

# UVa Bus.NET: Enhancing User Experiences on Smart Devices through Context-Aware Computing<sup>1</sup>

David Chu, Clement Song, Bei Zhang, and Marty Humphrey  
Department of Computer Science  
University of Virginia  
Charlottesville, VA 22911

## **Abstract:**

*The compact, mobile device such as the PocketPC is often regarded as little more than a handy, smaller “portal” by which to access information kept on larger back-end machines such as Exchange Servers and WWW Servers. However, the true value of the device may ultimately result from more sophisticated algorithms and approaches by which the device can fully recognize and change its behavior based on its context includes its location, remaining battery life, available networking, and user preferences. In this paper, we present “UVa Bus.NET”, a testbed at the University of Virginia for developing and evaluating general, context-aware, mobile solutions. We use .NET, .NET Compact Framework, GPS-enabled devices, and wireless networking inside buildings to notify students and professors of impending appointments and class meetings, give directions to their next appointment, and even direct them to the real-time location of the most appropriate bus to catch. The longer-term goals of UVa Bus.NET are also presented: to provide a more predictable experience to mobile device users by hiding and otherwise managing resource limitations.*

## **1 Introduction**

For a number of years, there has been the promise that ubiquitous wireless and rich, compact devices such as PocketPCs will both greatly increase worker productivity and enhance the general population’s quality of life. Three problems have most impeded the delivery of this vision. First, while wireless is increasingly being deployed across the enterprise, wireless to the consumer has emerged slowly, generally because of the cost necessary to the wireless provider. Even within the enterprise, wireless rarely extends outside of individual buildings, creating Internet voids far more often than workers should be forced to tolerate. Second, while mobile devices are becoming more powerful, they will always lag resource-rich desktops. As such, devices will perpetually play catch-up in providing the latest user experience available on desktops. Software developers for mobile devices have chosen to largely ignore resource limitations such as the reduced battery life, weak processing power, and intermittent network connectivity in hopes that the end-user somehow not notice or at least grow tolerant; most consumer mobile applications are simply “clipped” offshoots of desktop relatives. Third, mobile applications have generally failed to recognize and exploit the fact that they are indeed mobile: they can and should manage information and device behavior based on their location. In short, users think about what they see and where they are; so should the devices. However, the path to this goal is not exactly clear.

The general challenge of the mobile application is to directly acknowledge and accommodate the inherent resource limitations and exploit a first-class notion of *context*. Context-aware computing [6] [9] encompasses more than merely physical location. Any object,

---

<sup>1</sup> This work is supported in part by the National Science Foundation grants EIA-9974968, ANI-0222571, and ACI-0203960; and Microsoft Research.

physical or logical, relevant to the task at hand is part of the context. Common context categories are location, identity, user preferences, activity and time. Harnessing these context elements will allow us to explore novel spaces of user-centric services.

To provide a rich user experience, the mobile application must *recognize*, *adapt*, and *predict*. The smart device application must *recognize* the relevant aspects of its context. For example, periodic probing of the wireless networking can be used to conclude that the device is operating in an environment that does not sustain a highly-available connectivity with the Internet. The mobile application must then *dynamically adapt* to this context. For example, with only intermittent connectivity, the mobile application might change its behavior with back-end databases, choosing to otherwise cache data locally and resynchronize periodically as network connectivity is reestablished. The mobile application must also *predict* future context and plan accordingly. For example, if the network is predicted to be inaccessible in the subsequent minutes, the information that will be sought-after by the user when the network is unavailable can be obtained *while the network is still accessible and then subsequently delivered when required!* This illusion of network connectivity, leads to a more predictable user experience. Of course, in this case, the challenge is how to correctly predict the interactions with the Internet that will occur in the subsequent period of network outage.

In this paper, we describe the testbed, UVa Bus.NET, that we are developing at the University of Virginia which enables us to dynamically determine, accommodate, and predict context in mobile applications. The general idea of the testbed is that students and professors often have difficulty meeting their next classes or appointments when they rely on an unpunctual campus bus system. In our system, users are automatically alerted of appropriate arriving buses for upcoming appointments. We draw context from (1) the user's current location via a GPS enabled PDA, (2) the user's impending appointments via integration with Pocket Outlook [25], the standard PocketPC appointment scheduling software, (3) the local bus system via GPS enabled buses and our *Bus System Web Services*, and (4) local road information via digital maps to the departure bus stop. The user need not know the local bus system nor perform any additional decision-making besides application initialization. By leveraging several sources of contextual data, the system minimizes user interaction and maximizes useful information.

The rest of this paper is organized as follows. Section 2 describes the related work. Section 3 describes UVa Bus.NET. Section 4 outlines our vision for the context-aware mobile user experience. Section 5 summarizes our work.

## 2 Related Work

Active Badge [6], Xerox ParcTab [7], and Cyberguide [8] were pioneers of location-aware computing. These systems utilize knowledge of user location to facilitate a host of innovative applications. In the case of Active Badge, employees can find coworkers easily. In the case of Cyberguide, tourists are provided additional information depending on their location. These early systems understandably operated with simplified assumptions of the environment's available wireless access.

The authors of Xerox ParcTab characterize four typical uses of location context: proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions [12]. Several abstractions have since been proposed for location-awareness. Many suggest the notion of logical space distinct from physical space [11] [13] [14] [15]. The derived benefits are interaction with non-physical entities, security protection, and increased semantics.

Extending upon the ideas of location-awareness, context-awareness encompasses any "person, place, or object that is considered relevant to the interaction between a user and an application" [10]. Several frameworks have been created for context-aware application

development. Nexus supports “augmented worlds”, allowing interactions among nearby entities. The authors of [13] provide similar base components on which to model spatial relationships. The Context Toolkit [10] provides even more general context constructs, though some loss of specific context capability entails.

University of California San Diego’s ActiveCampus [19] offers location-aware user experiences in the campus setting. Users may know the whereabouts of colleagues, and post location-based graffiti. ActiveCampus’ location-awareness facilitates an “augmented world” similar to that of Nexus. TEA [17] present a multi-sensored approach, somewhat in the spirit of our method. However, their attempts to establish solely sensor-based context removed from the application domain proves difficult.

The concept of tracking bus locations is certainly not new. Several cities have implemented bus tracking and forecasting systems based on GPS [3]. Often these systems have public monitors at bus stops reporting next projected arrival times or provide websites to query bus arrivals. Interaction requires substantial work on the user’s behalf (going to the right bus stop or website), and necessitates bus system knowledge for any reasonable use. These solutions also are not tailored to the specific user.

The authors of [5] and [4] report algorithms to accurately forecast the arrival of transit vehicles. Our work uses forecasting implementations provided by MapPoint Web Services [24]. While accurate bus forecasting is an essential element to our work, it alone does not free users from complex decision making. Few have attempted meaningful integration of bus system knowledge with mobile devices. While [1] and [2] detail some efforts, they are arguably limited by failing to realize critical observations (Section 3.1): these systems still require too much of the user in return for too little.

### **3 UVa Bus.NET Design**

#### **3.1 Criteria**

Prior to the development of UVa Bus.NET, we observed that users can not be expected to periodically poll the device nor provide redundant information. Users might not even know anything about the local bus system! Yet users demand the most relevant information. Luckily, we can draw from the strength of the mobile device: its close coupling with the user’s context.

The tax of interacting with the device versus the value of information gained must be as low as possible. With this cardinal rule in mind, we designed a system which adopts a *lazy user* philosophy. The system should initiate interaction with the user only when it predicts interesting events. We tailor information specifically to the user, utilizing as many sources of relevant contextual data as possible. We use (1) a GPS enabled PDA, (2) Pocket Outlook appointments which include time and destination, (3) automated bus transit information, and (4) local road geography information via digital maps to the departure bus stop. These provide four sources of highly relevant context.

We provide event scheduling via existing familiar applications. We integrate with Pocket Outlook, the PocketPC appointment scheduling software. The user need not ever redundantly schedule appointments. ActiveSync [27] already enables synchronization between desktop Outlook and Pocket Outlook; the user may completely avoid the tedium of entering information via the device.

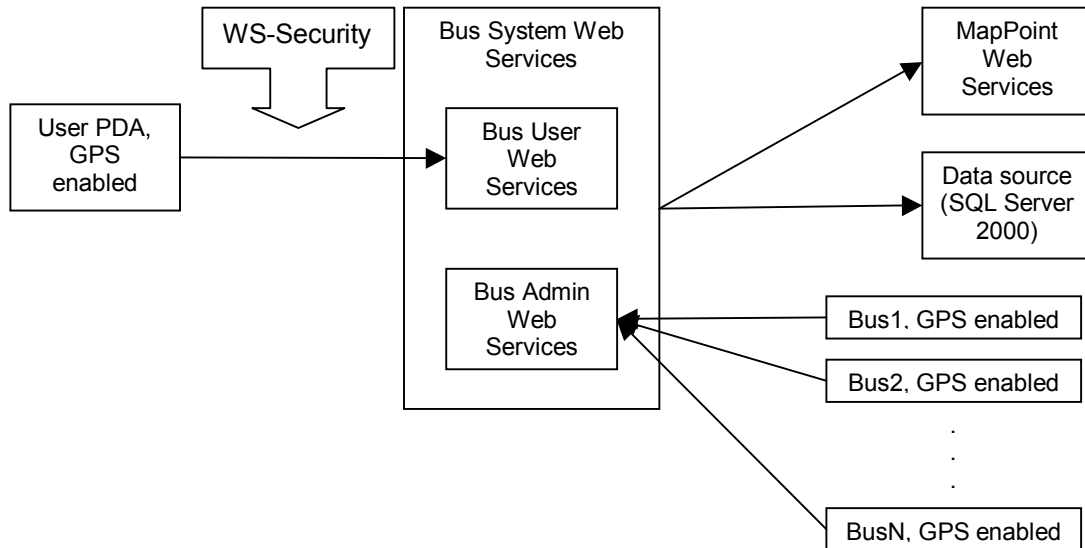


Figure 1: Bus System Architecture

### 3.2 Operation

Figure 1 presents the architecture. We use web services for all communication. Web services define a platform-independent remote method invocation protocol that offers application interoperability. Web services use open XML, SOAP, WSDL, and UDDI standards [20]. While not necessarily tied to the Internet, many web services often are exposed through HTTP. We deploy our web services on Microsoft Internet Information Services webserver. Our web services, *Bus System Web Services*, are built atop of the Microsoft .NET Framework [22]. This Windows software infrastructure manages much of the application-independent plumbing and provides a host of commonly used base classes. The client application runs on the .NET Compact Framework enabled PocketPCs. The .NET Compact Framework [23] is a modified version of the full framework specifically targeted at resource constrained platforms, such as mobile devices. The client application runs on any platform which supports the .NET Compact Framework.

UVaBus.NET follows the WS-Security standard [21]. This standard applies additional extra headers to every incoming and outgoing web service message to ensure security. On the server end, *Bus System Web Services* use the Microsoft Web Services Enhancements (WSE) [26]. On the client end, the client application adds the requisite header information with each request. At this time, the .NET Compact Framework does not directly support WSE.

The client application initiates a request to the *Bus User Web Service* with the user's location, destination and appointment time (We discuss the timing of this request in Section 3.4). The *Bus User Web Service* then responds with a travel itinerary containing (1) the optimal departure and arrival bus stop, (2) the corresponding departure and arrival times, and (3) the walking time from the user's current location to the departure stop and from the arrival stop to the appointment location. This itinerary is calculated from consulting the bus system data source, which contains static bus stop and bus line information, as well as dynamic bus location data. GPS enabled buses periodically report their location by using *Bus Admin Web Services*. Additionally, *Bus User Web Services* consult *MapPoint Web Services* to forecast arrival times, and provide a fallback cached map to the departure bus stop.

Upon receiving the travel itinerary, the client application sets a timer which will alert the user in time to catch the bus. At that time, the user may additionally request a map displaying the

route to the departure bus stop. Wireless connectivity is not necessary to retrieve this map; a cached map is available, though if possible, the client application will retrieve a fresh map.

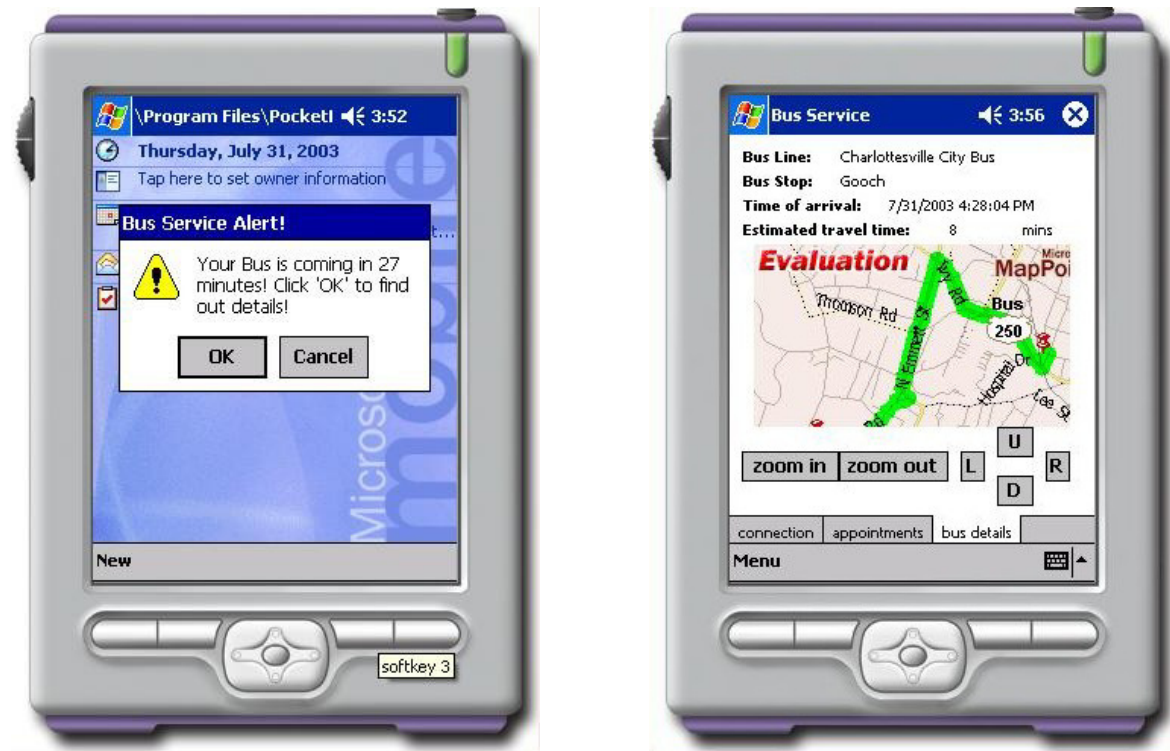


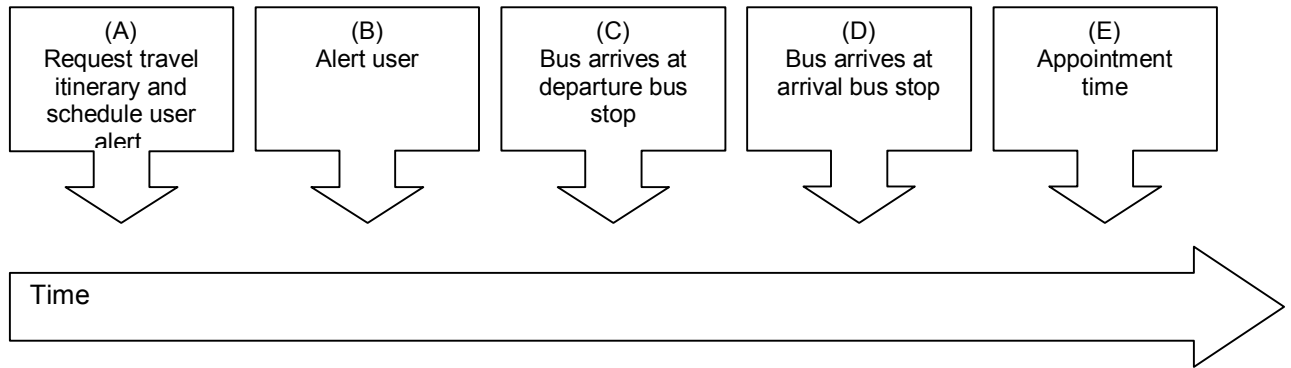
Figure 2: Screenshots of UVaBus.NET. The left image shows a bus alert. The right image shows a map with the highlighted route from the user's current destination to the departure bus stop.

### 3.3 Use Case

Upon initializing the application, the user sets some credentials to seemingly authenticate once with the server. In actuality, the client application adds WS-Security soap headers to every soap message. After initialization, the user's work is done and the application is hidden from the user's view.

The client application next engages the user's attention at a time predicted according to the prediction algorithm described in Section 3.4. The user is notified of the estimated arrival time of the bus and provided with a map of how to get from his current location to the departure bus stop. This repeats for every appointment on which a meaningful destination address is provided and the "Remind Me" flag, a native Pocket Outlook indicator, is set.

Currently, the modeled bus system is the University of Virginia transit system. In order to support other bus systems, only a data source substitution is necessary.



**Figure 3: Events sequence. If (A) occurs too early, then the departure bus stop may be inaccurate, whereas if (A) occurs too late, the user may miss the bus.**

### 3.4 Prediction Algorithm

The challenge is to develop a prediction algorithm for timing the travel itinerary request. As Figure 3 shows, catching the bus is contingent upon the timing of event A. The user’s future location is unknown, and it is infeasible to gratuitously request multiple travel itineraries, due to expense of computation and device resource limitations. We employ the heuristic that the user’s last appointment location is likely also his point of departure for the following appointment. This assumes (1) minor movement upon arriving at an appointment (which is often the case in our academic scenario) and (2) a complete schedule.

The algorithm further masks lack of GPS connectivity indoors by remembering the last active GPS signal. This enables the user to still receive an accurate travel itinerary when in buildings.

## 4 UVaBus.NET as a TestBed

UVaBus.NET withstands some environmental obstacles. Currently the system caches map information which has a high chance of relevancy (Section 3.2). This relieves the need for wireless connectivity at certain points while still masking the loss to the user. Similarly, the prediction algorithm handles the loss of GPS gracefully (Section 3.4).

Yet several aspects urge revisiting. Any system which purports to gracefully accommodate network loss yet not sacrifice information availability must predict network loss. Current applications rarely strive for this goal. This cross-application problem suggests a middleware approach. Knowledge of the wireless network topology would enable the system to cache relevant information to the user before he/she exits the networked region. Upon requesting information in the unnetworked region, the user still receives the requisite, albeit possibly stale, information. For example, UVaBus.NET might cache a selection of probable bus stops and forecasted bus arrival times when it predicts a user will lose wireless connectivity. The necessary topology would be constructed by recording the movements of wireless mobile device users, and tracking corresponding signal fading and loss.

As an additional freshness assurance mechanism, a human-routing system, such as UVaBus.NET, might even direct users through wireless hotspots en route. The user would then be able to choose to either pursue the shortest path for speed or the circuitous path for up-to-date information. Any such comparisons are subject to an analysis of the usefulness of the cached data versus the cost of retrieving fresh information [16].

Additionally, we envision location-aware Bluetooth as a facilitator of useful information exchange, especially in the unnetworked regions. Users in the same geographic proximity are likely to have information relevant to each other. For example, a passer-by might well have more recent bus forecasts which are not otherwise available due to the lack of Internet access. The challenge is to identify useful information, and propagate the data to all interested parties without relying on opportunistic chance encounters.

Overall, the potential of the research platform is to enable these and other approaches that provide a *more predictable* execution of applications. By better anticipating user patterns, we can deliver information when the user needs it and gains the most value from it, irrespective of the limitations of the operating environment. Ultimately, it is through these enhancements that we believe the most value can be delivered to the end-user.

## 5 Conclusions

The distinct lack of ubiquitous wireless coupled with mobile device resource limitations demands addressing. We have presented UVaBus.NET, a context-aware system which substantially eases bus system use. We utilize several points of context so that users are alerted only of relevant information. The result is a system where users perform relatively little decision making while saving time catching the bus.

Approached in a broader context, we position UVaBus.NET as a testbed to explore further techniques to marshal the right information to the right users. We have enumerated some of these future directions. Most notably, knowing the network topology opens the possibility of a host of prediction schemes which deliver timely information to the user.

## References

- [1] S.D. MacLean, D. J. Dailey. **Wireless Internet Access to Real-Time Transit Information.** *Transportation Research Record*, Vol. 1791, 2002, pp. 92-98
- [2] S.D. MacLean, D. J. Dailey. **Real-time Bus Information on Mobile Devices.** *Proceedings of IEEE Intelligent Transportation Systems Conference 2001*, Oakland, California, 2001.
- [3] MyBus.org, [www.mybus.org](http://www.mybus.org)
- [4] F.W. Cathey, D.J. Dailey. **A Prescription for Transit Arrival/Departure Prediction Using Automatic Vehicle Location Data.** *Transportation Research Conference*, to appear first quarter 2003.
- [5] D. J. Dailey, Z.R. Wall, S.D. MacLean, F.W. Cathey. **An Algorithm and Implementation to Predict the Arrival of Transit Vehicles.** *Proceedings of the IEEE Intelligent Transportation Systems Conference 2000*, Dearborn, Michigan, 1-3 October 2000.
- [6] R. Want, A. Hopper, V. Falcão, J. Gibbons. **The active badge location system.** *ACM Transactions on Information Systems*, Vol. 10, No. 1, Jan 1992, pp 91-102
- [7] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, J. Ellis, D. Goldberg, M. Weiser. **The ParcTab Ubiquitous Computing Experiment.** *Xerox PARC Technical Report CSL-95-1*, March 1995.
- [8] S. Long, D. Aust, G. Abowd, C. Atkeson. **Cyberguide: Prototyping Context-Aware Mobile Applications.** *Proceedings of the conference on Human Factors in Computing Systems (CHI 1996)*, Vancouver, Canada, 1996, pp 293-294.
- [9] K. Cheverst, N. Davies, K. Mitchell, A. Friday, C. Efstratiou. **Developing a context-aware electronic tourist guide: some issues and experiences.** *Proceedings of the SIGCHI conference on Human factors in computing systems*, April 2000.
- [10] A.K. Dey. **Understanding and Using Context.** *Personal and Ubiquitous Computing*, Vol. 5, No. 1, February 2001, pp 4-7.
- [11] S. Duri, A. Cole, J. Munson, J. Christensen. **An approach to providing a seamless end-user experience for location-aware applications.** *Proceedings of the first international workshop on Mobile commerce*, July 2001.

- [12] B. Schilit, N. Adams, R. Want. **Context-aware computing applications.** *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on.*, 8-9 December 1994, pp 85 -90.
- [13] A. Dix, T. Rodden, N. Davies, J. Trevor, A. Friday, K. Palfreyman. **Exploiting space and location as a design framework for interactive mobile systems.** *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 3, September 2000.
- [14] A. K. Narayanan. **Realms and states: a framework for location aware mobile computing.** *Proceedings of the first international workshop on Mobile commerce*, Rome, Italy, July 2001, pp 48-54.
- [15] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, M. Schwehm. **Next century challenges: Nexus—an open global infrastructure for spatial-aware applications.** *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, Seattle, Washington, August 1999, pp 249-255.
- [16] Vittoria de Nitto Personè, Vincenzo Grassi, Antonio Morlupi. **Modeling and evaluation of prefetching policies for context-aware information services.** *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking*, Dallas, Texas, October 1998, pp 55-65.
- [17] Hans W. Gellersen, Albercht Schmidt, Michael Beigl. **Multi-sensor context-awareness in mobile devices and smart artifacts.** *Mobile Networks and Applications*, Vol. 7, No. 5, October 2002.
- [18] Schilit B, Theimer M. **Disseminating active map information to mobile hosts.** *IEEE Network*, Vol. 8, 1994, pp 22–32.
- [19] W. G. Griswold, R. Boyer, S. W. Brown, T. M. Truong, E. Bhasker, G. R. Jay, and R. B. Shapiro. **Using Mobile Technology to Create Opportunistic Interactions on a University Campus.** *UbiComp 2002 Workshop on Supporting Spontaneous Interaction in Ubiquitous Computing Settings*, Technical Report CS2002-0724, Computer Science and Engineering, UC San Diego, September 2002.
- [20] Web Services, <http://www.w3.org/2002/ws/>
- [21] WS-Security, <http://www-106.ibm.com/developerworks/library/ws-secure/>
- [22] .NET Framework, <http://msdn.microsoft.com/netframework/>
- [23] .NET Compact Framework, <http://msdn.microsoft.com/vstudio/device/compactfx.aspx>
- [24] MapPoint Web Services, <http://www.microsoft.com/mappoint/net/>
- [25] Pocket Outlook Object Model, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcepoom/html/ceoriPocketOutlookObjectModel.asp>
- [26] Web Services Enhancements, <http://msdn.microsoft.com/webservices/building/wse/default.aspx>
- [27] ActiveSync, <http://www.microsoft.com/windowsmobile/resources/downloads/pocketpc/activesync37.msp>