats have yet been uniformly accepted by data and service providers. Retrieval and integration of geographic data is therefore a big challenge when developing web mapping applications.

2.2.1.1 Geographic data communication

In web-based GIS, geographic data needs to be transported from a remote server to the client making a request for the data. For the requesting client to correctly interpret the response, the data needs to be encoded in a format the client can understand and analyze. In a tightly coupled client-server architecture, where the developer of the client has thorough knowledge of the server's data encoding, this can be done in a straight-forward way with optimized performance. However, in loosely coupled client-server architectures the client developer may not have knowledge of the server implementation or internal data formats. The server thus needs to respond in some standardized format so that any client application can receive and interpret the response. One attempt to create an open standard for data encoding is XML (eXtensible Markup Language).

2.2.1.2 XML and XSLT

XML is a W3C Recommendation developed by the XML Working Group, a W3C organization. The language is a restriction of SGML (Standard Generalized Markup Language). [7] It was developed with the goal to create a means of encoding data in a form that:

- is both human-readable and machine-readable, as well as highly usable over the Internet
- can be used in a wide variety of applications
- is highly flexible

An XML document consists of a number of user-specified (the user in this case refers to the creator of the XML document) tags or elements to represent information. New tags (elements) can be added to the structure flexibly and elements can be nested to create complex data structures. Elements also have different user-specified attribute data and can contain values of different kinds, such as PCDATA (Parsed Character Data) or CDATA (Character Data). Users are thus able to create customized data structures or extend existing ones at will.

The structure of a specific XML document is defined by a document type definition (DTD), or by an XML Schema. An XML Schema is itself written in XML and is thus machine-readable as well as human-readable. If an XML document does not conform to its DTD or XML Schema, it is not well-formed. This means that Schemas or DTDs can be used to define specific dialects of XML, thus creating application-specific XML-based

markup languages, or formats. HTML and XHTML are two such application-specific dialects of XML.

When separate software components communicate using XML documents, they need to understand the structure of the XML documents to be able to parse or write to the document. That is, they need to know the XML Schema or the DTD of the document. One application might use a different internal data structure from the structure of the received XML document. In this case, the received structure needs to be converted into the application's internal data structure. XML documents can thus be used as a data communication format rather than an internal storage format.

Conversion of different XML formats can be done by using the extensible stylesheet language (XSL) to perform a XSL transformation (XSLT). An XSL document contains a set of rules for how an XML source document should be transformed into a resulting XML document. It is thus possible to convert one XML format such as GML (Geography Markup Language, discussed further below) to a different XML format such as SVG (Scalable Vector Graphics, discussed further below). [8]

2.3 Open Geospatial Consortium (OGC) initiatives

As stated on their website, "The Open Geospatial Consortium, Inc (OGC) is an international industry consortium of 398 companies, government agencies and universities participating in a consensus process to develop publicly available interface standards." [9]

The goal of OGC is to achieve greater interoperability in GIS by developing open specifications for standard architectures, interfaces and data formats. By conforming to any of these specifications, developers can guarantee that their services can be accessed and used as defined in the respective specification with expected response formats. A subset of the OGC specifications is described below.

2.3.1 The OGC Web Map Service (WMS) specification

A server that conforms to the OGC WMS specification mainly produces image maps. When receiving a well-formed HTTP GET request, it generates an image of the requested geospatial data and returns it to the client. The HTTP GET request includes well-defined parameters in a key-value pair structure that specify which server operation to invoke and some additional mandatory and possibly also some optional operation parameters.

A request can look like the following: 'http://hostname/wmsserver?service=WMS&request=GetCapabilities'. In this example, the two key-value pairs are service=WMS and request=GetCapabilities, which specify that the operation GetCapabilities of the WMS service should be called.

The server then responds in a well-defined format as defined by the WMS standard specification. For example, the response to a well-defined call of the operation GetMap would be a raster image of the specified map in a file format specified by the requesting client. Often the maps produced are in the PNG, GIF or JPEG file formats, but some servers may also produce maps in SVG format.

There are two conformance classes of the specification: one *basic* WMS standard which only implements the mandatory operations and one *queryable* WMS standard which conforms to the *basic* WMS standard but which also implements the optional operations. [10] The two mandatory operations are described in more detail below.

2.3.1.1 GetCapabilities

The GetCapabilities operation returns metadata about the WMS server. The operation requires the two HTTP GET parameters service and request to be WMS and GetCapabilities respectively. It also accepts the three optional parameters version, format and updatesequence.

The returned metadata contains metadata describing the server and information about the server's capabilities and its contained geographic information structured in layers as well as layer metadata. Layer metadata includes layer title, name, coordinate system (CRS) and bounding box.

2.3.1.2 GetMap

The second mandatory operation that a WMS server must support is GetMap. This operation is used to retrieve a map specified by the provided parameters. Required parameters are version, request (must be set to GetMap), layers, styles, CRS, BBOX (bounding box), width, height and format. Optional parameters are transparent, bgcolor, exceptions, time and elevation. [10]

The returned map is a raster image in the format specified by the format parameter, most commonly JPG, GIF or PNG. The map contains a representation of the geographic data within the specified bounding box and layers. The map projection is done with the coordinate system (CRS) specified in the request, if the specified CRS is supported by the server. Visualization styles of the geographic data can be specified with the styles parameter. The width and height parameters set the dimensions of the output image.

2.3.2 OGC Web Feature Service (WFS) standard

While a WMS returns raster images, a WFS returns geospatial data in the Geography Markup Language (GML) format. GML is an XML dialect used for encoding geospatial

data and is described further below. By requesting and receiving GML documents, a client can collect information about geospatial features structured according to the GML specification. Clients may also be able to update a server's geospatial data through WFS in transactions in case the server conforms to the Transactional WFS standard.

Similarly to a WMS, a WFS server receives HTTP GET requests containing key-value pairs. However, a WFS can also receive HTTP POST requests and the request filters can then be encoded in XML instead of key-value pairs. A client can retrieve geospatial feature information from several different WFS servers and integrate them for some visualization, analysis or storage.

A geographic feature is described in GML format and can represent geo-referenced objects such as roads, buildings, countries or other geographic objects described by different geometries. Each feature has a set of associated attributes and some geometry expressed in GML. WFS servers support INSERT, UPDATE, DELETE, LOCK, QUERY and DISCOVERY operations on geographic features.

WFS is separated into three different classes: *Basic* WFS, *Xlink* WFS and *Transaction* WFS. The basic WFS support the operations GetCapabilities, DescribeFeatureType and GetFeature. A basic WFS is a read-only WFS. An *Xlink* WFS implements the basic WFS but also the GetGmlObject operation which can be used for remote or local Xlinks. A *transaction* WFS implements the *basic* WFS but also the Transaction operation and optionally also the GetGmlObject and/or LockFeature operations. The GetCapabilities operation is similar to the GetCapabilities operation in WMS which is described above. The DescribeFeatureType and GetFeature operations are described further below.

2.3.2.1 DescribeFeatureType

The DescribeFeatureType operation is used to retrieve information about the different feature types serviced by a specific WFS server in the format of a GML application schema defined by an XML Schema. The information describes how specific features will be generated on output and how the WFS expects specific features to be encoded on input. A client can call the DescribeFeatureType operation to learn about the feature schemas so that the client can receive and interpret geographic features correctly and also so that the client insert or update features properly.

2.3.2.2 GetFeature

The GetFeature operation is used to retrieve a set of geographic features most commonly described in a GML document, although other encodings may also be supported by a WFS. However, GML is the only format a WFS is *required* to support. A feature is described by an XML element within the resulting GML document. Within the feature element there are a set of nested elements that define the properties of the feature.

The value of these properties can sometimes be other features. A WFS can respond to a GetFeature request by either generate a document containing all the matching features, or it can return the number of features that match the request. This behavior is specified by setting the resultType parameter of the GetFeature request. [11]

2.3.3 OGC Geography Markup Language (GML) standard

GML is an XML grammar written in XML Schema that is used to describe geographic application schemas, and also to encode geographic information mainly for transportation but also possibly for storage. GML can be used to describe geographic features and other geographic information in a structure that is defined by the associated GML application schema.

A feature is described by a set of properties as defined by its type. Feature properties can be other features, feature meta-information or feature geometries. Several features can be contained in a feature collection which may itself be considered a feature. Thus, a feature collection also has a feature type.

Feature types are described in GML application schemas specified by XML Schemas. GML application schemas can be designed by developers by extending or restricting the GML schema; hence, GML application schemas must import the GML schema. The GML schema defines the structures of features, feature collections, coverages, primitive and complex geometries, coordinate systems and the many other elements of GML. [12]

As GML is an XML grammar, it is extremely flexible, allowing designers to develop any number of different application schemas. It inherits the benefits of XML in terms of interoperability, human-readability and flexibility. However, the flexibility also introduces challenges to users of GML. Application developers need to enable the applications to interpret and parse user-defined GML application schemas. It may be easy to achieve this if every GML application schema to be used in the application is known in advance by the developer, but a general GML application schema interpreter is considerably more complex. [13]

Another issue with GML is that it is an inherently large-size data format. Descriptions of geography usually involve a lot of data and thus very large GML documents. This means that transferring, reading and parsing GML documents is relatively expensive in terms of time and memory consumption. GML documents also tend to be quite difficult for general users to understand. As GML is an XML grammar, it is possible to transform GML to another XML format for visualization such as SVG by using XSLT. [14] [15]

3 Rich Internet Applications (RIAs)

The following chapter surveys different RIA-related technologies and gives an overview of their most prominent concepts and features.

The term Rich Internet Application is loosely defined and can refer to a variety of technologies. It was coined in a white paper from Macromedia – which later merged with Adobe – in 2002. In the mid 1990s, there was an enormous growth of Internet users and applications. Internet applications traditionally used a thin-client architecture based on HTML. This made it easy to deploy applications to a large number of remote clients. The most processing was carried out on powerful servers that sent the processed response back to the thin clients. This architecture has inherently put many constraints on the development of Internet applications in terms of complexity, media playback and interactivity. Rich Internet Application technologies were developed to overcome these limitations of thin-client HTML-based architectures. [16]

Most RIA technologies run applications in some special run-time environment in a web browser plugin. Thus the applications can be platform-independent and interoperable and can be easily deployed in the heterogeneous environment that is the Internet. The run-time environments also ideally provide improved performance over traditional HTML-based models.

RIAs typically transfer much of the computational work from the server to the client. The server mainly hosts the data content while the processing of the content is done on the client-side. This allows the client to make less remote calls to server-side functions for processing and computation, thus saving network communication time.

The main goal of RIA technology is to increase the user experience of web applications and to allow web applications to converge with traditional desktop applications. As the Internet user base has grown enormously and continues to grow, it is becoming more important to improve the usability of web applications and add richness to the users' experience in terms of interactivity and good user interfaces. [17]

However, a major aspect of RIA technologies is the developer's perspective. Creating RIAs in an HTML environment requires the developer to have knowledge in a large number of technologies and techniques. It may not be feasible for one single developer to acquire knowledge of such variety of technologies. Research in RIA technology is working towards simplifying development by using common architectures and concepts as well as concepts of reusability, integrated development environments and platforms, service orientation and component-oriented architectures. [16] [18] [19]

This thesis makes a distinction between HTML and JavaScript-based RIA technologies and plugin-based technologies. The term plugin-based RIA refers to RIA

technologies that run applications in special run-time environments through web browser plugins. Examples of plugin-based technologies are the Adobe Flash and Microsoft Silverlight platforms.

3.1 Overview of RIA technology and definitions

There is no wide consensus of what constitutes RIA technology; it is a loosely defined term. However, web applications referred to as RIAs typically show a set of characteristics such as thick clients, asynchronous client-server communication, interactivity, rich user interfaces or more loosely a “desktop feel”.

There are a number of different technologies that are sometimes referred to as RIA technologies. Some define RIAs as web applications that run in specialized run-time environments via browser plugins or virtual machines in sandbox environments. The most popular platforms using this approach are Microsoft Silverlight, Adobe Flash and Java/JavaFX with market penetration rates of 54.56%, 96.59% and 79.37% respectively as of August 2010. [21] [22]

Others also consider JavaScript and Ajax-based web applications to be RIAs. Although JavaScript and Ajax-based applications do not run in browser plugins, but instead run in the native browser environment, they can still exhibit some of the RIA characteristics such as rich user experience, asynchronous background client-server communication, interactivity and a transfer of computation and processing complexity from server to client-side. [20]

However, as JavaScript applications run in the web browser environment, their platform independency can be debated, as different web browser have different support for JavaScript and Ajax functionality. JavaScript also lacks a rigorous set of standards and is a very heterogeneous development platform, which can increase application development efforts. [25] There is also debate over whether or not JavaScript and Ajax technologies actually provide the richness and performance of other RIA platforms. [23] [24]

The HTML5 draft specification, which is already partially implemented in many web browsers, introduces many new features that web application developers can utilize to create more RIA-like applications. Some of these new features are parallel JavaScript worker processes, extended local caching in the web browser, new form elements and native support for SVG, video and audio. [69]

Which RIA technology to choose for developing web applications largely depends on the specific application to be developed, and also on the developers’ skill-set and preferences. [25] [26] Differences between JavaScript/Ajax-based RIAs and the plugin-based RIA platforms Microsoft Silverlight and Adobe Flash are discussed below.

3.2 JavaScript and Ajax-based RIA technology

JavaScript is a dialect of the scripting language ECMAScript which is officially managed by the Mozilla Foundation. A scripting language is a programming language that controls the behavior of an existing application or system. That means that there is a core application that hosts the script and provides core functionality and object models. Script code is often interpreted from the source code, or bytecode, instead of being pre-compiled as many traditional programming languages. Such programming languages are called interpreted languages. [27]

ECMAScript – or JavaScript specifically – is a scripting language that is often used to create web applications. The JavaScript code is hosted by the web browser core functionality and object model. As there are many different web browsers that support JavaScript, each web browser is a different host environment for the JavaScript code. The web browser contains a JavaScript engine that interprets and executes JavaScript code and an object model, such as the Document Object Model (DOM). The JavaScript code is contained within the HTML and reacts to user interactions in an event-driven manner. [27]

The web browser provides functionality for the embedded JavaScript code, such as objects – exposed by the DOM API – representing windows, menus, pop-ups, dialogue and other user interface components. The hosting web browser also provides event propagation and handles attachment of JavaScript code – so called “observers”, event-listeners or event-handlers – to the different events. Events can be triggered by user interactions such as mouse-clicks or by document state changes such as the completion of loading a document component. [27]

Most web browsers use the DOM for objects representation and event management. XML or HTML documents are represented in the DOM as a tree of nodes – that is, with one root node that contains child nodes, which may each in turn contain further child nodes – referred to as the DOM tree. Each of these nodes is represented in the DOM as objects which can be manipulated by script code such as JavaScript via the DOM API. In recent years, a standardization of the DOM API has been developed by W3C to increase interoperability and strive towards platform and programming language independency. However, different web browsers still have different support for the DOM. [30] [31] [32] [33]

3.2.1 Ajax

Originally an acronym for Asynchronous JavaScript and XML, Ajax is not actually a single technology but a combination of several different technologies. The term was coined by Jesse James Garrett in 2005. [29] Ajax applications use the XMLHttpRequest object in the web browser to retrieve data from a server or service asynchronously. This means that data can be loaded into a web application in the background, without

interfering with the user interactions with the application. This enables a web application to present new information to the user by partially updating the screen without having to reload the web page or load a new one, as had traditionally been the paradigm for updating information on a web page. Ajax thus increases the dynamic and interactive behavior of web pages and can enrich the user experience.

Regardless of its name, Ajax does not rely on the JavaScript and XML technologies specifically. It has rather developed into a web application development paradigm independent of language or technology. It may be implemented in the Java or ActionScript languages just as well as the JavaScript language. [28] However, the term Ajax still commonly refers to the asynchronous JavaScript and XML combination of technologies.

The Ajax technique have become very popular in the past decade, largely because of some “killer apps” using Ajax developed by Google such as Gmail, Google Suggest and Google Maps.

3.3 Plugin-based RIA technology

This section describes the plugin-based platforms Adobe Flash and Microsoft Silverlight. They both differ from JavaScript technologies in that they run applications in their own run-time environments via the browser plugin. This frees application developers from the constraint of using only browser-provided functionality as in JavaScript applications. They also differ from JavaScript technologies in that they each provide an integrated development environment, with standard libraries, frameworks, tools and debugging functions.

Since Flash and Silverlight applications run in browser plugins, each browser that supports the plugin also supports all the functionality of the respective technology. That is, Flash and Silverlight applications are compatible with all the browsers that have the corresponding plugin installed. There is no need for developers to tailor the applications to any combination of browser technologies, as long as the browser has the required plugin. Adobe Flash has a penetration rate of 96.59% which means that almost all web browsers in use are compatible with Flash applications, making Flash almost ubiquitous. Microsoft Silverlight is a younger technology but it already has a penetration rate of 54.56% in web browsers, and the number can be expected to grow. [21] [22]

The aim of plugin-based technologies is generally to create a richer user experience by providing rich multimedia (video, audio), vector graphics and animation support, by improving the performance and responsiveness of applications and providing a richer development environment.

3.3.1 The Adobe Flash Platform overview

The Flash technology platform was developed by Macromedia who later merged with Adobe. The Adobe Flash Platform today encompasses several tools, technologies and services, such as the Adobe Flash Professional and Flex platforms and the Flash Player. The Macromedia Flash MX platform is a predecessor of Adobe Flash Professional and was presented in 2002 as a new platform for developing rich web clients in the white paper [16] which also coined the term Rich Internet Applications.

Applications developed in Flash run as binary Flash files and utilize the programming language ActionScript for adding interactivity and other functionality. Flash files have the Shockwave Flash/Small Web Format (SWF) format and can be run by the Adobe browser plugin called Flash Player. Flash applications are compiled into SWF files, which contain run-time code combining source code, data and media.

The Flash development platform was originally more of a multimedia-authoring platform than an application development platform, using concepts such as timelines. Thus it can be confusing for programmers to use. The Flex platform, which is a Software Development Kit (SDK), was therefore developed to serve as a more programming-oriented platform to simplify development for programmers.

Flex combines the tag-based language MXML and ActionScript to compile SWF files. Thus, both the Flash and the Flex development platform use ActionScript to compile SWF binary files to be run in the Flash Player. However, Adobe Flash is more suited for multimedia authoring using the timeline concept and Adobe Flex is more suited for programming and application development. [42]

3.3.1.1 Adobe AIR

In 2008 Adobe released their run-time environment and SDK called Adobe Integrated Runtime (AIR). It was an attempt to integrate several different techniques, tools and technologies such as JavaScript, ActionScript, Flash, HTML and Ajax. However, AIR applications are deployed as desktop applications and not web applications. This is done by adding special AIR packages, libraries and APIs to the existing web technologies to integrate them with the desktop environment.

The motivation for using technologies such as JavaScript or HTML for creating desktop applications is mainly simplicity and small file size. While there are programming languages such as Java or C++ that provide deeper access to desktop application functionality as well as access to web services, such languages are often quite complex and it takes time and effort to learn them and to create applications or systems. By using the AIR platform, developers can focus more on user interfaces and the more familiar web technologies and let the run-time handle desktop application functionality and thus save

development time. Since the application files are also typically relatively small, they are also suited for online distribution. [44]

3.3.2 Microsoft Silverlight overview

Silverlight was announced by Microsoft in 2007 as a tool to build RIAs with an open approach, using HTML and XAML, by using platforms such as Visual Studio .NET and Expression Blend. Even prior to the announcement, Microsoft had been talking about such technology, but had then used the term WPF/E – Windows Presentation Foundation Everywhere – which was later changed into Silverlight to better reflect the new technology. Silverlight is a sub-set of the WPF platform, which is reflected in the original WPF/E term. [45]

Silverlight 4 is the latest incarnation and was released in 2010. Just like Adobe's Flash, it runs in a web browser plugin and needs a run-time environment to be installed. The run-time is a Silverlight-specific version of the Common Language Runtime (CLR) of the .NET platform. Silverlight 4 is supported by the most popular web browsers including IE8, Mozilla Firefox, Apple Safari and Google Chrome on the Windows, Mac OS X and several Linux-based operating systems. [46] [47]

Silverlight applications can be developed using any of the .NET-supported programming languages, which include C#, VB and JavaScript. The Silverlight platform integrates the DOM model and provides an interface for communication between external JavaScript code. Microsoft advocates development in Visual Studio and Expression Blend. Silverlight applications can also use a specific sub-set the .NET framework library classes. [46] [48]

XAML – meaning Extensible Application Markup Language – is an XML dialect developed by Microsoft mainly for marking up user interface controls and .NET object hierarchies. In Silverlight development, XAML can be used for creating the layout of Silverlight applications, while the .NET-supported programming languages add interactivity to the applications – so-called code-behind. [46] This works similar to Adobe's Flash technology, which uses MXML and ActionScript correspondingly.

The aim of the Silverlight platform is to improve the richness of web applications and simplify and empower RIA development. Like Adobe Flash, Silverlight supports multimedia playback, vector graphics, animation and customizable user interface controls. [46]

3.4 HTML5 RIA technology

As applications and multimedia has become a larger and larger part of the Internet usage, the HTML4 specification has become more and more unfit for developing modern

web sites. HTML4 has virtually no support for representing multimedia such as video or audio, and it is also very limited in web application development as it is very constrained in terms of caching and it also lacks support for multi-threading JavaScript procedures.

To address this, the W3C has re-formed the HTML Working Group to develop an incremental update to the HTML4 standard together with the WHAT Working Group (WHATWG) – it has been dubbed HTML5. Some of the goals of HTML5 are to introduce native support for video and audio content as well as vector graphics. It also introduces support for multi-threading JavaScript via Web Workers, adds increased support for caching through HTML5 storage, enables offline usage of web applications and adds a set of new elements such as new form elements – as well as many other new features. HTML4 fully conforms to HTML5, which means that every web site conforming to HTML4 will also work properly in HTML5.

This chapter briefly describes Web Workers, storage, forms and offline use; vector graphics and multimedia support is discussed in the subsequent chapter.

3.4.1 Parallel JavaScript processes – Web Workers

In JavaScript code can only run in one single thread, forcing execution to be sequential. This means that when a web site uses some computationally heavy JavaScript code, it executes that code in the same thread that it uses to display and manage the user interface. With very expensive computations this might cause the interface to freeze for an unacceptably long time or even the web browser to crash.

To resolve this problem HTML5 has introduced something called Web Workers that can run in parallel with the main JavaScript thread. They are not technically multi-threaded but aim at simulating multi-threaded behavior. Web Workers can perform computationally heavy procedures in the background, thus relieving the main JavaScript thread of performing computations that could freeze up the user interface and decrease user experience.

To retain thread safety and other security aspects there are many constraints on Web Workers. Data sharing is done through message passing between different workers and the main JavaScript thread. Web Workers also have no access to the DOM; hence they are best suited for performing computationally heavy calculations, but not manipulating the DOM tree. Web Workers are relatively expensive in terms of start-up performance cost and memory cost, and so they are not meant to be used in very large numbers and are meant to have a long lifetime. [67]

To use a web worker, a `Worker` object is created in the main JavaScript code. The constructor of the `Worker` objects takes as a parameter the URL to a separate JavaScript file containing the worker procedure. The provided procedure is then executed in a separate

thread and can send messages back to the main JavaScript containing data which can then be used in the main JavaScript. To receive responses from the worker, the main JavaScript needs to define a function as an event listener, listening to the onmessage event, where the response can be handled. The following example from [67] shows a web worker computing prime numbers in a separate thread in the background:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Worker example: One-core computation</title>
  </head>
  <body>
    <p>The highest prime number discovered so far is: <output id="res-
ult"></output></p>
    <script>
      var worker = new Worker('worker.js');
      worker.onmessage = function (event) {
        document.getElementById('result').textContent = event.data;
    };
    </script>
  </body>
</html>
```

The worker procedure itself is stored in a separate JavaScript file named worker.js and looks as follows:

```
var n = 1;
search: while (true) {
  n += 1;
  for (var i = 2; i <= Math.sqrt(n); i += 1)
    if (n % i == 0)
     continue search;
  // found a prime!
  postMessage(n);
}
```

Web Workers are currently supported in Firefox 3.5+, Safari 4+ and Chrome 5+.

3.4.2 HTML5 Web Storage

In HTML4, web sites have been able to cache some information in website-specific cookies. However, the storage space provided by cookies is very limited and often not sufficient for fairly advanced web applications. Cookies are also transmitted to the server with every request – keeping a large amount of data in a cookie would therefore be expensive in terms of performance. Yet another restriction of cookies is that they can leak page state information across windows. This means that if a user interacts with the same page in different windows and the page uses a cookie to store state information, and the

user makes a new request by e.g. clicking something on the page in one window, this cookie's state information would affect the state of the page in the other window, possibly without the user being aware of this. This can be critical in particularly pages such as booking systems etc.

HTML5 introduces a concept called Web Storage for storing data on the client-side. By using the two properties sessionStorage and localStorage, information can be stored in the web browser either until the user closes the window or permanently, across sessions. By storing data in the sessionStorage attribute, each page has a session storage area that is unique to each window and that is stored until the window is closed. [68] The following simple example shows how the sessionStorage attribute can be used to store session-specific information:

```
<form>
  <input type="checkbox" onchange="sessionStorage.subscribe = checked" />
  I want to subscribe to the newsletter!
</form>

If (sessionStorage.subscribe) { … }
```

The localStorage attribute stores site-specific information that is persistent across sessions in the web browser's local storage area in a similar fashion. A difference from sessionStorage is that the stored information is accessible across windows. The following example shows the use of localStorage:

```
<form>
  <input type="button" onclick="handleClick();" />
  <p>
    You have clicked <span id="numberClicks">an unknown number</span> of times.
  </p>
</form>

<script>
  function handleClick () {
    if (!localStorage.clicks) { localStorage.clicks=1; } else { localStorage.clicks++; }
  }

  document.getElementById("numberClicks").textContent = localStorage.clicks;
</script>
```

3.4.3 HTML5 Offline Web Applications

Many web applications require a connection to the web server to function. If a user goes offline, the content of the web application often becomes inaccessible and the user has to rely on the small HTTP cache to view content offline. In HTML5 it is possible to use offline application caching. This is done by setting the manifest attribute on a HTML

page. The attribute gives a URI to a manifest file which states which of the application resources should be cached. [69] Below is an example from [70] that shows a manifest file that declares which resources the application requires the web browser to fetch from the server for offline caching:

```
CACHE MANIFEST
index.html
help.html
style/default.css
images/logo.png
images/backgound.png

NETWORK:
server.cgi
```

In the example above, the file(s) declared under CACHE MANIFEST will be fetched from the server and cached locally for later offline retrieval, while the file(s) declared under NETWORK: will be required to be fetched from the server in each request.

3.4.4 HTML5 UI elements

Traditionally web developers have been rather constrained when designing web page user interfaces since they have been restricted to use the native HTML4 elements and user controls. The alternative has been to create custom controls in CSS or JavaScript, a task which can be rather time-consuming and cumbersome. In HTML5 there are several new elements and more specifically the HTML5 introduces new form features. For a full overview of the new elements refer to [69]. A few of them are listed below:

- Placeholder text in input fields gives the fields a grayed-out pre-set default text.
- Autofocus on input fields automatically puts the focus and marker on the input field.
- Slider controls for selecting numbers.
- Date and color picker controls are now native to HTML5, removing the need to use JavaScript for such functionality.

Browser support for new elements is still lagging behind considerably, but with the use of fallback for non-supportive browsers some elements may still be appropriate to use in web applications, and the browser support can be expected to increase in the future.

3.5 Vector graphics

Vector graphics technology utilizes geometrical primitives such as points, lines and polygons to create images with mathematical expressions and functions. They are different

from raster images traditionally more used on the web, which are images created by drawing an array of pixels in different colors.

Vector graphics have several advantages over raster images in certain applications on the web. As vector graphics consist of a set of geometrical shapes expressed mathematically, they can be considerably smaller in file size than raster images. Vector graphics are also very well suited for scalability, as opposed to raster images which become blurry when scaled. The geometrical shapes in vector graphics can also be manipulated and transformed by using well-known vector mathematics, and thus vector graphics are well suited for certain web-based animations and user manipulations.

Vector graphics can be drawn in many different ways. They can be drawn by using some programming language, or they can be expressed in markup languages such as XML. For web applications, there are a number of different formats for vector graphics. Many formats are proprietary and non-compatible. Hence, W3C has made an attempt to create a standardized format for marking up vector graphics called Scalable Vector Graphics (SVG), which is an XML-based format. Some other vector graphics formats are Vector Markup Language (VML) which was developed by Microsoft, Adobe's SWF format which is used in Flash and the Adobe Illustrator (AI) format.

3.5.1 SVG

SVG is an XML-based vector markup language developed by W3C to create a standardized vector graphics format to be used on the web. It enables styling of elements and shapes, transparency, filter effects and animation either through embedding of animated SVG elements or scripting. Access to individual SVG elements is given through the SVG DOM; hence it is possible to access and manipulate SVG elements with scripting languages such as ECMAScript-based languages. Below is a small example drawing a rectangle containing a circle, also giving the elements IDs to allow access from scripts through the DOM:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg id=”mySvg” width=”1000” height=”800” version=”1.1” >
  <rect id=”myRect” width=”500” height=”300” stroke=”black” fill=”blue”>
    <circle id=”myCircle” cx=”200” cy=”150” r=”50” stroke=”red”></circle>
  </rect>
</svg>
```

Because SVG is XML-based, it can be generated, transformed and styled by using XSLT techniques. Thus, a styled SVG document can be generated from a different format XML source, such as a GML document. Conversely, an SVG document can also be transformed to other XML-based formats with XSLT. The SVG rendering style can also be

defined by a Cascading Style Sheet (CSS) document. SVG also gains other XML benefits, such as human-readability and extensibility, but it also inherits the disadvantages of XML, such as large document sizes.

By using the SVG DOM, events such as mouse-interactions or state changes and other SVG-specific events can be captured to add interactivity to the SVG elements by using a script to register functions for handling specific events. Events can for example trigger transformations of the SVG element (e.g. for animations) or the element styling to change. It is also possible to create hyperlinks to external documents in an SVG element. [49]

To view SVG in a web browser, an SVG viewer is needed. The Mozilla Firefox, Apple Safari and Google Chrome web browsers currently has native SVG support, although IE8 does not. However, there are SVG plugins that can be installed to a web browser to gain SVG support. One such plugin is the Adobe SVG Viewer, which is available for IE8, thus enabling also IE8 to view SVG. [50]

3.5.2 Adobe Flash vector graphics support

Vector graphics and animations are integral parts of Flash and surely one key aspect of the success of Flash on the web. Developers can draw vector graphics in Flash by using the ActionScript drawing API which provides a number of methods for drawing vector shapes such as moveTo(x, y), lineTo(x, y) or curveTo(x1, y1, x2, y2). Vector graphics can be drawn directly on graphics container objects such as Shape, Sprite or MovieClip by manipulating their contained graphics property, which is an instantiated object of the Graphics class. [43] Below is a small example drawing a rectangle in a MovieClip container:

```
_root.createEmptyMovieClip("myRect",1);
myRect.moveTo(50,50); // Position marker at start point
myRect.lineTo(100,50);
myRect.lineTo(100,100);
myRect.lineTo(50,100);
myRect.lineTo(50,50);
```

Vector graphics can also be expressed in the markup language MXML Graphics available in the Flex SDK, which is used to mark up elements that are tied to specific ActionScript graphics classes similarly to SVG. The MXML vector elements can thusly be accessed directly from ActionScript. [53] This is similar to how scripts can access SVG elements in HTML. Below is an example drawing a rectangle in MXML similar to the one in the above ActionScript example:

```
<s:Graphic>
  <s:Rect id=" myRect" x=”50” y=”50” width="50" height="50">
  </s:Rect>
</s:Graphic>
```

In the above example the <s:Graphic> element is backed by the ActionScript class spark.primitives.Graphic and the <s:Rect> element is backed by the class spark.primitives.Rect which makes it possible to add ActionScript code to the elements.

However, there is no native support in Flash for displaying the XML-based vector graphics formats SVG or GML. There are some third-party products for converting SVG documents into SWF files, but there is no native support in Flash for transforming SVG into SWF at run-time. It should in principle be possible for developers to parse and analyze an SVG or GML document and produce a sequence of ActionScript drawing function calls or create an MXML structure from the same data. Google has for example developed a Flash application that renders SVG structures called SVG Web. [51] Unfortunately, since MXML is a proprietary standard, no standardized procedures for transforming XML-based formats into well-formed MXML have been found during this thesis work.

To improve vector graphics data interchangeability, Adobe has created an XML-based graphics interchange format called Flash XML Graphics (FXG) for the Flash platform and released a public specification of the format. [52] It should be fully possible to transform SVG and GML documents into well-formed FXG format using XSLT. The FXG document can then utilize the rendering capabilities of Flash Player 10 to visualize the vector graphics.

FXG elements could originally not be referenced or accessed by ActionScript code since they do not have backing ActionScript classes like MXML elements do. However, with Flex 4, a new skinning and component architecture called Spark was introduced. FXG elements can be added to Spark Group containers, and it is thus possible to indirectly reference FXG elements from ActionScript code through the Spark Group containers, enabling interactivity with FXG elements.

3.5.3 Microsoft Silverlight vector graphics support

Silverlight provides a Shape class that is a super-class for graphical shapes. It has sub-classes such as Line, Rectangle and Ellipse. These can be contained in instances of the Panel class, such as Grid or Canvas which are used for defining the layout of UI elements. Graphics can be marked up in XAML to create layout and tie the elements to their backing classes. [54] It is in principle very similar to MXML in Flash as described above and will not be explained further in this section. Here is an example of drawing a rectangle similar to the example for MXML above:

```
<Canvas>
  <Rectangle Name=”myRect” Canvas.Left="20" Canvas.Top="20" Width="100"   Height="100" />
</Canvas>
```

Like Flash, Silverlight does not support SVG or other vector graphics markup languages. There are third-party tools for converting SVG documents to XAML, but there is no standard for transformation. Consequently, there is no standard method for transforming GML documents to XAML. It seems that there are at present no tools or libraries available for converting SVG to XAML at run-time in Silverlight.

3.5.4 HTML5 vector graphics support

In the current version of the HTML5 specification, a new SVG element has been introduced for embedding SVG content into an HTML document. This means that all web browsers that fully conform to the HTML5 standard – or even only support the SVG element of HTML5 – will have native support for the SVG format.

The HTML5 specification also introduces the new canvas element for dynamically drawing graphics which may be in both bitmap and vector format. Vector graphics can be dynamically drawn on the canvas element by using JavaScript functions, such as rect(x, y, width, height) for drawing a rectangle. A canvas element is embedded into an HTML page by simply declaring a <canvas> tag. A small example drawing a rectangle is shown below:

```
…
<script type=”text/javascript”>
  function init() {
    var canvas = document.getElementById('myCanvas');
    draw = canvas.getContext('2d');
    draw.rect(10, 10, 150, 150);
  }
</script>
…
<body onload=”init();”>
  <canvas id=”myCanvas” width=”500” height=”500”></canvas>
</body>
…
```

Whether to use SVG or the canvas element for drawing vector graphics largely depends on the application. When mouse interaction with the vector elements is required, SVG is the best suited method, but when there is no mouse interactivity required and vector graphics are continuously rendered and updated, the canvas element can be used.

Furthermore, the new IE9 and Firefox 4 web browsers have also introduced hardware acceleration for 2D graphics, using DirectX video acceleration technology. [55] This means that the GPU of a computer hardware system can be utilized for rendering SVG content, thus relieving the CPU. It is not unreasonable to predict that similar solutions will be attempted in the future by competing web browser vendors.

3.5.5 Vector graphics in web mapping applications

There are a number of inherent characteristics of vector graphics that make vector formats attractive for use in web mapping applications. Traditionally, most web map servers have mainly provided raster images with their services. The geographic vector data have been integrated onto a static raster image on the server side before sending it to the client. Raster images are cheap to draw and they can in principle contain an unlimited number of points without straining the web browser when rendered. This contrasts with vector graphics, where each shape is expressed in mathematics and is rendered on the client-side. As a result, the rendering complexity increases as the number of points increases.

However, raster images are static, and in a sense "unintelligent". They simply consist of an array of pixels in different colors. Vector formats, on the other hand, allow interactivity. Each element on the map can be interacted with by the user. Vector elements can be directly transformed or manipulated using scripts and mathematical methods. Each element can be tied to some metadata describing different properties of the element, and computations can be made on this metadata. For example, it could be possible to calculate the distance between two elements, or to calculate the length of a line, the area of a surface, and so on.

Vector graphics also scale in resolution very well. Where raster images become blurry when zooming, vector graphics remain sharp. Since zooming in and out in a map is a common action in web mapping applications, this is another feature of vector formats that makes vector formats interesting for web mapping applications.

Since there are XML-based formats for describing and laying out vector graphics, it is ideal for interoperability and data interchange. GML is XML-based, and so are the FXG and SVG formats. By using technologies such as XSLT, it should be possible to transform GML documents into either one of the XAML, MXML or SVG formats.

One big issue with using vector representation of maps in web applications is the vector rendering capability of web browsers. Vector graphics support is not mandated by the HTML4 standard, although it is included in the HTML5 draft specification. This means that web browsers may or may not provide vector graphics support. Flash and Silverlight vector graphics are of course rendered by their corresponding plugins, enabling browsers with the corresponding plugin installed to properly display vector graphics.

As maps often contain a lot of geographic data with perhaps hundreds of thousands or even millions of points, there are often a lot of vector shapes and elements that need to be drawn by a client. A GetFeature request to an OGC WFS map server can result in a very large GML response document containing a lot of elements and points. In addition to transforming the GML document into an appropriate vector format for visualization, the

client needs to render a possibly great number of points, lines and curves. This may be costly in terms of rendering performance, and the performance largely depends on which vector renderer is used by the client.

3.6 Multimedia playback support

One of the key features of RIAs has been multimedia playback support. HTML web pages were initially intended mainly for displaying and navigating hyper-linked text documents. It was not intended as an application or multimedia platform. The requirements of web pages have been raised during the last decade, and the Internet now transfers a lot of multimedia and web applications are converging with desktop applications.

However, there is no native support for multimedia playback, i.e. video and audio, in HTML4. One solution to get around this limitation has been to introduce the plugin-based platforms, e.g. Flash and Silverlight, which have video and audio support. In the developing HTML5 standard there is also video and audio support, thus giving conforming browsers native support for presentation of video and audio.

3.6.1 Adobe Flash multimedia support

The Flash platform uses several file formats for video containers. The most common are the Flash Video (FLV) and the SWF formats. More recently, Adobe has introduced the F4V format, which removes some limitations of the FLV format. F4V is supported by Flash Player 10. Supported video codecs in Flash Player 10 include H.264, On2 VP6 and Sorenson Spark.

Flash Player 10 also has native support for several audio formats, such as ADPCM, HE-AAC (MPEG-4 Part 3) and MP3. [57]

3.6.2 Microsoft Silverlight multimedia support

In Silverlight, media is contained in objects of the MediaElement class. The MediaElement class supports the following video codecs regardless of the file format (i.e. container): YV12, WMV1, WMV2, WMV3, WMVA, WVC1, H.264 and raw video.

The MediaElement class also supports the following audio formats: MP3, AAC, WMA Standard, WMA Professional and WAV. [54]

3.6.3 HTML5 multimedia support

One of the most anticipated features of HTML5 has been the introduction of multimedia playback support. HTML5 have added the video and audio elements to the

standard. However, the current version of the HTML5 specification is far from final and one of the most debated issues is the choice of video codec. As a result, different web browsers currently support different codecs, and there is no single combination of video and audio codecs that currently works for all browsers. [58]

There are three main combinations of audio and video codecs that are supported by the most popular web browsers. These are the H.264 and Theora video codecs and the AAC and Vorbis audio codecs, as well as the WebM media container format which includes the video codec VP8 and the audio codec Vorbis.

To make a web application that is compatible with most web browsers, it is currently necessary to provide support for each of these formats and codecs. It is unclear which formats will be supported in future web browsers and the discussion of which, if any, formats should be mandatory in the HTML5 specification is still ongoing. [58]

The issue of video codec choice involves patent issues. The popular web video format H.264 is a proprietary codec and is patent-encumbered. This arguably makes it inappropriate for an open web standard such as HTML5. The WebM container format is patent-free and open, and was developed by Google to serve as an alternative to H.264 with comparable performance and quality. [77] That has made it a candidate as the HTML5 multimedia container format.

4 Web mapping on RIA platforms

This chapter describes some of the technology used by some current web mapping RIAs and also discusses RIA-related features and concepts that are relevant in the context of web mapping. A distinction has been made between JavaScript/Ajax-based RIAs and plugin-based RIAs.

4.1 Web mapping on JavaScript-based RIA platforms

Many web mapping applications use a JavaScript/Ajax-based approach to creating web mapping services. Ajax-related techniques such as pre-fetching of data and asynchronous RPC can be successfully applied to web mapping to create a more dynamic and rich user experience. The concept of Web Services is also becoming increasingly popular in web mapping. There are a number of different Web Services and open APIs available for web mapping applications, such as the Google Maps API and the Bing Maps API.

4.1.1 Typical implementations of JavaScript/Ajax-based web mapping applications

As JavaScript is such a flexible and dynamic language with few uniformly adopted standards, there is a wide variety of possible architectures for web mapping applications. However, there are some techniques and concepts that are commonly shared among popular web mapping applications. Some of those are described below.

4.1.1.1 Web Services architecture in mapping applications

Many web mapping services use the service-oriented approach by calling a number of different distributed Web services. [37] The OGC are developing a common architecture for applying the Web services approach to web mapping. [38] This is an attempt to adapt OGC services to the Web Services architecture, in order to gain the benefits of interoperability, distribution and a common architectural approach to web mapping applications.

4.1.1.2 Ajax-based image tiling

Most web mapping applications use a technique called image tiling. This is done by creating a grid of partial map images and tiling them together. Each tile makes up a uniformly sized rectangle containing an image. The map images are fetched from one or many different GIS map servers and added to the grid asynchronously and dynamically using Ajax techniques. When the user requests to view a map of a certain area, the images required for this specific area are displayed in the grid.

However, the adjacent image tiles are also asynchronously pre-loaded in the background. If the user then wants to scroll the map in any direction, the images of the requested area are already loaded and can be readily displayed. Consequently, the image tiles adjacent to the new actively displayed map area are loaded, and so on.

Similarly, a map can be divided into different tiling layers corresponding to different levels. The tiles are then indexed to create a tree-structure. Each tile thus has a sub-layer of tiles, corresponding to a lower level of the map. This allows a user to zoom to different levels of the map. Zooming in on a subset of the displayed tiles then loads those specific tiles' sub-levels of higher-resolution tiles, if there are any available. Corresponding to tile pre-loading, tile layers adjacent to the displayed layer may also be pre-loaded by using Ajax techniques to allow for a more seamless transition between different map levels.

4.1.2 XML parsing and transformation in JavaScript

Different web browsers have different built-in XML parsers. ECMAScript for XML (E4X) is an extension to the ECMAScript to provide native support for XML data. [39] By using E4X, JavaScript gain native support for XML data and can thus be used for parsing. It also extends JavaScript with certain operators that are useful when working with XML. However, Firefox is currently the only web browser with a wide support of E4X.

Most browsers use the DOM as an interface for manipulating XML data. In the DOM, XML data is represented as a tree. To read and write to the XML document, the application uses the tree to traverse and manipulate the nodes of the tree which represent the XML elements. The XML DOM tree is loaded into memory to allow for navigating up and down the tree. Hence, using the DOM approach to XML parsing is suitable in cases where the user needs to re-visit elements and repeatedly browse the XML document. However, if the XML document contains a lot of data, loading the document into memory might lead to performance issues.

4.1.3 Survey of JavaScript mapping APIs and applications

There are a number of JavaScript web mapping APIs available. Some of them are provided by enterprises, but there are also some open source APIs. Some of the APIs are described below.

4.1.3.1 Google Maps

The Google Maps JavaScript API [40] makes it possible to embed data from Google Maps in a JavaScript client. The Google Maps provide a standard UI that can be disabled and customized by either loading specific pre-defined controls from the Google Maps API or by defining customized controls. It is compatible with the most popular web browsers. [40]

There are three main viewing modes that can be used; the Map mode which provides a topographic street map, the Satellite mode which provides high-resolution aerial photographs and the Hybrid mode which overlays a street map over aerial photographs. The hybrid view could previously be accessed in the Google Maps web application by clicking a special button, but now it is accessible by checking the "Show labels" checkbox under the Satellite view button. It is turned on by default.

During a period, there was also a "Terrain" viewing mode that was accessed by clicking a button. The feature has now been moved and can be found in the "More" dropdown menu while browsing in the Map mode. The terrain view contains information about the physical landscape in the map.

Google Maps has also recently introduced a bird's eye view perspective, which means that the satellite images are tilted in a 45 degree angle to provide a three-dimensional-like effect. The feature is currently only available in a few cities in Google Maps and appears to the user when the user zooms in on an area where such images are available.

Additionally, Google has added an "Earth" button which lets the user use Google Earth functionality in the Google Maps web application. However, to use the Google Earth desktop application functionality in the web browser, a plugin needs to be installed and it does not use JavaScript technology.

It is also possible to add overlays such as markers, polygons or pop-up windows called InfoWindows. Google Maps also support their own XML-based vector file format, the Keyhole Markup Language (KML) format, which can be overlaid on maps using what is called GeoXML. The API also provides services for geocoding and routing and driving directions management.

4.1.3.2 Bing Maps

Microsoft's Bing Maps, formerly known as Virtual Earth, provides a JavaScript mapping API called Bing Maps AJAX Control. [41] The API is available in two versions: the core functionality version and the full functionality version.

There are three main viewing modes in Bing Maps. There is a Road mode which is similar to the Map mode in Google Maps, the Aerial which is similar to the Satellite mode in Google Maps and the Birdseye which was introduced before and is similar to the Google Maps bird's eye view. The Birdseye view is only available in certain areas of the earth and provide angled, high-resolution aerial images.

The Aerial and Birdseye viewing modes also have the additional AerialWithLabels and BirdseyeWithLabels modes, which add geographic information to the Aerial and Birdseye views respectively. This is similar to the Google Maps hybrid mode.

Just like Google Maps, Bing Maps also has a 3D mode that incorporates Virtual Earth-like functionality to the web mapping application. Just like the Google Maps' "Earth" mode, it also needs a browser plugin to run.

Bing Maps provides functionality for adding shapes to maps for use as markers, highlighting geographical shapes on the map and overlay information on the map. The shapes can be polygons, polylines or pushpins. Bing Maps also provides functionality for routing and driving directions.

4.2 Web mapping on plugin-based RIA platforms

Traditionally, most web mapping applications have been based on JavaScript/Ajax technology. Basically, they have used asynchronous loading of pre-rendered raster images for presenting maps and to let users navigate. So far, plugin-based web mapping applications have not been as common. This may perhaps change in the future, as e.g. Microsoft has released a Silverlight version of their Bing Maps application and API.

4.2.1 XML parsing and transformation on plugin-based RIA platforms

Both Silverlight and Flash have the ability to construct and render vector graphics – in fact it is one of the key features of both platforms. However, since Flash and Silverlight use the MXML and XAML markup languages to construct vectors, the GML data retrieved from an OGC WFS server must be parsed and transformed into an appropriate XML format for representation.

Neither Silverlight nor Flash has native support for XSLT that can be used to transform GML data into XAML or MXML. An alternative solution is to perform the XSLT transformation in an external JavaScript module and load the output data into a Silverlight or Flash module.

4.2.1.1 XML in Adobe Flash

ActionScript 3 has an XML class in the flash.xml library that can be used for handling XML data according to the E4X specification [39]. The XML class has a set of methods for loading and traversing XML documents. As the XML document is loaded into Flash by using the load function, it is converted into an AS3 object. The resulting object has a tree structure, which can be used to access the XML elements and parameters. XML documents can also be created with the XML class by using methods such as createElement

and appendChild. Element attributes are set by directly setting the attributes field of the nodes.

To convert from one XML format to another in Flash, one can load and traverse the original XML document, analyze it and then manually create a new XML document of the target format by using the standard AS3 methods.

4.2.1.2 XML in Microsoft Silverlight

In Silverlight, XML data can be parsed and processed by using a set of classes in the System.Xml namespace. The two main classes in the library are the XmlReader and the XmlWriter classes for parsing and writing XML documents, respectively. XmlReader conforms to the XML 1.0 recommendation and is a non-caching forward-only parser. XmlReader contains several different methods for reading XML element data and attributes.

A combination of XmlReader and XmlWriter objects can be used to parse a GML document, and create a new XAML document representing the GML data. The transformation logic must then be created by using methods of the XmlReader and XmlWriter classes as well as potentially other Silverlight-native library methods.

4.2.2 Survey of plugin-based mapping APIs, development tools and applications

Plugin-based mapping applications have not yet been as common or popular as JavaScript/Ajax-based mapping applications. This is most likely because the largest web mapping service providers such as Google, Yahoo and Microsoft have used mostly JavaScript/Ajax-based technology in their services. However, Microsoft has recently released their Bing Maps Silverlight API and a Silverlight version of their application Bing Maps which is now the standard version. Plugin-based applications and APIs may therefore become increasingly used in the future.

4.2.2.1 Adobe Flash

There are a few proprietary development tools for Flash mapping, such as FusionMaps, FlashMapOne and amMap. [62] [63] [65] These tools are typically integrated tool sets for creating dynamic maps in Flash.

Google has developed a Flash API for their Google Maps service as an ActionScript alternative to their JavaScript API. [64] This lets developers integrate Google Maps with their Flash applications. The Google Maps API for Flash includes most functionality of the JavaScript API but also adds functions to add Flash content onto Google Maps.

Yahoo! has also developed a Flash API for their Yahoo! Maps service. [66] It is used by adding their Yahoo! Maps AS3 Component to an application, which lets developers integrate Yahoo! Maps services and create map applications that utilize the Yahoo! Maps engine.

4.2.2.2 Microsoft Silverlight

Microsoft has recently released their Bing Maps Silverlight Control which is an API for Bing Maps in Silverlight. [66] They have also released a Silverlight version of their Bing Maps application, assumedly in an attempt to gain a competitive advantage over mainly Google Maps, as well as to drive the installation of Silverlight runtime plugins. No other published APIs for Silverlight map development have been found during this study.

5 Comparison of HTML5 and plugin-based RIA platforms in a web mapping context

The preceding chapters have introduced and described JavaScript/Ajax-based web technologies and platforms as well as plugin-based web technologies and platforms in both a general and a mapping context. The following chapter summarizes and compares the two approaches. The purpose is to elicit the key differences in functionality, performance and development environment and thus be able to select the most suitable approach to develop a rich web map client prototype.

5.1 Comparison of Silverlight/Flash and HTML5 performance

This study has selected two key performance parameters for comparison that have been proposed especially vital to web mapping applications: the performance of parsing large GML documents and transforming them into a suitable vector graphics format for presentation, and the performance of rendering large sets of vector graphics data.

5.1.1 General performance of Flash, Silverlight and JavaScript

This section describes some general aspects of RIA performance and discusses some benchmark test results that have been found during the study. Benchmark results should only be regarded as an indication of performance of some particular implementations running on some particular platforms.

JavaScript is a dynamically typed language. This means that type security checking is deferred to run-time instead of compile-time. Although this has advantages over statically typed languages in certain aspects such as flexibility and potentially lowered development time, it is generally considered to perform slower in terms of execution time and memory usage. [34] [36] Fields and methods can be changed in an object and also its parents. Such flexibility makes it difficult to perform compiler optimization of JavaScript code. However, there are techniques being implemented to help increase compiler performance, such as Just-In-Time (JIT) compiling. [35]

Different web browsers use different JavaScript engines for interpreting JavaScript, each with their own approaches to performance optimization. The performance of a JavaScript application thus depends on the web browser it is run on. Google Chrome, for example, uses a JavaScript engine called V8 that has been found to perform distinctly better than Microsoft's Internet Explorer 8 (IE8) browser and also better than Mozilla's Firefox 3.6 browser in tests using Apple's SunSpider and Google's V8 Test Suite benchmarking tools. Apple's Safari has also been found to be considerably faster than Internet Explorer and Firefox, although slightly slower than Chrome. [36]

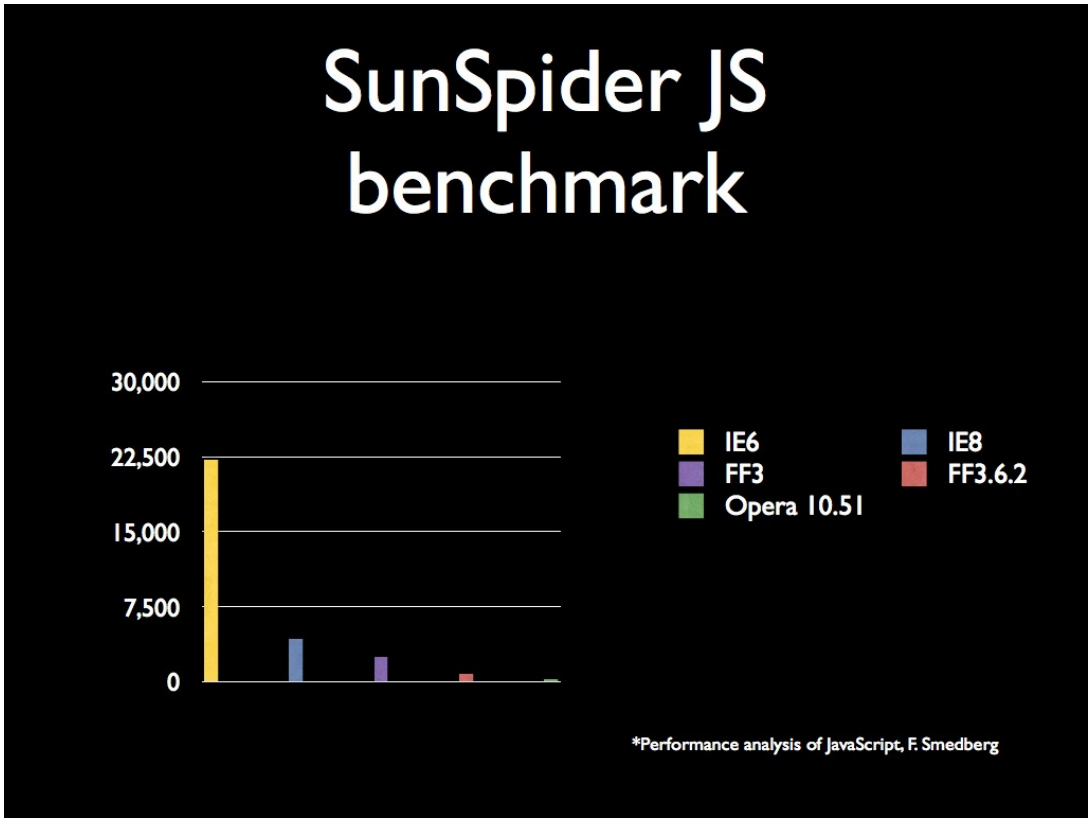

Illustration 1: Sunspider benchmark results.

Even though performance is varying between different JavaScript engines, it can be concluded that JavaScript performance has greatly improved over the last decade in all of the above mentioned web browsers. In a comparative test using SunSpider, the performance of the IE engine has increased with a factor of five, IE6 of 2001 achieving a test result of 22237,4 milliseconds while IE8 achieved a result of 4345,2 milliseconds as reported by [36]. In the same test, the Firefox 3 browser of 2008 achieved a result of 2567.6 milliseconds while the Firefox 3.6.2 of 2010 achieved a result of 839 milliseconds. The Opera 10.51 browser achieved a result of 353.8 milliseconds. The upcoming Firefox 4 and IE9 releases are expected to further narrow the gap between browser JavaScript engines performance.

A comparative study of RIA performance in [56] using the Focus-test benchmark tool suggests that there is no big difference in performance of JavaScript, Flash and Silverlight on most platforms, although the benchmark of JavaScript running in Opera 10.10 and IE8 generally showed relatively very poor results compared to Flash, Silverlight or JavaScript on other platforms.

5.1.2 Rendering performance of vector representation of GML data

GML responses to a GetFeature request to an OGC WFS server may contain a vast quantity of elements and points that need to be rendered after being converted into a vector graphics representation format. This means that vector rendering performance may play a significant part in the overall performance of a vector-based mapping application. Different web browsers have different vector graphics rendering engines, and the performance even varies within a browser on different operating systems.

The rendering performance in plugin-based technologies such as Flash and Silverlight should in principle be platform-independent since the rendering is done by the plugin. However, in practice the rendering performance can vary also in Silverlight and Flash depending on which platform they run on. However, some benchmark tests have shown that the performance varies slightly over different platforms, although the variation is not as large as in JavaScript implementations on different platforms. [56]

5.1.2.1 Performance of vector drawing using HTML5 Canvas and using the ActionScript drawing API

When performing a stress test of vector rendering by using the GUIMark2 benchmark test developed by Sean Christmann in [55], results show that rendering of vector graphics in HTML5 and Flash 10 is performed in an average frame rate almost twice as high in Flash 10 (29.15 frames per second in average) as in HTML5 (15.64 frames per second) on Windows 7, and almost four times as high in Flash 10 (21.29 frames per second) as HTML5 (5.53 frames per second) on Mac OS X Snow Leopard. In this test, the HTML5 version uses the canvas element and JavaScript to dynamically render vector graphics. In the Flash version, ActionScript has been used to draw vector graphics in a Sprite container. Marked-up vector formats such as SVG or MXML have not been tested. The results of the benchmark test in [55] are summarized in the table below but should not be regarded as a scientific study – it is shown here as an indication of a case study of the frame rate of rendering vector graphics on different platforms:

	HTML5	Flash
Windows 7		
Internet Explorer 8.0.7600	N/A	30.7
Firefox 3.6.3	15.73	29.65
Chrome 4.1.249	6.41	26
Opera 10.53	24.77	29.9
Safari 4.0.5	Not applicable	29.5
	Avg 15.64 fps	Avg 29.15 fps
Snow Leopard		
Safari 4.0.5	4.04	20.55

Firefox 3.6.3	3	23.92
Chrome 5.0.342	2.86	25.48
Opera 10.10	12.22	15.24
	Avg 5.53 fps	Avg 21.29 fps

For more details of this particular test, refer to [55].

5.1.2.2 Performance of rendering SVG and XAML marked-up vector graphics

During this thesis work, no scientific studies were found comparing the performance of rendering SVG elements in HTML and rendering XAML or MXML markups of vector graphics in Silverlight or Flash respectively. However, one benchmark test was found that compares the performance of drawing vector graphics using the canvas JavaScript drawing API and using JavaScript to draw SVG elements. [72] The benchmark results indicate that the canvas drawing API is consistently somewhat faster than SVG.

The study in [73] analyses the scalability of visualization methods using SVG, Canvas and Java implementations of parallel coordinates and squarified treemaps. The study concludes that SVG has adequate interactivity performance in small to medium sized data sets. SVG is also considered capable of rendering very large data sets if redundant elements in paths can be eliminated and other optimizations made.

5.1.3 Performance of the Bubblemark 2D animation benchmark

Bubblemark is a simple open-source benchmark tool for comparing the performance of a simple 2D animation test implemented with different web technologies such as DHTML, Silverlight (CLR) and Flash (Flex). [74]

When the test was run on some of the latest versions of the most popular web browsers, the results summarized in the table below were achieved. The three values for each platform refers to the framerate (frames per second) when running the test with 32, 64 and 128 balls respectively. The test was run on Windows XP.

	Silverlight 3.0 (CLR, CacheMode="BitmapCache")	**Flash (Flex cacheAsBitmap=true)**	**DHTML**
Firefox 4 beta 6	76, 73, 71	56, 56, 56	100, 100, 99
Opera 10.63	76, 74, 69	52, 52, 52	501, 246, 115
Chrome 6	69, 69, 65	52, 52, 56	176, 195, 148
Safari 5.0.2	72, 70, 67	52, 52, 55	85, 85, 69

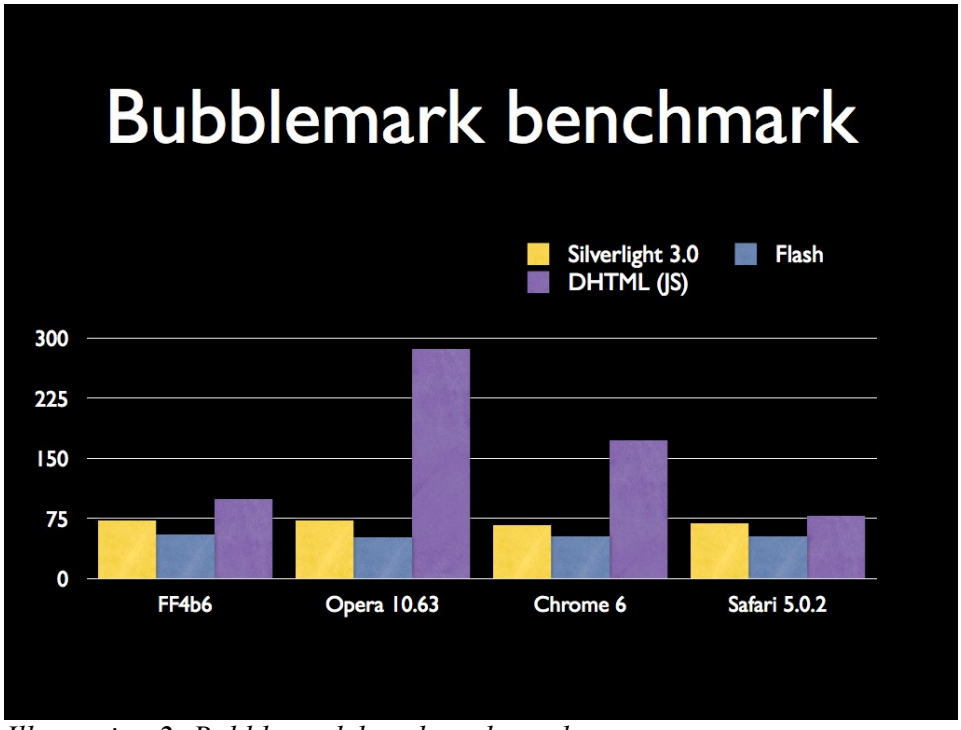

Illustration 2: Bubblemark benchmark results.

The test described above is not a scientific study and should only be regarded as an approximation. However, there are some tendencies that may be discerned. The DHTML implementation of the test produces significantly better on all web browsers but also has the most cross-browser variation. The Flash implementation of the test consistently produces the lowest framerate results across all browsers but shows a very small cross-browser variation and also produces the same framerate results regardless of the number of balls.

5.1.4 Hardware acceleration in Silverlight, Flash and HTML5

Hardware acceleration of web content rendering usually refers to using the GPU of the computer system to render the graphical contents of a web site, e.g. video, animations or images.

Hardware acceleration techniques are supported on both the Flash and the Silverlight platform. Flash Player 10.1 can utilize hardware acceleration when decoding and rendering H.264 and AAC media on OpenGL-supported platforms. Silverlight supports hardware acceleration using Direct2D on Windows platforms and OpenGL on Mac platforms for rendering animations and video.

To enable hardware acceleration, a parameter must be set in the embedded Flash or Silverlight application. Once the parameter is set to true, the plugin will utilize hardware acceleration for video rendering. Additionally, applications can also utilize a technique called bitmap caching. With bitmap caching, a vector graphics object can be transformed into a static bitmap image, thus lowering performance costs.

Hardware acceleration is not a part of the HTML5 specification. However, several web browsers are adding hardware acceleration support for rendering video content and animations. The beta version of Firefox 4 enables hardware acceleration by using Direct2D on DirectX 10-supported hardware. [78] The Internet Explorer 9 has also introduced hardware acceleration using the DirectX technology. [79]

In [80] a test was made to compare CPU utilization when playing YouTube video in Flash Player 10.1 and HTML5 on different web browsers and operating systems. The results of that test suggest that:

- On the Mac OS X platform, Flash Player 10.1 utilizes considerably more (159%) CPU power than HTML5 on the Safari browser and utilizes the CPU power equally (0%) to HTML5 on the Chrome browser.
- On the Windows platform, Flash Player 10.1 utilizes considerably less (-58%) CPU power than HTML5 on the Chrome browser. Other browsers did not support HTML5 video playback on the Windows platform.

5.1.5 Conclusions of Flash, Silverlight and JavaScript performance comparison

It is very hard to draw any definite conclusions on the relative performance of Flash, Silverlight and JavaScript. The benchmark tests discussed above are all dependent of particular implementations using the different technologies and must therefore only be considered as indications of relative performance.

Furthermore, the found test results are greatly varying over different platforms and different applications. Combinations of web browsers and operating systems constitute different platforms showing different test results. Each of the three technologies shows performance advantages in some particular tests.

The conclusion drawn from the above discussion is that the relative performance of the three technology platforms is not a completely decisive factor in the choice of platform for developing the rich web mapping application prototype. Test results do not consistently favor any particular technology over the others. However, the general performance of JavaScript engines in web browsers has shown a relatively big improvement during the last couple of years.

5.2 Silverlight, Flash and HTML5 web browser compatibility and standards

Both Microsoft Silverlight and Adobe Flash are proprietary technologies that require a web browser plugin to run applications. Neither technology is native to the most popular web browsers today. However, the Flash Player plugin has become vastly popular, and has a penetration rate of 96.59% according to [22] and is thusly almost ubiquitous. Silverlight is younger on the market but has already reached a 54.56% market penetration rate. [22] This number can be assumed to continue to increase in the following years.

Flash Player 10 is currently available for the following web browsers [59] [61]:

- Internet Explorer and other browsers supporting Internet Explorer ActiveX controls and plugins (Windows)
- AOL (Windows)
- Firefox (Windows, Mac OS X, Solaris and Linux)
- Netscape (Windows)
- Opera (Windows, Mac OS X)
- Safari (Windows, Mac OS X)
- Google Chrome (Windows, Mac OS X and Linux)
- SeaMonkey (Linux)

The Silverlight runtime is as of today (19 October 2010) currently available for the following web browsers [60]:

- Internet Explorer (Windows)
- Firefox (Windows, Mac OS X)
- SeaMonkey (Windows)
- Safari (Windows, Mac OS X)
- Google Chrome (Windows)

There is also a third-party plugin called Moonlight available for Linux platforms that works as a Silverlight runtime.

In contrast, HTML5 is an open standard that each web browser may or may not implement partially or completely. Different web browsers may support different aspects of the standard, and leave others still unsupported. Therefore the question of whether a web browser supports HTML5 is perhaps incorrectly stated, as there are several elements of HTML5 that may or may not be supported. The level of conformity to HTML5 in web browsers today still varies greatly and HTML5 will perhaps not be fully implemented by most web browsers in at least a decade. Even then there might be variance in browser support, as is still the case today with HTML4.

The HTML5 standard is still many years from a finished proposed recommendation. The HTML5 editor Ian Hickson estimated in an interview [76] from 2008 that the proposed recommendation will be released in 2022. Even so, as is always the case with standards development, many web browsers already partially implement the specification and development of HTML5 implementations will not be stalled until the final release of the recommendation. In the HTML standardization process, pre-release implementations and proprietary quirks implementations change and drive the HTML standardization – web browser developers do not and arguably ought not to wait until a finished specification is released before starting to implement parts of it.

5.3 Brief assessment of Silverlight/Flash and JavaScript development environments and tools

Microsoft and Adobe provide advanced and integrated development platforms for developing Silverlight and Flash applications, respectively. Silverlight applications can be developed on the widely deployed .NET platform with tools like Visual Studio and Expression Blend and other tools. Flash applications can be developed with the Flash CS3 tool or with the Flex 3 platform and other tools.

As HTML is an open specification there are no official development tools, even though there are many freely available libraries and development tools available for JavaScript/Ajax development such as the NetBeans and Eclipse IDEs. Debugging JavaScript can also be done with web browser plugins such as FireBug for Firefox.

One of the most popular JavaScript libraries is jQuery which aims to simplify JavaScript development by providing functionality for document traversing, event handling, animation and Ajax interactions. [75] Modernizr is an MIT-licensed JavaScript library for detecting browser HTML5 support. [71]

One key factor for development in Flash or Silverlight and JavaScript is that developers can assume that all the web browsers will support all the functionality of the platform, as long as they have the plugin installed. That assumption cannot be made for JavaScript, as different web browsers will have different support for JavaScript, HTML and the DOM. This means that developers may need to include a lot of fallback functions and adapt the application to several different platform scenarios. A web application developer might be in a situation where he or she needs to develop several implementations of the application to ensure compatibility for different versions of HTML, DOM and CSS.

The lack of advanced, integrated and enterprise-supported development and debugging tools and platforms for JavaScript/HTML/CSS development and the extremely heterogeneous deployment environment of different web browsers brings extraordinary challenges to application developers.

5.4 Conclusion of HTML5 and plugin-based RIA comparison

As discussed above, the overall performance of web browsers' JavaScript engines has improved greatly over the last few years. Web browser developers are continuing to push the performance optimization in a very high pace. The differences in performance between plugin-based technologies and JavaScript/HTML technology has narrowed and in most tests discussed above the performance of JavaScript implementations has been either better or comparable to the performance of plugin implementations.

It is also concluded that with the introduction of the SVG and Canvas elements in HTML5, the vector rendering capabilities and performance of JavaScript/HTML technology is either comparable or better than that of plugin-based technologies. Some web browsers are also beginning to introduce hardware acceleration for video, image or vector rendering, which has earlier been exclusive for plugin-based technologies.

From the above discussion, it is also concluded that the development environments and tools for plugin-based technologies are still a lot more integrated, streamlined and rich than for JavaScript/HTML technology, although different developers may have different preferences and therefore prefer either development environment or toolset.

6 Rationale for selection of prototype technology platform

For the web mapping application prototype implementation developed in this thesis project, the HTML5/JavaScript technology platform has been selected as most suitable. The motivations for this choice are summarized in the following list:

- The performance of JavaScript has come so close to or even surpassed the performance of plugin-based technologies in many web browsers that the performance aspect has not been deemed a decisive factor in the choice of technology platform.
- Carmenta currently has a web mapping client that is based on HTML and JavaScript. Choosing the HTML5/JavaScript technology platform for the prototype development means that the prototype can be built on the current Carmenta client, which thus provides many functions such as image tiling and server communication that would otherwise have to be implemented from scratch.
- By building the prototype from the current Carmenta client, an evaluation can be made that very clearly compares the current Carmenta client and the developed prototype, both in terms of functionality and performance.
- HTML5 is an open standard specification. That means that the technology is not dependent of some private enterprise that controls the technology. It also means that the specification can be thoroughly accessed and analyzed. The openness of the technology has also been deemed more academically interesting than a closed technology.

7 Implementation of an HTML5/Javascript web mapping application prototype

The following chapter describes the implementation of a web mapping application prototype using HTML5 technology and related RIA technology. It also presents an analysis of the prototype. The prototype was built by extending the current Carmenta Rich Web Client (RWC), which is described further in the next sub-chapter.

The main concepts that have been used to increase the user experience and interactivity of the Carmenta RWC are the following:

SVG support has been incorporated to allow for the users to interact with the map elements in a richer way than in the current situation with the RWC, which uses static raster images.

Video support has been added through the HTML5 video element to give the user a richer visual experience. The video feature of this prototype serves as a proof of concept of how future implementations can use geo-tagged video content in a format native to the HTML standard, with no plugins or enterprise technologies required.

The caching capabilities of HTML5 through the localStorage and sessionStorage properties have been used to reduce the need of large client-server data transfers, thus improving the responsiveness and interactivity of the client.

Visually rich and interactive **user interface controls** have been added with the use of the jQuery library.

Each of these added features are described further below.

7.1 Carmenta Rich Web Client

The current Carmenta RWC is a JavaScript/HTML client that utilizes the Carmenta Server web map server to produce maps. The Carmenta Server is a map server that implements several different interface standards, such as the OGC WFS and WMS interfaces.

Internally, the Carmenta Server utilizes map configuration files created with the Carmenta Space Lab product. The map configurations contain information about geographic data sources, filter operations, visualization configurations among many other things. Certain attributes determines whether a specific map layer should be accessible to the RWC via the Carmenta Server interfaces and which interface to use for access, e.g. WFS or WMS, or other vendor-specific interfaces.

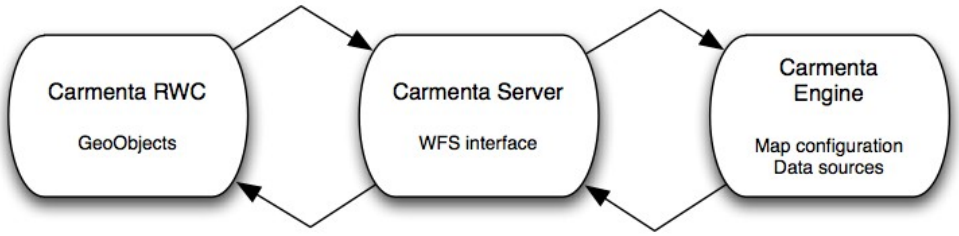

Illustration 3: Carmenta RWC architecture overview

The decision to build the rich web map client on top of the Carmenta RWC was based on the fact that the RWC already implements several key features that would otherwise have had to be implemented from scratch. Some of the key features of the Carmenta RWC that are utilized by the rich web map client prototype developed in this thesis are image tiling, server communication, GML parsing and GML feature encapsulation.

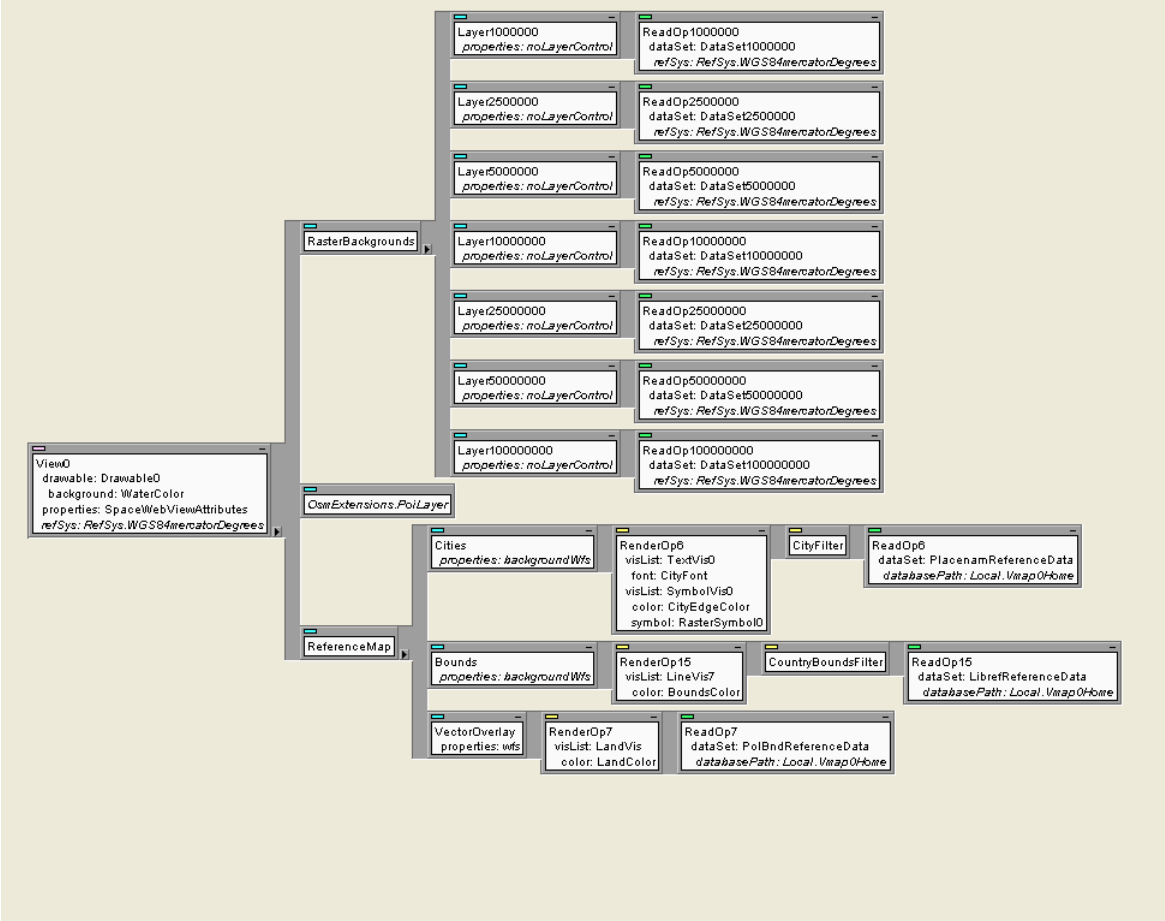

Illustration 4: SpaceLab map configuration

7.1.1 Geographic data retrieval, storage and visualization

The Carmenta RWC communicates with the Carmenta Server through XmlHttpRequest calls in typical Ajax fashion. It can access several different interfaces, but the main interface used in the rich web map client prototype is the OGC WFS interface for vector data retrieval. The Carmenta RWC contains functions for constructing WFS requests and it specifies a JavaScript-specific data container format called JSON as output format from the WFS server. JSON is often an ideal data transfer format in JavaScript clients since it is light-weight and native to JavaScript.

When the Carmenta RWC receives a server response containing geographic features, it constructs JavaScript objects called GeoObjects that encapsulate the GML feature data (which has been converted into JSON by the server). Each GeoObject constructed is then stored layer-wise in the Carmenta RWC and can then be drawn on the screen. When using the WFS interface, the GeoObjects are stored in a so-called

OverlayLayer, separate from raster data layers. Each layer corresponds to the specified typeName parameter of the WFS request.

There are different functions for drawing GeoObjects in the Carmenta RWC, and which drawing function to use is determined by the type of layer to be drawn. The layer types are specified in the map configuration file by setting an rwcType attribute of the layer. If a layer is of type 'WFS', it will be represented in the Carmenta RWC as a JavaScript object called OverlayLayer. , in contrast to raster layers which are represented as JavaScript objects called simply Layer.

In the current Carmenta RWC, WFS data is drawn either in a VML format or on the Canvas element in case of polygon or line geometries, or as images in case of point geometries.

7.2 Adding SVG support to the Carmenta RWC

Since the Canvas element does not support interactivity in terms of e.g. mouse events, it is not a suitable way to increase interactivity of the rich web map client. Unlike SVG elements, it is not possible to add event listeners to specific elements inside the Canvas element.

The VML format that is currently supported by the Carmenta RWC is very similar to SVG in structure and syntax. However, since VML is not an open standard and is not native to the HTML specification, SVG support was considered a necessary addition to the Canvas and VML drawing functions.

To give the prototype SVG support, a new drawing function was added in addition to the existing functions. The SVG drawing function is similar in structure and principle to the VML drawing function, but with some restructuring and of course, an adaption to the SVG syntax.

The SVG functionality is enabled and disabled by simply adding a boolean parameter to the RWC initialization function. If the passed argument is false or null, the client will perform as normal, without any of the added functionality.

When SVG functionality is enabled, one SVG layer is added to the map for each OverlayLayer that has been loaded from the map configuration, i.e. one SVG layer for each OverlayLayer available. When a GeoObject has been constructed and drawn using the SVG drawing function, the corresponding SVG element is added to the associated SVG layer container.

7.2.1 Adding interactivity to SVG elements

To be able to capture user interaction events, there is currently only support for one single SVG layer to be interactive at a time. This is because each SVG layer container covers the whole map area, thus covering all underlying layers. The layers are not added in a nested structure, but on top of each other, thus only enabling the topmost layer to capture mouse events. In case several SVG layers are used, only the topmost captures the mouse events. For this reason, the prototype has been tested and evaluated with only one SVG layer.

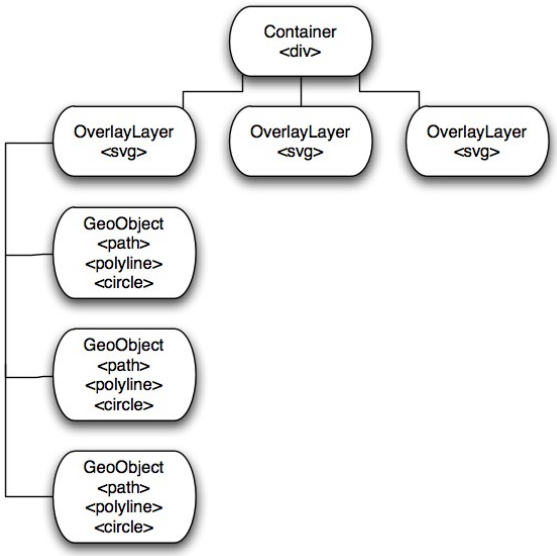

Illustration 5: The SVG layer structure

Event listeners are added to the SVG layer by hooking then onto each SVG element, corresponding to a GeoObject. For example, to handle mouse clicks on GeoObjects, a mouse click event listener with an associated callback function is added to the corresponding GeoObject in the SVG drawing function. In the prototype developed, the events handled are: mouse click, mouse over and mouse out.

In the developed prototype, a GeoObject that is hovered triggers an event handler that switches the fill-opacity of the SVG element from 0 to a value between 0 and 1, thus ”highlighting” the element. That is, in case the SVG layer consists of polygons representing countries, a hovered ”country” will be highlighted. When the mouse moves outside the country, the opacity is again set to 0, thus making the SVG element invisible (or rather 0% opaque).

If the user clicks an element, a popup-panel is displayed containing information and interactive functionality for that particular GeoObject. This popup-panel is described further in the next section and will be referred to as ”the popup-panel” in this report.

7.3 Interactive popup-panel

To allow for more detailed vector data associated with a GeoObject, a separate popup-panel was created that is displayed when a user clicks on a GeoObject in the main map view. The reason for this is that drawing all the vector data within each OverlayLayer on the whole map would require a potentially very large set of data to be drawn. This is not feasible for two reasons: the client-server transfer of the data would take much too long and sometimes even cause the server to overload, and the memory allocation required would be too large and the client computer could potentially run out of working memory.

On the other hand, for one single GeoObject, vector data from more than one layer can be feasibly drawn as long as the layers do not contain too much data. The amount of data each layer contains depends on the map configuration, and hence the map configuration must be properly set up as to not overload the server or to cause the client to allocate too much memory.

Each OverlayLayer can be configured not to be automatically loaded or displayed on the main map. Instead, a layer can be set up to stay in the background and be manually accessible via the WFS interface. In this way, they will not be drawn on the main map, but they can be retrieved manually and displayed in the popup-panel, which only displays data associated with one particular GeoObject, which will be referred to as the popup-panel's ”root GeoObject” hereafter.

The result is a popup-panel containing three tabs: one Layers tab, one Wiki tab and one Video tab:

The Layer tab contains a list of available WFS-enabled layers that can be displayed or hidden by checking or un-checking corresponding checkboxes. When a layer checkbox is clicked, a WFS request is sent to the server. As the server responds, each received GeoObject is drawn in a drawing function specific to the popup-panel, i.e. not the same drawing function as is used to draw GeoObjects on the main map. The reason for this is that each GeoObject is drawn in the popup-panel in one single SVG element (or layer). This allows for each GeoObject to be interactive.

As in the main map, the events handled are mouse over, mouse out and mouse clicks. Mouse over and mouse out events highlights or de-highlights the elements as in the main map, but mouse clicks lock the highlighting to the clicked element until a new one is clicked, thus allowing the user to keep the element highlighted when moving the mouse out or when browsing tabs.

The Wiki tab in this prototype simply embeds a wikipedia article based on a search for some of the clicked GeoObject's attributes. This solution is tailored for a particular map configuration where country names are used as a search string, as a proof of concept. This would have to be adapted to each specific map configuration used.

The Video tab has an embedded video element. In this prototype a static video reference for proof of concept. In a real application, one would need video data associated to each GeoObject. Another solution would be to perform a search for some of the GeoObject's attributes in some video resource and embed a search result in the panel.

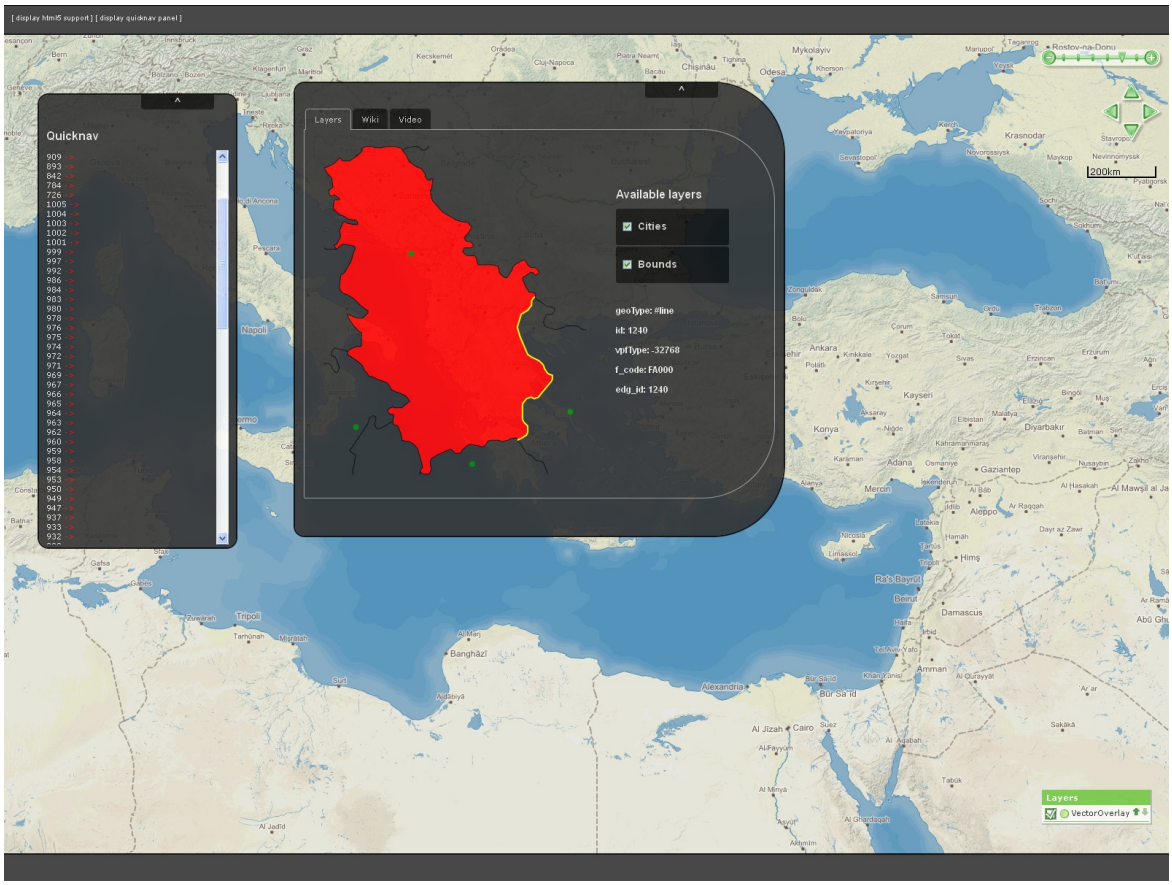

Illustration 6: The popup panel interface

7.4 Adding advanced user interface controls using jQuery

jQuery, which is introduced above, was used in the development of the prototype to gain access to some visual and interactive user interface controls. It was also used to gain

some other utilitarian JavaScript functionality, such as manipulation of elements' CSS attributes and DOM node selection.

Some of the jQuery functions used in the prototype are sliding animations, ability to drag and drop elements on the screen and also the tabs functionality in the popup-panel. This provides some visually rich additions to the user interface and gives a modern look and feel to the client.

7.5 Extending caching capabilities with HTML5 Web Storage

Early testing of the prototype quickly revealed that, as expected, the server-client communication and transfer of geographic data via Ajax calls is the biggest performance bottleneck of the application.

To reduce the number of requests sent to the server by the client, opportunities for caching data on the client's host machine were investigated. It quickly became evident that the most web browser-supported HTML5 client-side storage technique at this stage was the Web Storage feature, or more specifically the `localStorage` property of the DOM `window` element. The Web SQL Database and IndexedDB features are not yet widely supported by web browsers and are still in an early development phase.

The prototype utilizes Web Storage by storing the JSON-formatted server responses to the client's WFS requests sent from the popup-window. Each JSON response is associated to its corresponding GeoObject's layer and also to the root GeoObject displayed in the popup-panel. This way, if a user has already loaded a specific layer into the popup-panel for a specific root GeoObject (a ”country” in this particular prototype), the JSON-encapsulated data will be loaded from the client's computer instead of being fetched once again from the server. This allows for a much more responsive user interface and it also saves a lot of work on the server-side.

On the main map area, GeoObject's are fetched and drawn with SVG based on which bounding box is specified in the WFS request. There is no trivial method to determine which particular GeoObjects are contained within the bounding box prior to the request. This is in contrast with the popup-panel requests, where the root GeoObject and requested layer is already known and the request will therefore be identical for each specific root GeoObject.

Based on this discussion, it was determined that on the main map, caching will be performed on a per-bounding box level. That is, each JSON response is cached and associated with the corresponding WFS request's bounding box argument. If a request is then constructed that has the same bounding box as a previously requested bounding box, the JSON data is fetched from the client's machine instead of sending the request to the server.

The amount of space allocated to Web Storage caching is in many web browsers around 5 MB per default. If the available space is too small for caching a JSON response, an exception will be thrown and handled in the client by calling the `localStorage.clear()` function, which will completely empty the cache. A new attempt to cache the same JSON response will then be made.

7.6 Evaluation and testing of prototype

This section describes the results of the evaluation and testing of the prototype that was made at the final stage of development. It discusses both user-client interactions factors and client performance and responsiveness factors.

7.6.1 Conclusions of user-client interactions in prototype

The developed prototype introduces some fundamentally different interaction modes compared to the existing Carmenta RWC. By adding SVG elements to the map, the map becomes in a sense more ”intelligent”. The user can interact with specific elements that have inherent metadata that can be accessed.

In a traditional web map client, the maps consist of different layers of static images. The layers can in fact be very detailed and present a lot of visuals to the user, and the images can actually be interacted with. However, each layer does contain static images that are necessarily rectangular in shape and thus very hard to interleave. Using SVG elements instead, each geographic object on the map can be interacted with individually. One may say that the map itself becomes interactive, in contrast with a static map that contains one or several layers of images.

By utilizing modern JavaScript libraries such as jQuery, it is relatively effortless to develop a rich user interface with interactive user controls that can be minimized, dragged and dynamically turned on or off with a visually rich and animated presentation. This can lead to a more streamlined, rich and consistent user interface with a look and feel that can be easily tailored to specific needs and preferences.

7.6.2 Results of performance and responsiveness analysis

The performance of the developed prototype was tested mainly by using the Chrome Developer Tools that are included in Google’s Chrome web browser. The tool-set gives the abilities to profile a web page in terms of CPU utilization as well as viewing snapshots of the heap, view the storage used by the web page in terms of databases, localStorage and sessionStorage as well as a break-down of the time and space requirements to load the web page. This gives a view of how much time and space is utilized by different processes such as loading documents, running script, making XmlHttpRequest calls and rendering

images. The results include both the time to fetch the data from the server and to load it into the web page.

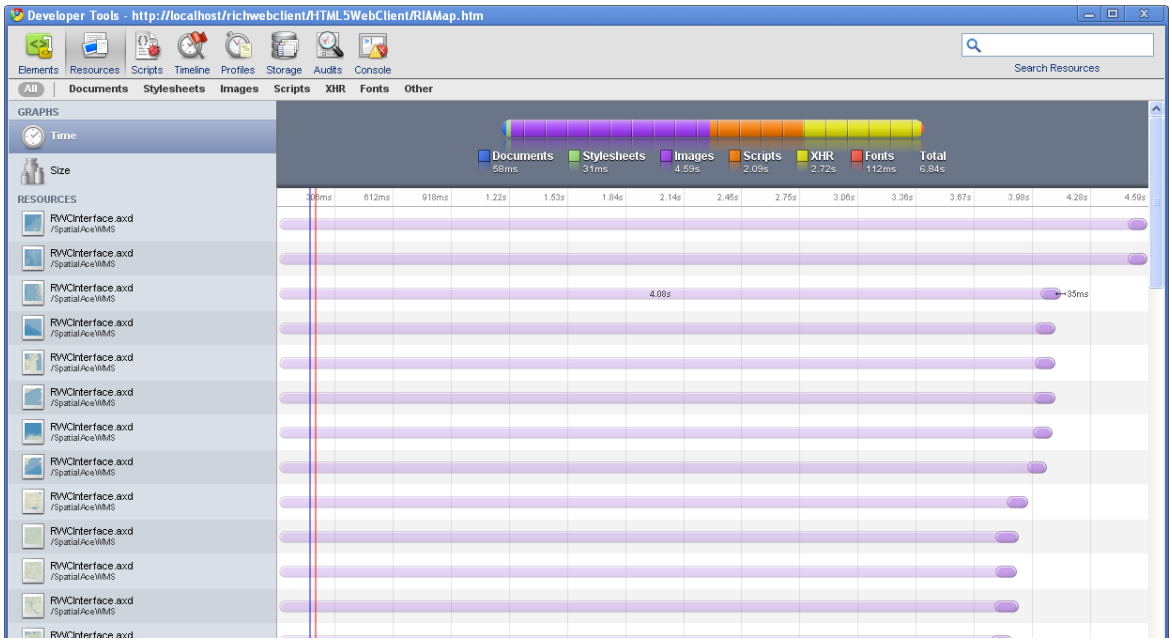

Illustration 7: Results of test run using Chrome Developer Tools with uncached geoObjects

The test run shown above displays the time taken to load the main view of the map configuration used in the developed prototype. The results are summarized in the table below.

Component	Time in msecs	Portion of total time
Documents	58	0.6%
Stylesheets	31	0.3%
Images	4590	47.8%
Scripts	2090	21.8%
XHR	2720	28.3%
Fonts	112	1.2%
All	9601	100%

A few clarifications of the above results are necessary. It can be seen that the time taken to load images (4590 milliseconds) also includes the fetching and rendering of all the map layers of raster images that are displayed. This also includes the XmlHttpRequest calls to fetch those images, and hence those calls are not included in the *XHR* category above. Instead, the *XHR* category includes the time taken to fetch the vector data for all

the vector layers on the map, and also the time taken to render the SVG elements. The *Scripts* category above includes the time to load and execute all the JavaScript on the page. It can be noted that loading the raster images is by far the most time-consuming step in loading the page. It takes almost twice as long as fetching and drawing the SVG elements.

When refreshing the same page without panning the map, the SVG elements have been cached in the localStorage and hence should not be fetched from the server again. Running the same test again shows a decrease in the XHR category time.

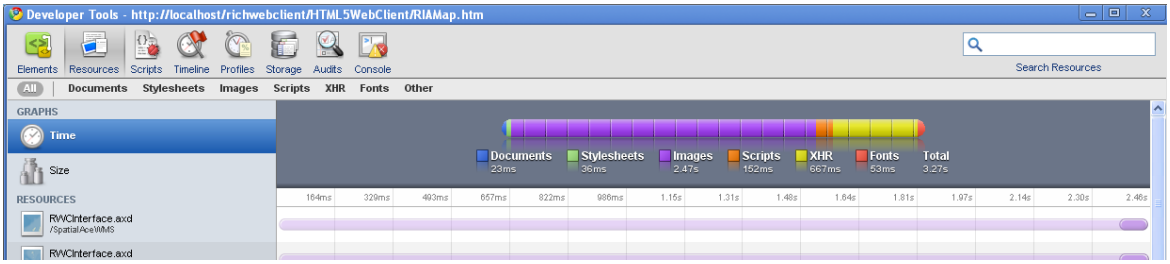

Illustration 8: Results of test run using Chrome Developer Tools with cached geoObjects

As shown above, the overall time to load the page on a refresh is lower, but the time taken for XHR is also relatively smaller, decreasing from 2720 milliseconds to 667 milliseconds. The Images category decreases from 4590 milliseconds to 2460 milliseconds. The time for fetching and rendering of the raster images become even more dominant when SVG elements are cached in localStorage. There is also a decrease in the Scripts category, which drops from 2090 milliseconds to 414 milliseconds. This is most probably partially because the SVG elements are not re-constructed in the script as they are cached.

Additionally, a test site was developed to compare the performance of rendering a specific map configuration using both the SVG and the Canvas drawing method of the client. The test simply calls the draw() function of each layer in the OverlayLayer objects that have been loaded to the maps, which in turn call either the drawUsingCanvas() or the drawUsingSVG() functions of each GeoObject loaded into each layer. The exact same map configuration is used for both drawing functions. This means that the same layers' draw() function are called for both maps.

The time to render all the layers of the map configuration is measured and reported back to the user. The testing revealed that the SVG drawing function was actually faster than the Canvas drawing function. The average time taken to render the map over 10 different test runs is given below.

7.6.3 Conclusions of performance testing

It can be concluded from the results discussed above that the biggest performance bottleneck is the loading of raster images to the map. This issue has not been targeted during the development of the prototype. It can also be concluded that the loading and rendering of SVG elements has a relatively low performance cost compared to the loading and rendering of raster images. When SVG elements are cached in localStorage the difference becomes even bigger, making the loading and rendering of the SVG layers approximately four times faster.

It can also be concluded that the SVG drawing function is considerably faster than the Canvas drawing function which was previously available in the Carmenta RWC, which gives the SVG paradigm further advantage in addition to the advantage of element-by-element mouse interaction capabilities.

8 Future work and development

The development of the prototype in this thesis work has focused mainly on the SVG and caching capabilities of HTML5. However, a certain map configuration specified with Carmenta's SpaceLab has been assumed. This means that if the map configuration was different, the web client may not behave similar to this specific prototype. A more generalized application could be developed to be more independent from specific map configuration or vendor technologies.

The map configuration used in the development of the prototype also has rather limited amounts of vector data that is used by the client. In a real situation the amount of data that needs to be transferred to and rendered in the client may be greater and may cause great performance loss. Thus, an investigation using greater amounts of data would need to be performed before deploying a similar client in a real-world situation.

To keep the performance of the client feasible with greater amounts of data, SVG rendering optimizations would probably need to be performed, as well as better caching algorithms to reduce the number of client-server requests. In the prototype developed in this work sometimes renders redundant SVG elements, and it should be possible to implement a better rendering function that does not re-render elements or render perfectly overlapping elements. The caching of the current prototype should also be developed to be more fine-grained.

In the future, web browsers will most likely have greater support of the Web SQL and IndexedDB features of HTML5 which could allow for greater caching capabilities for web map clients than is possible with the Web Storage features used in this prototype.

There are also other HTML5 technologies that may be applicable for geo-information clients that have not been investigated during the implementation of the prototype. The WebGL feature of HTML5 enables hardware-accelerated rendering of 3D graphics in the web browser and may be useful for certain geospatial applications. In combination with X3D standard that might be included in the HTML5 standard in the future, web clients should be able to render 3D vector data that some map servers provide.

9 Conclusion

After having been mostly used by experts in desktop environments, GISs have become increasingly used by amateurs on the web. The great amounts of data associated in many GISs, the advanced user interfaces that are needed and the heterogeneous environment that is the web, new challenges have been introduced to GISs. Web map applications thus need to deal with challenges such as client-server transfer costs, data format diversity and the limited host environments that are web browsers.

However, in recent years the development of web browsers has probably been more fast-paced and interesting than ever. Web browser plugin technologies from companies such as Adobe (Flash) and Microsoft (Silverlight) have given the web browsers extended capabilities and allow for advanced web application development. They have introduced multimedia playback support, extended caching capabilities and a specialized runtime environment that the web applications run in.

In the most recent years there has also been intense work done to develop the HTML5 standard which will introduce many new features and capabilities that can be utilized by web application developers to create applications that can be run directly in any native HTML5-compliant web browser. Many web browsers are already implementing many of the most interesting features of HTML5, even though support still varies and the standard is still far from finished.

Alongside the development of the HTML5 standard, web browsers today have better capabilities of running JavaScript code and the web browsers' performance is rapidly increasing. The performance of JavaScript engines in web browsers today is many times greater than it was only a year ago from this writing. Graphics rendering performance has also improved and hardware acceleration is being introduced.

Considering the development of web browsers and the HTML5 standard, it has been determined in this thesis work that when also considering the fact that HTML5 is an open standard, the development of a rich web map client on an HTML5/JavaScript environment was as feasible as and more desirable than development on a vendor-specific plugin-based platform in this thesis work. It should however also be noted that HTML5 and plugin-based solutions are not necessarily in opposition to each other but rather complement each other and will most likely exist alongside each other also in the future.

By using the SVG capabilities available in many web browsers today, the prototype developed during the thesis work has been able to add more inherent "intelligence" to the web maps, where the elements drawn on the map are actually individual SVG elements rather than part of many tiled raster images. This enables user-client interactivity with each specific SVG element and geospatial analysis can be performed based on the elements' metadata.

By also using the native video and audio playback capabilities of HTML5-compliant web browsers, the user experience can be increased by adding geo-referenced video and audio content to the map application. That requires, however, that the service provider actually has relevant geo-referenced video or audio data that can be presented to the user, something which has not been demonstrated in this thesis work.

HTML5 also introduces features for greater caching capabilities than previously available in web browsers. Data can be stored on the client's host machine or in a web browser-contained database through the Web Storage, Web SQL or IndexedDB features; although at present most web browsers mostly support the first-mentioned.

It remains to be seen how plugin-based platform vendors will react to the introduction of HTML5. Most likely, new features will be introduced to plugin-based technologies in the future to compete with or complement HTML5. It also remains to be seen how well-supported HTML5 will be by web browsers in the future.

10 References

[1] USGS Open File Report 88-105 [A process for evaluating geographic information systems]
[2] The Development and Impact of Web-based Geographic Information Services, W. Tang and J. Selwood (http://www.gisdevelopment.net/technology/gis/mi03002.htm) Retrieved on 2010-09-15
[3] Recent developments in Internet GIS, M. Tsou (http://www.gisdevelopment.net/technology/gis/techgis_002.htm) Retrieved on 2010-09-15
[4] GIS Basics, S. Fazal
[5] Geographic Information Systems Demystified, S. R. Galati
[6] The Convergence of Web and Desktop GIS, J. Winslow, Directions Magazine, August 30th 2010
[7] Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008
[8] XML Demystified, J. Keogh and K. Davidson
[9] About OGC, The Open Geospatial Consortium (http://www.opengeospatial.org/ogc) Retrieved on 2010-09-15
[10] OpenGIS Web Map Service Implementation Specification version 1.3.0
[11] OpenGIS Web Feature Service Implementation Specification version 1.1.0
[12] OpenGIS Geography Markup Language (GML) Encoding Standard version 3.2.1
[13] GML – User perspective, D. Murray (http://www.gisdevelopment.net/technology/gis/ma04263.htm) Retrieved on 2010-09-15
[14] The research and applications of WFS based GML, J. Jiang, C. Yang, Y. Ren, M. Jiang
[15] Visualization of GML data using XSLT, W.T.M.S.B. Tennakoon
[16] Macromedia Flash MX – A next-generation rich client, J. Allaire
[17] The Essence of Effective Rich Internet Applications, K. Mullet
[18] The Triad-based Design of Rich User Interfaces for Internet Applications, Francisco J. Martínez-Ruiz
[19] A Web Compliance Engineering framework to support the development of accessible Rich Internet Applications, C.A Velasco, D. Denev, D. Stegemann and Y. Mohamad
[20] 7 Difference Between RIA and Traditional Web Application (http://flextutorial.org/2009/07/31/7-difference-between-ria-and-traditional-web-application/) Retrieved on 2010-09-15
[21] JavaFX as Rich Internet Application Platform (http://blogs.sun.com/jonathan/entry/rocking_the_free_world) Retrieved on 2010-09-15
[22] Rich Internet Application Market Share, StatOWL (http://www.statowl.com/custom_ria_market_penetration.php)
[23] IE Inadequate? No, Ajax is Inadequate. (http://www.zdnet.com/blog/stewart/ie-inadequate-no-ajax-is-inadequate/27) Retrieved on 2010-09-15

[24] Ajax vs RIA - Ajax still rules (http://www.zdnet.com/blog/web2explorer/ajax-vs-ria-ajax-still-rules/195) Retrieved on 2010-09-15
[25] Rich Internet Applications and AJAX - Selecting the best product, M. Domenig (http://www.javalobby.org/articles/ajax-ria-overview/) Retrieved on 2010-09-15
[26] RIA War Is Brewing, J. Rapoza (http://etech.eweek.com/content/application_development/ria_war_is_brewing.html) Retrieved on 2010-09-15
[27] ECMA-262 5th edition – ECMAScript Language Specification
[28] AJAX: Asynchronous Java + XML?, C.K Wei (http://www.developer.com/design/article.php/3526681/AJAX-Asynchronous-Java--XML.htm) Retrieved on 2010-09-15
[29] Ajax: A New Approach to Web Applications, J.J Garrett (http://www.adaptivepath.com/ideas/essays/archives/000385.php) Retrieved on 2010-09-15
[30] The W3C Document Object Model (DOM), P. Le Hégaret (http://www.w3.org/2002/07/26-dom-article.html) Retrieved on 2010-09-16
[31] Web browser DOM support (http://www.webdevout.net/browser-support-dom) Retreived on 2010-09-16
[32] Quirksmode Compatibility Master Table (http://www.quirksmode.org/compatibility.html) Retrieved on 2010-09-16
[33] Document Object Model (DOM) Conformance Test Suites (http://www.w3.org/DOM/Test/) Retrieved 2010-09-16
[34] Dynamically Typed Languages, L. Tratt
[35] An Analysis of the Dynamic Behavior of JavaScript Programs, G. Richards, S. Lebresne, B. Burg and J. Vitek
[36] Performance analysis of JavaScript, F. Smedberg
[37] Integrating AJAX Approach into GIS Visualization Web Services, A. Sayar, M. Pierce and G. Fox
[38] Web Services Architecture - W3C Working Group Note 11 February 2004
[39] ECMA-357 2nd edition - ECMAScript for XML (E4X) Specification
[40] Google Maps JavaScript API V3 (http://code.google.com/intl/sv/apis/maps/documentation/javascript/) Retrieved on 2010-09-17
[41] Bing Maps AJAX Control (http://msdn.microsoft.com/en-us/library/bb429619.aspx) Retrieved on 2010-09-17
[42] Flex 3: A Beginner's Guide, M.E. Davis and J.A. Phillips
[43] Introduction to ActionScript 3.0 (http://help.adobe.com/en_US/as3/learn/WS5b3ccc516d4fbf351e63e3d118a9b90204-8000.html) Retrieved on 2010-09-20
[44] AIR Bible, B. Gorton
[45] Silverlight 1.0, D. Rader, G. Hinkson and J. Beres
[46] Silverlight Overview (http://msdn.microsoft.com/en-us/library/bb404700(v=VS.95).aspx) Retrieved on 2010-09-21

[47] Silverlight Overview – Technical whitepaper (http://channel9.msdn.com/Learn/Courses/Silverlight4/Overview/Overview) Retrieved on 2010-09-21
[48] Expression Blend Overview (http://www.microsoft.com/expression/products/Blend_Overview.aspx) Retrieved on 2010-09-21
[49] Scalable Vector Graphics (SVG) 1.1 (Second Edition) (http://www.w3.org/TR/SVG11/) Retrieved on 2010-09-21
[50] Adobe SVG Viewer (http://www.adobe.com/svg/viewer/install/) Retrieved on 2010-09-21
[51] SVG Web (http://code.google.com/p/svgweb/) Retrieved on 2010-09-21
[52] FXG 2.0 Specification (http://opensource.adobe.com/wiki/display/flexsdk/FXG+2.0+Specification) Retrieved on 2010-09-21
[53] FXG and MXML Graphics (http://help.adobe.com/en_US/flex/using/WS145DAB0B-A958-423f-8A01-12B679BA0CC7.html) Retrieved on 2010-09-21
[54] Silverlight Documentation (http://msdn.microsoft.com/en-us/library/cc838158(v=VS.95).aspx) Retrieved on 2010-09-21
[55] GUIMark 2: The rise of HTML5 (http://www.craftymind.com/guimark2/) Retrieved on 2010-09-23
[56] Performance Analysis and Acceleration for Rich Internet Application Technologies, T. Ernst
[57] List of codecs supported by Adobe Flash Player (http://kb2.adobe.com/cps/402/kb402866.html) Retrieved on 2010-10-04
[58] Dive into HTML5 (http://diveintohtml5.org/video.html) Retrieved on 2010-10-04
[59] Flash Player System Requirements (http://www.adobe.com/products/flashplayer/systemreqs/) Retrieved on 2010-10-08
[60] Silverlight FAQ (http://www.microsoft.com/getsilverlight/Get-Started/Install/Default.aspx) Retrieved on 2010-10-08
[61] About Adobe Flash Player (http://www.adobe.com/software/flash/about/) Retrieved on 2010-10-08
[62] FusionMaps (http://www.fusioncharts.com/maps/Default.asp) Retrieved on 2010-10-08
[63] amMap (http://www.ammap.com/) Retrieved on 2010-10-08
[64] Google Maps API for Flash (http://code.google.com/intl/sv-SE/apis/maps/documentation/flash/) Retrieved on 2010-10-08
[65] FlashMapOne (http://www.flash-map-one.com) Retrieved on 2010-10-08
[66] Bing Maps Silverlight Control (http://msdn.microsoft.com/en-us/library/ee681884.aspx) Retrieved on 2010-10-08
[67] Web Workers, W3C Working Draft 29 October 2009
[68] Web Storage, W3C Working Draft 22 December 2009
[69] HTML5 – A vocabulary and associated APIs for HTML and XHTML, W3C Working Draft 24 June 2010

[70] Offline Web Applications, W3C Working Group Note 30 May 2008
[71] Modernizr (http://www.modernizr.com/) Retrieved on 2010-10-12
[72] Performance of Canvas versus SVG, Boris Smus (http://www.borismus.com/canvas-vs-svg-performance/) Retrieved on 2010-10-14
[73] A Scalability Study of Web-Native Information Visualization, D.W Johnson and T.J. Jankun-Kelly
[74] Bubblemark animation test (http://bubblemark.com/) Retrieved on 2010-10-15
[75] jQuery (http://jquery.com/) Retrieved on 2010-10-15
[76] HTML 5 Editor Ian Hickson discusses features, pain points, adoption rate, and more (http://blogs.techrepublic.com.com/programming-and-development/?p=718) Retrieved on 2010-10-19
[77] http://www.webmproject.org/ Retrieved on 2010-10-20
[78] Firefox 4 Beta: Bringing Hardware Acceleration (http://www.basschouten.com/blog1.php/2010/09/07/firefox-4-beta-bringing-hardware-acceler) Retrieved on 2010-10-20
[79] The Architecture of Full Hardware Acceleration of All Web Page Content (http://blogs.msdn.com/b/ie/archive/2010/09/10/the-architecture-of-full-hardware-acceleration-of-all-web-page-content.aspx) Retrieved on 2010-10-20
[80] Flash Player: CPU Hog or Hot Tamale? It Depends. (http://www.streaminglearningcenter.com/articles/flash-player-cpu-hog-or-hot-tamale-it-depends-.html) Retrieved on 2010-10-20