Integrating the Web and the World: Contextual Trails on the Move

Frank Allan Hansen
Niels Olof Bouvin
Bent G. Christensen
Kaj Grønbæk
Dept. of Computer Science
University of Aarhus, Denmark

Torben Bach Pedersen
Dept. of Computer Science
Aalborg University, Denmark

Jevgenij Gagach
Euman Ltd., Denmark

ABSTRACT

This paper presents applications of HyCon, a framework for context aware hypermedia systems. The HyCon framework encompasses annotations, links, and guided tours associating locations and RFID- or Bluetooth-tagged objects with maps, Web pages, and collections of resources. The user-created annotations, links and guided tours, are represented as XLink structures, and HyCon introduces the use of XLink for the representation of recorded geographical paths with annotations and links. The HyCon architecture extends upon earlier location based hypermedia systems by supporting authoring in the field and by providing access to browsing and searching information through a novel geo-based search (GBS) interface for the Web. Interface-wise, the HyCon prototype utilizes SVG on an interface level, for graphics as well as for user interface widgets on tablet PCs and mobile phones.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypermedia

Keywords

Context aware Hypermedia, XLink, SVG, Open Hypermedia

General Terms

Algorithms, Design, Experimentation, Human Factors, Standardization

1. INTRODUCTION

The ability to access the Web from mobile devices has become commonplace with Web and WAP browsers found in many mobile phones and PDAs. Limitations found in these devices provide a number of challenges, such as limited storage, bandwidth and processing power as well as small screens. Much work [4, 8, 16, 24] have been done to address these limitations and provide a satisfactory user experience. Hand-held devices have some unique advantages compared to more stationary computers: They are mobile, in-the-field, and they can to an increasing degree determine their own location with high accuracy. Knowledge of location can be used to offset the constraints of small user interfaces by automatically providing the user with localized information. While location-based information systems for, e.g., tourists, are well known [5], these systems usually rely on predefining all available information. This reliance on special purpose authored content is unfortunate, as it puts the onus of creating content on the maintainers of the system rather than on its users or the Web in general. Furthermore, users should certainly be allowed to add their own material and observations to a system.

We present in this paper applications of the HyCon framework for context-aware mobile hypermedia. The HyCon architecture encompasses user authored annotations, links, and guided tours associating locations with maps and Web pages. The architecture provides a general SVG based user interface deployable on a number of devices ranging from mobile phones to tablet PCs. The system uses SVG in a novel way to provide application integration. The paper also demonstrates the use of location derived addresses for seeding Web searches—Geo-Based Searching (GBS). GBS is used to alleviate the problem of relying on specially authored content found in many location-based information systems. The main goals in the development of HyCon have been to:

Extend hypermedia to the physical world: hypermedia in the physical world should allow users to link and tag physical locations and objects with digital content such as multimedia documents and comments similar to traditional hypermedia systems. HyCon aims at going beyond existing mobile guide, navigation, and browsing systems and support users in both creating and retrieving hypermedia structures in the field. An important contribution of this paper is the illustration of support for the classical hypermedia mechanisms for browsing, searching, annotations, and guided tours in the physical world, allowing users to link objects in both digital and physical space.

Support automatic collection of context information: hypermedia in the physical world necessitates awareness. Information produced by users should automatically be tagged with context information allowing later retrieval through either browsing or searching of the context. Integrating physical sensors into the hypermedia system and utilizing sensor data can automate this process. Structuring information by context can thus be handled "behind the scenes" requiring no explicit user action.

HT'04, August 9–13, 2004, Santa Cruz, California, USA.

Support social computing: groups of users may utilize the structuring mechanism as a means of communication. This can include anything from leaving personal tags as "digital graffiti", restaurant visitors leaving comments for other visitors to read, to contractors documenting repair work in the field.

Support heterogeneous mobile devices: investigating techniques supporting hypermedia on a wide variety of heterogeneous mobile devices have been a key concern in the development of HyCon. We address these issues by presenting novel techniques for using SVG for application integration as well as a presentation medium for hypermedia structures on devices with very different network and interface capabilities.

The work in this paper is done within the ContextIT project involving two industrial partners Euman Ltd. and TeleDanmark Ltd. The research and prototyping activities have resulted in the HyCon framework (Section 2–3) and a Symbian SVG browser (Section 4). A number of concepts from the HyCon prototype such as the annotation facilities and the SVG-based client framework will be integrated in the next version of Euman's "LifePilot"[1].

The paper is structured as follows: The HyCon framework is described in Section 2. Section 3 describes the prototypes produced so far and delves into the intricacies of expressing geo-spatial hypermedia in XLink. Section 4 describes our use of SVG. Section 5 covers related work, Section 6 describes directions for future work, and the paper is concluded in Section 7.

2. THE HYCON FRAMEWORK

The HyCon framework and architecture for context-aware mobile hypermedia was developed to provide a general platform suited for experiments with hypermedia mechanisms in a context-aware and mobile environment as described in detail in [3].

The logical layers and infrastructure of the architecture are seen in Figure 1, with four layers divided into Storage, Server, Terminal, and Sensor. The bottom layer, the Storage layer, handles persistent storage and retrieval of hypermedia structures produced in the system. The Server layer includes components handling annotation, link, location, and Search functionality. The functionalities of these components are offered through services implementing the Service Interface, which are realised as Web Services and Java servlets. Through these interfaces applications in the Terminal layer communicate with the services in the Service layer. Applications in the Terminal layer are not limited to a specific platform, but may be running on a variety of hardware platforms and software environments (phones, tablets, laptops, Web browsers). The key property of the HyCon framework is the last layer, the Sensor layer. This layer is introduced to logically group all sensors deployed to obtain contextual information. The sensors may be further divided into two categories: *local sensors* and *remote sensors*.

Local sensors are attached directly or integrated with the equipment on which the sensed information is used. This kind of sensor can also be described as private sensors, since they only supply contextual information directly to a single application. Examples of this type of sensor are: phone cameras, built-in infra red sensors, built-in sound recorders, and GPS receivers.

Remote sensors offer the sensed information through a network connection. These sensors can also be described as public sensors, as they are often placed at a fixed location, from where they provide their contextual information to applications. Examples of these sensors are RFID tag readers, and Bluetooth sensors sweeping the area for nearby Bluetooth units. To avoid misuse of these public sensors, an authentication mechanism similar to the one used by most Wireless LAN access point could be deployed e.g., filtering by MAC-address, IP number, or IP subnet.

[1] `http://www.life-pilot.com/` (in Danish)

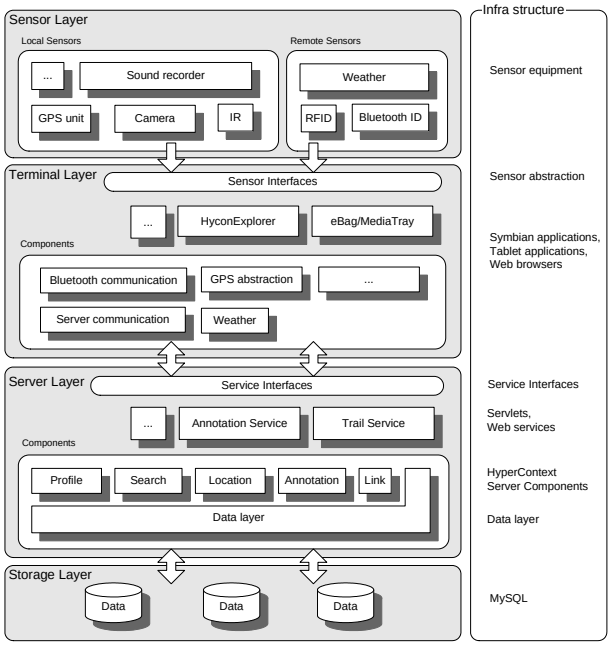


Figure 1: The HyCon service framework architecture. The framework is divided into four layers: the Storage layer, the Server layer, the Terminal layer, and at the top the Sensor layer providing sensed data for the terminal applications.

All sensors in the Sensor layer can be accessed from applications in the Terminal layer through a Sensor interface, which provide an abstraction for both local and remote sensors.

2.1 Data model

The HyCon hypermedia model has been designed to support modeling of context information, annotations, link trails, and general objects representations. The data model have similarities with earlier hypermedia data models, e.g., the OHSWG data model [22], but additionally focuses on modeling context information.

The data model is illustrated in Figure 2 using Whitehead's Containment Model notation [31]. All objects in the model are described as subclasses of the abstract `AbstractObject` class. Instances of its subclasses all share common attributes: they all have globally unique identifiers (GUIDs), meta-data describing their creator and modification time-stamps, and a set of property-value pairs. Instances of the `Context` class are referential composites, which can hold other objects belonging to the same physical or digital context. Objects from the `Location` class represent physical locations and are used as the primary mechanism for identifying physical locations in the model. The `Annotation` class represent the data model's support for annotations. Annotations can include multimedia content such as text, images, sound, and video. Annotations can be associated with any type of object derived from the `Object` class. Hence, the model supports the general notion of annotations annotating other annotations. Links and link trails are modeled by the `Link`, `Arc`, and `Locator` classes describing XLink [9] based structures on top of the other objects in the system. The Locator objects are used to represent digital resources

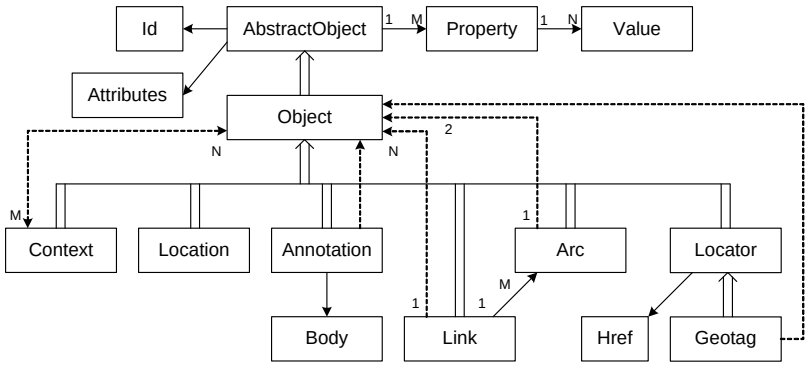


Figure 2: The HyCon framework data model.

reachable by a URI in the model with Arc objects specifying the traversal behavior of the link. The data model has been generalized to support linking instances of the generic `Object` class and thus support easy linking of `Locator`-type objects as well as classes derived from the `Object` class. The `GeoTag` class is used to tag elements directly with URIs (e.g., to tag a location with a descriptive Web page).

3. THE HYCON TABLET PC PROTOTYPE

The HyConExplorer is a HyCon Terminal layer application prototype. The prototype is implemented in Java and runs on commercially available tablet PCs connected to the network through WLAN or GPRS connections on cell phones. The tablet PCs are equipped with cameras for capturing images, video, and audio. The user's physical location is registered through a small Bluetooth-enabled GPS unit (see Figure 3). The prototype implements the full data model described in the previous section. Communication to server layer services is primarily through the Web Service SOAP interface, with hypermedia structures and context objects being sent as XML encoded Java objects. However, upload and download of shared multimedia files are done through the CGI interface.

The HyConExplorer prototype has been used as a test bed for several context-aware hypermedia techniques: location-based browsing, geo-based searching, and annotation and linking. These techniques are discussed in the following sections. HyConExplorer also supports presentation on large displays of material produced with the system. Presentations can be created by selecting annotations and adding them to a slideshow structure. The slideshow can then be displayed in full-screen mode with the HyConExplorer and the tablet can be connected to a projector as a convenient way to present the collected material.

3.1 Location-based hypermedia functionality

HyCon adds context-aware support to the classical hypermedia concepts of browsing, searching, and creating and following links, annotations and guided tours. Using the HyConExplorer system, users can create links, annotations, and trails of links, which are automatically tagged with context information sensed from the physical environment. When moving about in the physical world, these hypermedia structures are presented to the user, if the user's context matches that of the structures. Similarly, automatic execution of searches through unstructured Web resources for information relating to user's context may be initiated when parameters change in the user's physical context (time, location, etc.).

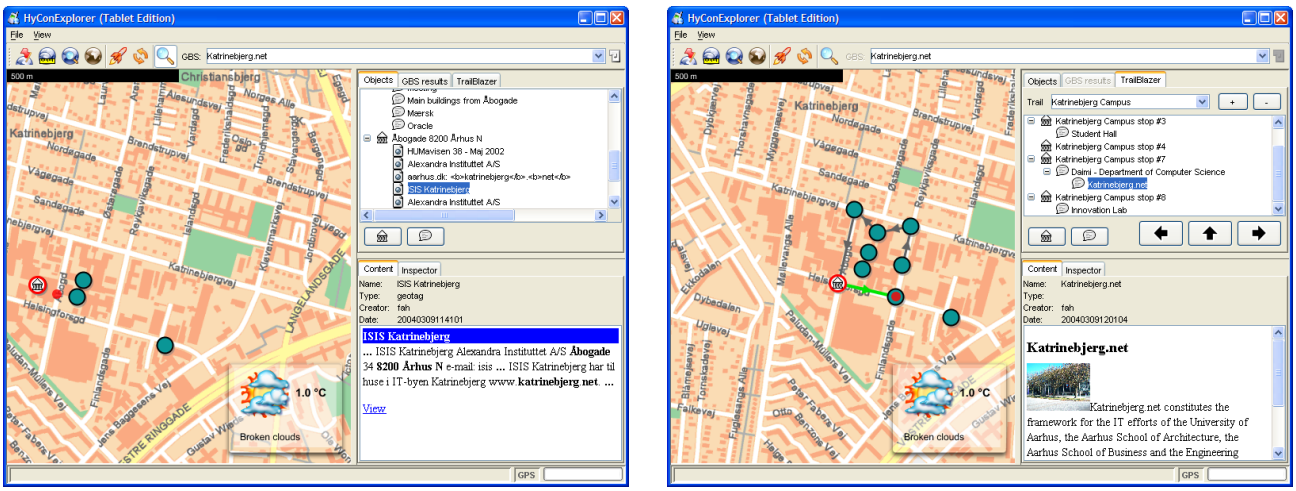
In the following we describe how context information can be utilized when browsing, searching, and creating hypermedia structures.

Figure 3: Context-aware hypermedia on a mobile tablet PC with camera. The tablet has network connection through a Bluetooth and GPRS enabled cell phone and reads GPS coordinates from a Bluetooth GPS unit.

3.1.1 Location-based browsing

We have in HyCon experimented with both explicit user controlled browsing (as found in, e.g., Web browsers) as well as implicit browsing, where the user initiates browsing by moving around. Thus, the user can engage in *indirect representational navigation* and *direct physical navigation* [15]. Indirect representational navigation is browsing information related to an area without necessarily being present, e.g., by clicking on a map over a city or studying a list of museums. Direct physical navigation on the other hand is linked to time and space, as the information presented to the user is determined by the user's whereabouts. This approach has found use in a number of tourist guide systems, such as GUIDE [5], where information presented to the user depends on location (which attractions are nearby) and time (which of these are open).

Browsing the hypermedia structures through direct physical navigation requires no explicit user intervention—the action takes place "behind the scenes". The mobile client captures contextual information from the physical environment and sends it to the HyCon server where it is mapped to relevant hypermedia structures. Hence, instead of browsing through information in the digital space, users may "bump into" information resources by simply changing their position in physical time and space. We see this kind of browsing as an interesting alternative to the common indirect representational navigation, especially when dealing with small mobile devices with limited interfaces and input controls, where traditional browsing easily becomes tedious.



(a) Geo-based Search.

(b) Following a trail.

Figure 4: Browsing and searching in the HyConExplorer prototype.

3.1.2 *Geo-based search*

The HyConExplorer prototype supports the notion of Geo-based search (GBS). In essence, GBS is Web searches augmented with information about the user's current location. The goal of GBS is to limit search results to pages covering both a topic of interest (specified by user supplied key words) and the particular geographical area the user is located in. This mechanism is especially useful when using small devices with limited display capabilities and poor support for browsing through large numbers of search results.

The majority of Web pages related to locations (such as landmarks or many businesses) are not indexed according to their physical location, as proposed by standards such as GeoTags[2] and GeoURL[3]. However, most such Web sites include their postal addresses and these addresses are indexed by search engines. The postal addresses can then later be retrieved as exemplified by Google's "Search by Location" service[4]. Thus, to use existing search engines not prepared for location-based searches, the search criteria has to be formulated in a way appropriate for their indexing techniques.

When using GPS sensor equipment, location information acquired from the sensors is typically encoded as UTM (x, y)-coordinates or (longitude, latitude)-pairs. Since existing mainstream search engines such as Google do not index pages by these GPS coordinates, the coordinates have to be transformed to some other information. We achieve this by mapping the coordinates to postal addresses. A database of all public postal addresses in Denmark and their GPS coordinates is freely available from the Danish chart provider KMS[5]. Using this database as the basis, mapping raw location information to textual postal addresses provides much more useful input to search engines. As discussed in section 5, McCurley [20] describes several other approaches to determining the geographical location of Web pages making it possible to support searching schemes like GBS.

The current implementation of the GBS component uses Google as the back-end search engine and utilizes the Google Web APIs[6]. From the GPS coordinates the names of every street within a fifty meters radius is determined and optionally combined with user supplied key words to formulate search strings. The matching Web pages contains the keywords and the postal address printed somewhere in the pages. Currently, we return the first ten search results for each street name with no further filtering. The search result is then plotted onto a map over the searched area to intuitively present the connection between geographical locations and Web pages.

Figure 4(a) illustrates this kind of browsing and searching in the HyConExplorer prototype. Based on a few search terms and the captured context information, the context server has located two roads near the user matching the search. Search hits are marked with link markers (dots) on the digital map and is shown in a corresponding list view. When activating a link marker, the link endpoint is presented by a snippet of text. If the endpoint looks interesting it can be viewed in an external viewer (Web browser, video player, etc.).

3.1.3 *Location-based link trails*

The HyCon data model implements a link mechanism based on the XLink standard [9]. This has resulted in a model supporting 2-ary links linking external Web pages to objects in the link-base and general extended links (n-ary links) linking a collection of objects (typically locations) into a single logical trail through the objects.

While the main application of XLink is expected to be linking within XML documents, the standard itself is not limited to address solely XML locations. XLink supports simple links similar to the links currently found on the Web (i.e., unidirectional one-ary links), as well as extended links, which can be bi-directional n-ary typed links stored externally to the constituent documents. An extended link consists of an `extended`-type link element containing a number of `locator`-type child elements designating remote resources (or `resource`-type elements holding local resources). XLink `arc`-type elements are used to describe traversal order in a link (i.e., from one resource to another).

HyConExplorer supports users leaving trails of information as

[2] `http://www.geotags.com/`
[3] `http://www.geourl.org/`
[4] `http://labs.google.com/location/`
[5] `http://www.kms.dk/`
[6] `http://www.google.com/apis/`

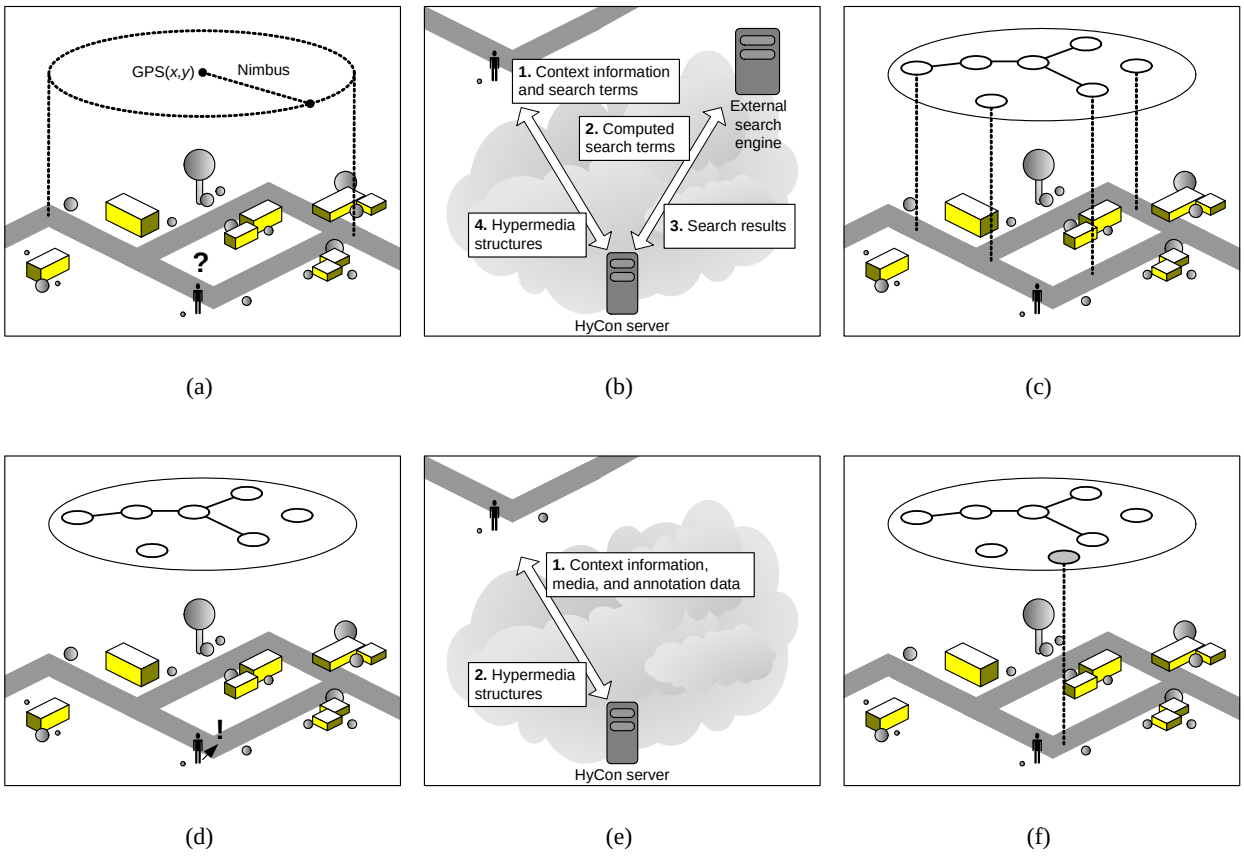


Figure 5: A typical pattern for context-aware browsing, searching, and annotation of resources.

they move about in the physical world. Similarly to many tourist guide systems which features (predefined) guided tours or direction specifications to points of interest in e.g., larger cities, these user created trails can be viewed and followed by other users present in the same area. However, combined with the annotation facilities the trail may serve as personalized impressions of a given area and even allow users to share their impressions through discussions linked to point of interests along the trail. Figure 4(b) illustrates a user browsing through such a trail in the HyConExplorer.

HyCon trails are structured as extended n-ary XLink based links. The trail structure is a generalization of the XLink based guided tours implemented in the Xspect system [6]. Each link represents a trail with location objects designating points of interest along the trail and arc objects spanning the route between each pair of locations. Client applications, such as the HyConExplorer, which implement the full data model and communicate with the HyCon server through the Web service interface may simply rely on the objects defined in the linking model when implementing the linking and trail mechanisms. The `Link`, `Arc`, `Locator`, and `Location` class objects are identified by their GUID fields and are associated through object references. However, client applications which do not implement the data model and communicate with the server through the CGI interface need to rely on the link structures in standard XLink format. The conversion from object structures to XLink files are straightforward. When a link structure has been computed by one of the server components, a `toXML()` method is called for each object. This method generates a generic XML representation of the objects which is passed through a XSLT stylesheet filter chain before the structures are returned to the client. The stylesheet filters transform the generic XML representation to an XLink format structure. However, applications using the CGI interface have no direct access to linked objects but have to make calls through the CGI interface to retrieve these objects. This is accomplished by creating `href`-attributes in the `locator`-type elements pointing to services capable of returning a representation of the object. Consider the following link fragment:

```
<link xlink:type="extended"
  xlink:role="http://fahbentor.daimi.au.dk/gt">

  <loc xlink:type="locator"
    xlink:href="http://fahbentor.daimi.au.dk:15342\
    /contextservices/locationservice?op=getLocation&id=59"
    xlink:label="id59"/>
  <loc xlink:type="locator"
    xlink:href="http://fahbentor.daimi.au.dk:15342\
    /contextservices/locationservice?op=getLocation&id=93"
    xlink:label="id93"/>
  ...
  <arc xlink:type="arc"
    xlink:from="id59" xlink:to="id93"/>
  ...
</link>
```

The fragment illustrates part of a trail with an `arc`-type element connecting two `locator`-type elements. The locators' `href`-attributes have been computed in the stylesheet to point to the location service on the HyCon server. The exact format used for XLink format trails can be found in the Xspect DTD[7].

[7] `http://fahbentor.daimi.au.dk/xspect.dtd`

3.1.4 *Location-based annotations*

A significant feature of the HyConExplorer which characterize it from other related systems is the support for "the user as producer". Users are enabled to produce and organize digital situated content. This facilitates advanced documentation of situations or points (or items) of interest, and a way to share material with others. To assist the user in creating material, contextual information provided by sensors is associated with the material. Context information such as: the user's location, time of day, name of the user, and type of the material is associated with the produced content when saved and structured in the system. One related system which has taken a similar approach to user produced annotations is GeoNotes [21]. However, GeoNotes features a much more advanced filtering scheme for retrieval of annotations than currently implemented in the HyConExplorer.

The annotation mechanism supports social computing by allowing users to share material by e.g. leaving comments outside a restaurant about how tasteful the food is, so that other (potential visitors) passing by may discover this location relevant material. As the data model supports annotations to be associated with all instances of subclasses of the generic Object class in the model, users are allowed to comment on each others annotations. This mechanism can be used to create entire situated discussion threads in a given context.

3.1.5 *Usage Patterns of the HyConExplorer*

Supporting mechanisms for browsing, searching, annotations, and guided tours give rise to some typical usage patterns.

Figure 5 illustrates a typical pattern for browsing and searching based on direct physical navigation and the GBS search technique. As the user, equipped with a mobile terminal (cell phone or tablet PC) and GPS receiver, walks down a street, the time and his/her position are continuously updated in the system. When the position has changed by a certain delta (the nimbus), the system makes an update request to the HyCon server (Figure 5(a)). The request includes the context parameters captured by the system, the user's nimbus (the region the user projects him/herself to as introduces in the MASSIVE CVE system [14]), and any additional user specified search terms (see Figure 5(b)). The nimbus is controlled with a slider in the user interface and is visualized by the map scale—a large nimbus will correspond to a larger map area. The update delta is computed as the length of the nimbus with origin equal to the last known updated position. The nimbus used in GBS is fixed to a small distance of 60 meters. This choice have been made to limit the number of hits to roads nearby the user and from our experiments we have found this distance sufficient in urban areas. From the request, the server computes appropriate search terms and retrieves search results from the external search engine. These results, together with any matching hypermedia structures (links, annotations, and trails) stored in the server's database, are returned to the user (see Figure 5(c)).

Figure 5(d)– 5(f) illustrates a typical pattern for creating annotations in the HyConExplorer. The HyConExplorer is updated with several existing hypermedia objects relating to the user's current location acquired from browsing and searching. Link markers on the map are displayed for objects which can be annotated: locations, linked documents, annotations, and search results. The user chooses, however, not to annotate any of the existing objects but a new location. After having photographed the location and entered a small comment, the annotation data is sent as a request to the context server together with collected context information. The context server updates the database with the new location and annotation data and returns the updated structures to the client. The annotation is now associated with context information and can later be retrieved by other users on the same location.

4. SVG-BASED CLIENT FRAMEWORK

Developing applications for heterogeneous computing platforms with different abilities poses a number of challenges, not least of which is that of the user interface. For the HyConExplorer, we have been developing on phones with a screen resolutions of 176×208 pixels and tablet PCs with a screen resolution of 1024×768 pixels—more than 21 times the resolution! Clearly, being able to provide a comparable user experience at such disparity is quite a challenge. Furthermore, as we expect future versions of the HyConExplorer to be ported to other mobile devices with dissimilar screen resolutions, a more general solution is needed.

Scalable Vector Graphics (SVG) [13] has since its recommendation been a technology in rapid growth. It is a very comprehensive standard for vector graphics. In addition, through the SVG Tiny and SVG Mobile standards, it is also realistic to support resolution independent graphics on hand-held devices. While rich in drawing primitives, SVG also supports an event model well suited for scripting. This facilitates the development of user interface primitives directly in SVG. Developing user interfaces in SVG has a number of advantages: SVG is resolution independent; it can provide a rich interface; and it fits well in a XML production line.

4.1 The Symbian SVG Browser

A special effort has been put into creating an SVG-based "browser" for the Symbian operating system, with Nokia 7650 and 3650 terminals. We have extended the tools currently available on the Symbian platform with support for *SVG scripting*, e.g., displaying a picture when the cursor moves over an SVG item, *server communication*, and *local file management*, i.e., with the advanced features typically found in a PC Web browser with an SVG plugin. This allows us to create a complete client application using only SVG with embedded scripts.

Figure 6 shows the following three screenshots from the Mobile SVG Browser, from left to right: 1) the overview map with indication of the user location (the dot surrounded by a dotted circle, red on the user's screen) and two location-based annotations (the dots surrounded by dotted squares, green on the user's screen), 2) the screen for making a location-based annotation, and 3) the picture (of the participating school children) contained in the location-based annotations. Note that each screen only contains the most central information and functionality for the given purpose.

The architecture of our mobile SVG browser is seen in Figure 7. The box surrounding the top of the figure indicates the functionality available in a PC-based browser and SVG plug-in. The central entity is the SVG DOM tree (the parsed version of the SVG document) (seen in the middle of Figure 7) which contains the relevant data for display to the user, as well as other data (more on this below). The SVG DOM tree is rendered on the screen by the SVG

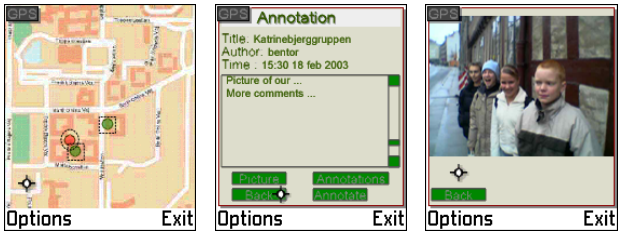


Figure 6: Mobile SVG Browser Screenshots

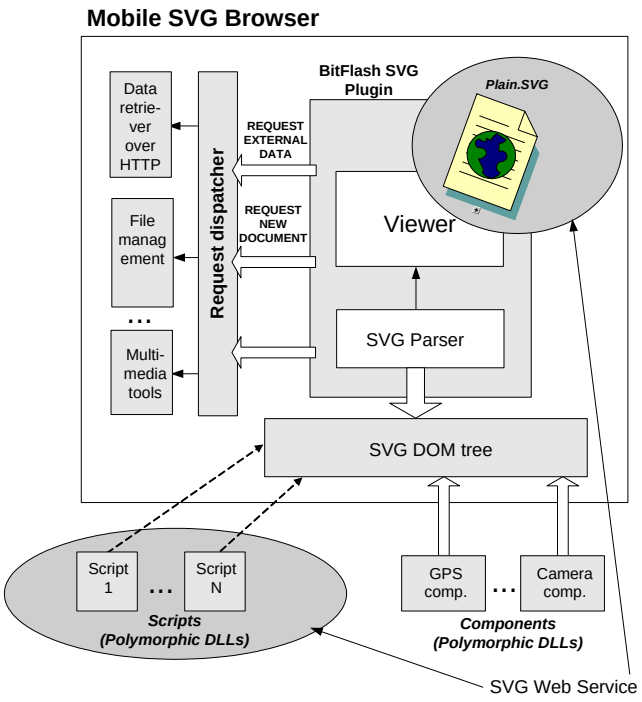


Figure 7: Architecture of the Mobile SVG Browser.

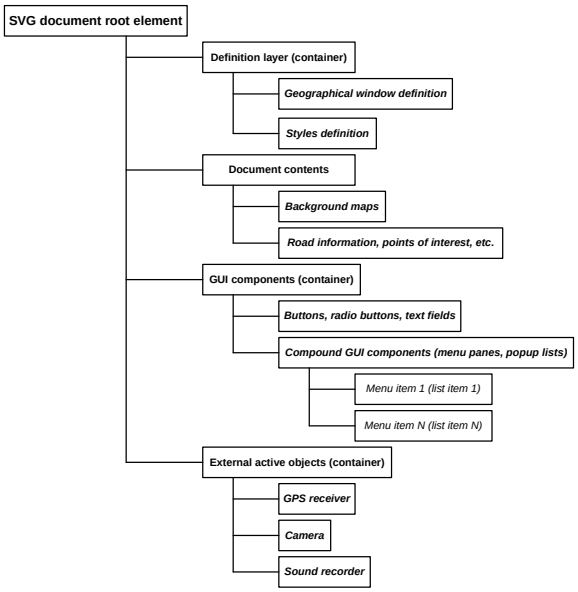


Figure 8: Structure of SVG DOM Tree.

plug-in (in the middle top half of Figure 7). We use the BitFlash SVG plug-in[8], which is one of only two SVG viewers currently available for the Symbian OS. The SVG plug-in provides functionality for parsing and viewing the SVG document, as well as reading and manipulating the SVG DOM tree. As the BitFlash viewer does not yet support scripting, we have invented our own scripting mechanism, termed *pseudo-scripting*. Here, we use polymorphic DLLs written in C++ and compiled for the Symbian platform to perform the scripting tasks, allowing us to dynamically manipulate the DOM tree and thus achieve the benefits of embedded SVG scripting. The scripts and SVG viewer can issue requests via the Request dispatcher to a number of runtime services (upper left part of Figure 7). These include a Data Retriever services for getting and sending data over HTTP, a File Management service for manipulating local files on the terminal, and one or more services for supporting multimedia tools, e.g., JPEG or WMA manipulation. Finally, a number of *components* (bottom right) handle external devices, of such as GPS units, cameras, etc. The components communicate with other applications on the terminal using the DOM tree, as explained below.

An SVG Web service/application is then implemented by providing functionality for creating one or more SVG documents in the server layer (exemplified by Plain.SVG in Figure 7) and sending the SVG document to the terminal layer, as well as a number of "pseudo-scripts" on the terminal layer (bottom left in Figure 7) to handle the (dynamic) aspects of the application logic on the terminal. The application logic on the terminal layer is (logically) divided into three parts. First, the *Context Handler* manages the context of the user and the terminal. The location context can be provided in several ways, of which we have implemented three: an external Bluetooth GPS unit for outdoor positioning, an "Internet Positioning" that can convert typed-in addresses into locations using an Internet service, and a barcode-based service that can convert scanned bar-codes into locations. Additionally, context information about the time, e.g., time and calendar and the weather is available. Second, the *Tools Handler* handles external or built-in devices that are used as explicit tools by the user (rather than providing implicit context). Here, we have implemented support for a digital camera and a sound recorder (both built into the 7650 and 3650). Third, the *application specific logic* analyzes the context and figures out which tools are available on the given terminal, in order to customize the user interface. Also, algorithms and decision logic specific to the actual applications reside here.

[8] `http://www.bitflash.com/`

4.2 Using SVG for application integration

Communication between the individual parts of the application logic is done by manipulating the DOM tree and dispatching or receiving the corresponding DOM events. For example, a GPS receiver creates an SVG element in the current SVG document with attributes to represent the geographical coordinates of the GPS receiver and modifies the attribute when a new coordinate is obtained. It then creates a DOM event and propagates it to the DOM tree. Any service can register for receiving this event and thus be notified whenever the coordinate attributes of the GPS SVG element change. On the other hand, a camera, e.g., in the mobile phone, will produce a whole new SVG document, which can then be displayed.

The structure of the SVG DOM tree presented in Figure 8 is as follows. Under the SVG document root element, we have four major types of elements. First, we have a *definition layer* that contains definitions that pertain to the whole document or the application the document is part of. This includes style definitions as well as location-based definitions such as the geographical window the user is interested in. Second, we have the actual *document contents* layer that contains information to be displayed to the user such as background maps, road information, points of interest, etc. Third, the *GUI components* layer defines the controls and widgets to be used in the user interface, e.g., buttons, radio buttons, text fields, menus, and pop-up lists. Fourth, the *External active objects* layer contains information from the devices in the sensor layer. Here, we

have so far implemented support for a Bluetooth GPS receiver, a digital camera, and a sound recorder. This layer contains information that is more or less "internal", i.e., not necessarily displayed to the user, to the applications running on the terminal. For example, the GPS receiver component will define elements containing the geographical coordinates of the receiver in both UTM and (longitude,latitude) formats. This information can then be accessed by other components and the application logic.

To our knowledge, the fourth layer illustrates a *novel way of using SVG*, namely to integrate *"internal" application logic and components*, i.e., functionality that is not directly connected to the GUI. The application logic/components need only know what information in the SVG DOM tree is relevant for them, and then 1) update the DOM tree and dispatch a DOM event if they want to update the information, and 2) register to receive the relevant DOM events when others update the information. For example, the Bluetooth GPS component may define SVG elements for its coordinates as described above. Whenever the coordinates change by more than a few meters, the component will change the coordinates in the SVG DOM tree and dispatch the corresponding DOM event. This event will then automatically be forwarded to other components and/or parts of the application logic that have subscribed to this event. This could include a background service fetching a new background map from an Internet service when the user moves outside the current map, and a service that uses the Geo-based search described above to provide content about the users' surroundings.

The advantages of using SVG/DOM as the integration mechanism are that they are W3C standards and thus independent of the terminal platform (OS, programming language, terminal hardware), while at the same time providing advanced mechanisms such as event handling that are normally only available within specific programming languages such as Java. Thus, multi-platform services are much easier to create. Additionally, the individual components and parts of the application logic need not be aware of each other, e.g., the application logic just needs to know what the current location is, but not how the location is obtained.

5. RELATED WORK

The notion of computers responding according to their users' implicit stated context is an intriguing and challenging one. The idea of utilizing location in computing systems was advocated by Mark Weiser [30], and the concept was further developed by Schilit *et al.* [23] and Dey [10, 11]. Dey states that there are three types of context-aware application support: presentation of information and services, automatic execution of services, and tagging of context to information to support later retrieval. The HyConExplorer described in this paper illustrates all three modes of operation, indeed it allows its users to freely annotate and link while in the field.

Context-aware systems have taken advantages of context in different ways. Location is a predominant feature used by guide systems such as the Cyberguide [1], the Touring Machine [12], PARCTab [29], or GUIDE [5]. These systems differ in scope (e.g., PARCTab was for indoor use), interface ranging from PDAs (PARCTab) over tablet PCs (GUIDE) to Augmented Reality goggles (the Touring Machine). Common is the ability to provide the user with situated information related to the user's current surroundings. Location may be determined through GPS (the Touring Machine), WiFi access points (GUIDE), or infrared broadcast (PARCTab). An early example of a Web based context aware system was Mobisaic [26]—a modified Mosaic Web browser which could parse dynamically updated local environment variables (containing e.g., the user's location) in URLs, allowing the user to access Web pages generated to e.g., reflect the surroundings. The combination of virtual and real in the context of gaming has been explored in the *Can You See Me Now?* game [7]. Museums have seen some development, as seen in the EQUATOR City [18] project where visitors can share their experiences across a number of media (the system is also notable for its ultrasonic positioning system).

These systems are however largely concerned with situated *browsing* of information rather than situated *authoring*, which has been an aim of HyCon. While the HyConExplorer presented herein predominantly relies on location as contextual information (as do most context aware systems), there are different aspects of context, such as "digital context" (e.g., the nearest printer or most suitable display). For a deeper investigation of this topic, see [3]. At this point, HyCon supports, in addition to location, the use of Bluetooth and RFID tagged objects for authentication purposes.

Mobile phones have in recent years turned into "smart phones", featuring applications such as WAP or Web browsers, email clients, digital cameras, and Bluetooth. This kind of functionality creates a new venue for the development of context-aware applications. Apart from ordinary digital communication such as text and multimedia messaging, this has opened up for a merging of the mobile phone with the Web, few places seen clearer than in the so-called "Moblog". Blogs (W*eb log*) are the widespread phenomena of online journals. The combination of mobile phones and blogs yields the moblog, where the maintainer can add content from a mobile phone, so impressions and pictures can be added while on the move, and thus creating situated content. Lacking is the automatic addition of location, which is left for the maintainer to do. In addition to moblog, another variant is that of "blogmapping"[9], an endeavor to encode location (using a special XML name space) into blog entries, thus allowing users to map locations with annotations. Blogging naturally lends itself to social computing, where a group of users may e.g., collaborate to map all worthwhile clubs or restaurants in an urban area. A system that has explored social filtering for geo-spatial information systems is GeoNotes [21], which supports annotating locations.

The coupling with online material with location has also resulted in systems such as the World-Wide Media eXchange (WWMX) [25], where users can tag images with metadata such as geographical location, time, and owner. The location tagging can be automated, providing the user is using a GPS receiver. The images can then later be shared with others and browsed based on location.

Commercially, some systems have explored the combination of location, annotation, and mobile phones, e.g., *TagandScan*[10]. TagandScan users can annotate a location (determined from their current (mobile phone) cell) with text, image, and a category (e.g., restaurant), and publish this annotation to the general public or just share it with their friends. Using cell information for location has the advantages of not requiring a GPS receiver—on the other hand, it is not supported by all telephone companies and is much less precise.

As described in Section 3.1, there are a number of initiatives advocating the use of geo-tagging Web pages. Should these approaches become widespread, they will fit nicely within our system as we would be able to bypass the process of transforming locations into addresses. McCurley [20] describes several approaches to determining the geographical location of Web pages. In a sense, we are interested in the opposite—we already *have* a location and desire related Web pages. Given that we rely solely on nearby addresses, this is a much simpler approach, given the ability of translating locations into addresses.

[9] `http://www.blogmapper.com/`
[10] `http://www.tagandscan.com/`

Our approach is similar to the Google Search by Location service, as both allow the user to search for Web pages given an address. However, whereas the Google approach requires the user to actually know (and enter) the address, GBS allows the user to search for "here". Our implementation only works for Danish addresses, and Google currently supports solely American addresses, but if Google at a future point added Danish addresses to its database, it would fit very well indeed with our system.

We are by no means the first to investigate SVG on small devices, as evidenced by SVG Tiny and SVG Mobile. Technical aspects apart, there still remains the challenge of adapting SVG content to smaller screen. This has been considered in [19]. Standard SVG applications can do scripting of user interface logic through e.g., JavaScript, as in [6]. In contrast, we extend the use of SVG to location-based services on mobile terminals and additionally use SVG as a foundation for device and application integration.

The current Euman system [17] integrates location-based context information in one *physical database*, based on a sophisticated model of the road network, i.e., a *tightly coupled, data warehouse-like* integration approach. Trails in the current Euman system can thus only be recorded for objects moving solely within the road network, such as cars. In contrast, the HyCon system utilizes hypertext constructs to achieve a *loosely coupled* integration approach that allows seamless integration of context information residing in external sources. Additionally, trails in the HyCon system are not confined to the road network, but can be defined over any 2D area, e.g., in a field or a park.

6. FUTURE WORK

The HyConExplorer is, in its current state, quite specific to Denmark. It relies on a Danish map service and can only translate GPS coordinates into Danish addresses. As noted by McCurley [20], generalized address encoding is not a simple matter, and much less so is recognizing addresses globally, as schemes vary tremendously. This is however an area where an bottom-up approach seems most appropriate—rather than attempting to create an all-encompassing solution: it is better to solve the problem gradually. The crucial component with regards to internationalization of the HyConExplorer is the "GPS position to address" resolver. This resolver relies on a database mapping positions to addresses. If such a database exists for a given locale, our approach works, especially as each locale presumably will encode addresses according to local practice, which will also be used on the relevant, local Web pages.

Currently, the GBS search blindly returns the first 10 matches—a clear area for future work would to explore more advanced forms of filtering.

Another line of future work will focus on developing a general approach for more or less automatically *adapting* SVG-based user interfaces to the characteristics of a given terminal device. It should be clear that one cannot simply take a user interface for one device, resize it, and expect it to work on another platform. Not only do different screen resolutions have consequences, there are also different modes of interaction to consider—viz. the availability of a mouse, the number of buttons, etc. We have come some part of the way in this paper by using the HyCon framework on two very diverse types of terminals. However, a general solution is left for future work. Automatic adaption of SVG-based user interfaces could relieve the application developer of much of the (currently needed) tedious work associated with deploying mobile applications on heterogeneous terminals.

7. CONCLUSION

We have in this paper described the HyConExplorer application, which provides location-based annotations, links, and guided tours for Web pages and information produced by users on the move. The architecture includes interfaces for a Sensor layer, which encapsulates GPS and other sensors.

HyCon features a novel use of XLinks for representing recorded geographical paths with annotations and links. The HyConExplorer support serendipity, allowing users to "bump into" information from the Web by moving about in the World, and make paths with multimedia annotations as well as linking relevant information. This information can be shared among users.

We have shown how a SVG based user interface provides a new way of distributing GUI objects as SVG elements such that the GUI easily adapts to devices ranging from mobile phones to tablet PCs. To support the SVG GUI on mobile phones, we have developed a SVG browser for Symbian OS, which provides the necessary support for scripting to provide an interactive GUI. The browser also utilizes SVG in a novel way to provide application integration.

One of the powerful features of the HyConExplorer is the real-time geo-based searches (GBS) via Google collecting hits within a specified nimbus around the user's GPS position. HyConExplorer is currently undergoing testing by school kids conducting project work on subjects related to areas of the city of Århus.

We see many prospects of further development of these location based hypermedia technologies. The HyConExplorer and framework will be further developed and utilized e.g., in the recently started project on Interactive School Environments under the Center for Interactive Spaces[11].

Acknowledgments

The work has been funded by the Danish National Center of IT research through project #333, ContextIT. ContextIT is associated the Center for Pervasive Computing[12]. We wish to thank all of our colleagues in the ContextIT project including the companies Euman[13] and TDC Innovationlab[14], and especially architect Gunner Kramp for his modification of the prototype tablet PC. Finally, we would like to thank the reviewers for their helpful comments.

[11] http://www.interactivespaces.net/
[12] http://www.pervasive.dk/
[13] http://www.euman.com/
[14] http://www.innovationlab.net/

8. REFERENCES

[1] G. D. Abowd. Software engineering issues for ubiquitous computing. In *Proceedings of the 21st international conference on Software engineering*, pages 75–84. IEEE Computer Society Press, 1999.

[2] K. M. Anderson, S. Moulthrop, and J. Blustein, editors. *Proceedings of the 13th ACM Hypertext Conference*, College Park, Maryland, USA, June 2002. ACM Press.

[3] N. O. Bouvin, B. G. Christensen, K. Grønbæk, and F. A. Hansen. HyCon: A framework for context-aware mobile hypermedia. *The New Review of Hypermedia and Multimedia*, 9, 2003. in press.

[4] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting Web page structure for adaptive viewing on small form factor devices. In *Proceedings of the 12th International World Wide Web Conference* [28], pages 225–233.

[5] K. Cheverst, K. Mitchell, and N. Davies. The role of adaptive hypermedia in a context-aware tourist GUIDE. *Communications of the ACM*, 45(5):47–51, 2002.

[6] B. G. Christensen, F. A. Hansen, and N. O. Bouvin. Xspect: bridging open hypermedia and XLink. In *Proceedings of the 12th International World Wide Web Conference* [28], pages 490–499.

[7] A. Crabtree, S. Benford, T. Rodden, C. Greenhalgh, M. Flintham, R. Anastasi, A. Drozd, M. Adams, J. Row-Farr, N. Tandavanitj, and A. Steed. Orchestrating a mixed reality game 'on the ground'. In *Proceedings of the 2004 conference on Human factors in computing systems*, pages 391–398. ACM Press, 2004.

[8] O. de Bruijn, R. Spence, and M. Y. Chong. RSVP browser: Web browsing on small screen devices. *Personal and Ubiquitous Computing*, 6(4):245–252, 2002.

[9] S. DeRose, E. Maler, D. Orchard, and B. Trafford (editors). XML Linking Language (XLink). W3C Recommendation 27 June 2001, W3C, June 2001. `http://www.w3.org/TR/xlink/`.

[10] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[11] A. K. Dey, D. Salber, and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction (HCI) Journal*, 16(2-4):97–166, 2001.

[12] S. Feiner, B. MacIntyre, T. Höllerer, and T. Webster. A Touring Machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. *Personal Technologies*, 1(4):208–217, 1997.

[13] J. Ferraiolo, J. Fujisawa, and D. Jackson. Scalable Vector Graphics (SVG) 1.1. W3C recommendation, W3C, Jan. 2003. `http://www.w3.org/TR/SVG11/`.

[14] C. Greenhalgh and S. Benford. MASSIVE: a collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction*, 2(3):239–261, 1995.

[15] K. Grønbæk, P. P. Vestergaard, and P. Ørbæk. Towards geo-spatial hypermedia: Concepts and prototype implementation. In Anderson et al. [2], pages 117–126.

[16] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. DOM-based content extraction of HTML documents. In *Proceedings of the 12th International World Wide Web Conference* [28], pages 207–214.

[17] C. Hage, C. S. Jensen, T. B. Pedersen, L. Speicys, and I. Timko. Integrated data management for mobile services in the real world. In *Proceedings of 29th Conference on Very Large Data Bases*, pages 1019–1030. Morgan Kaufmann, 2003.

[18] I. MacColl, D. Millard, C. Randell, A. Steed, B. Brown, S. Benford, M. Chalmers, R. Conroy, N. Dalton, A. Galani, C. Greenhalgh, D. Michaelides, T. Rodden, I. Taylor, and M. Weal. Shared visiting in EQUATOR city. In *Proceedings of the 4th international conference on Collaborative Virtual Environments*, pages 88–94. ACM Press, 2002.

[19] K. Marriott, B. Meyer, and L. Tardif. Fast and efficient client-side adaptivity for svg. In *Proceedings of the 11th International World Wide Web Conference* [27], pages 496–507.

[20] K. S. McCurley. Geospatial Mapping and Navigation of the Web. In *Proceedings of the 10th International World Wide Web Conference*, pages 221–229, Hong Kong, May 2001. W3C.

[21] P. Persson, F. Espinoza, and E. Cacciatore. GeoNotes: social enhancement of physical space. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, pages 43–44. ACM Press, 2001.

[22] S. Reich, U. K. Wiil, P. J. Nürnberg, H. C. Davis, K. Grønbæk, K. M. Anderson, D. E. Millard, and J. Haake. Addressing interoperability in open hypermedia: the design of the open hypermedia protocol. *The New Review of Hypermedia and Multimedia*, 5:207–248, 1999.

[23] B. N. Schilit, N. I. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90, Santa Cruz, CA, USA, Dec. 1994. IEEE Computer Society.

[24] J. Steinberg and J. Pasquale. A Web middleware architecture for dynamic customization of content for wireless clients. In *Proceedings of the 11th International World Wide Web Conference* [27], pages 639–650.

[25] K. Toyama, R. Logan, and A. Roseway. Geographic location tags on digital images. In *Proceedings of the 11th ACM Conference on Multimedia*, pages 156–166. ACM Press, 2003.

[26] G. M. Voelker and B. N. Bershad. Mobisaic, an information system for a mobile wireless computing environment & engineering. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90, Santa Cruz, CA, USA, Dec. 1994. IEEE Computer Society.

[27] W3C. *Proceedings of the 11th International World Wide Web Conference*, Honolulu, USA, May 2002.

[28] W3C. *Proceedings of the 12th International World Wide Web Conference*, Budapest, Hungary, May 2003. ACM Press.

[29] R. Want, B. Schilit, D. A. Norman, R. Gold, D. Goldberg, K. Petersen, J. Ellis, and M. Weiser. An overview of the PARCTab ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–43, 1995.

[30] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, Feb. 1991.

[31] E. J. Whitehead, Jr. Uniform comparison of data models using containment modeling. In Anderson et al. [2], pages 182–191.