Journal of Internet Services and Applications manuscript No.
(will be inserted by the editor)

R&D Challenges and Solutions for Mobile Cyber-Physical Applications and Supporting Internet Services

Jules White · Siobhan Clarke · Christin Groba · Brian Dougherty · Chris Thompson · Douglas C. Schmidt

the date of receipt and acceptance should be inserted later

Abstract *The powerful processors and variety of sensors in new and planned mobile Internet devices, such as Apple's iPhone and Android-based smartphones, can be leveraged to build cyber-physical applications that collect sensor data from the real world and communicate it back to Internet services for processing and aggregation. This article presents key R&D challenges facing developers of mobile cyber-physical applications that integrate with Internet services and summarizes emerging solutions to address these challenges. For example, application software should be architected to conserve power, which motivates R&D on tools that can predict the power consumption characteristics of mobile software architectures. Other R&D challenges involve the relative paucity of work on software and sensor data collection architectures that cater to the powerful capabilities and cyber-physical aspects of mobile Internet devices, which motivates R&D on architectures tailored to the latest mobile Internet devices.*

J. White
Vanderbilt University
E-mail: jules@dre.vanderbilt.edu

B. Dougherty
Vanderbilt University
E-mail: briand@dre.vanderbilt.edu

C. Thompson
Vanderbilt University
E-mail: cthompson@dre.vanderbilt.edu

D.C. Schmidt
Vanderbilt University
E-mail: schmidt@dre.vanderbilt.edu

C. Groba
Trinity College Dublin E-mail: grobac@cs.tcd.ie

S. Clarke
Trinity College Dublin E-mail: siobhan.clarke@scss.tcd.ie

Keywords Cyber-physical Applications, Mobile Computing · Sensor Networks · Software Product-lines · Model-driven Engineering · Software Architectures

1 Introduction

Emerging trends and opportunities. Mobile Internet devices, such as the iPhone and Android-based phones have become incredibly popular. For example, Apple has sold over 33.8 million iPhones and the Motorola Droid phone sold over 400,000 units in its first week. The proliferation of these devices is expected to increase, *e.g.*, the Android platform will likely be available on dozens of phones in scores of countries within a year.

The broad dissemination of these mobile Internet devices, their accelerated processing power, range of sensors, and pervasive cellular connections make them ideal platforms for building novel *mobile cyber-physical applications*. A cyber-physical application is a computer system that processes and reacts to data from external stimuli from the physical world and make decisions that also impact the physical world [31]. Traditional cyber-physical applications include flight avionics, electronic medical devices, and power grid control systems. Since cyber-physical applications can impact the physical world and must respond to physical events, they often require rigid performance and safety assurance.

Mobile Internet devices possess a variety of sensors (such as ambient light sensors, accelerometers, GPS sensors, microphones, and cameras) that cyber-physical applications can use to sense environmental stimuli, When cyber-physical applications are combined with Internet services, they can detect context information from user environments and react to social network

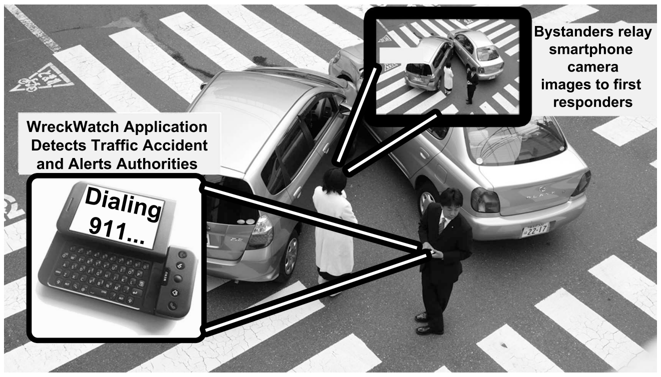
Bystanders relay smartphone camera images to first responders
WreckWatch Application Detects Traffic Accident and Alerts Authorities
Dialing 911...

Fig. 1 Mobile Cyber-physical Application to Detect and Report Traffic Accidents

information derived from the user contacts, Facebook account, and other social networking databases. Combining data that is both immediately present in device environments with information streams and processing power available through the Internet facilitates novel mobile cyber-physical applications.

R&D efforts are tapping into the significant potential of these devices. For example, developers have built cyber-physical applications and Internet services to detect and track user activities for health purposes [29], track and analyze CO_2 emissions [11], detect traffic accidents and provide situational awareness services to first responders [32,16] (shown in Figure 1), measure traffic and derive road quality [28,24], and monitor cardiac patients [18].

Compared with developing specialized hardware and software solutions, building cyber-physical applications atop mobile Internet devices offers a range of benefits with equivalent functionality, including:

- Maintenance of customized hardware and software solutions, such as wireless sensor networks, has historically been a key issue to address [20]. Not only must sensors be kept in working order, they must also have adequate battery power. In contrast, cyber-physical applications based on mobile Internet devices can rely on their owners to maintain and charge the devices.
- Complex networking strategies have been required in traditional custom hardware solutions to communicate data back to base stations for compute-intensive processing [21]. In contrast, mobile cyber-physical applications can communicate with Internet services using standard IP networking to transmit data for aggregation and receive processed results.
- Conventional sensor network nodes are often stationary due to the high power cost of movement. In contrast, cyber-physical applications built on mobile Internet devices travel with their owners and can take measurements at multiple locations throughout the day. Moreover, monitoring human-centered phenomena (such as traffic congestion) can be easier and less costly when the sensors travel with mobile Internet device users.

Open R&D challenges. Despite the benefits of building mobile cyber-physical applications atop mobile Internet devices and Internet services, however, various open R&D challenges limit their development and deployment in practice. This article presents key R&D challenges for mobile cyber-physical applications and supporting Internet services, including the following:

1. **Optimizing power consumption** early in the application development lifecycle is hard, which makes it expensive and time-consuming to develop applications that run for extended periods of time on mobile Internet devices.
2. **Avoiding costly overprovisioning** to support Internet data processing services for mobile cyber-physical applications is hard since average processing loads can be significantly lighter than peak load and overprovisioning for occasional peak loads wastes resources for common usage conditions.
3. Developing a configurable cyber-physical software product for a wide range of targets is hard due to the **variations between target platforms** that make it hard to optimize the software for each platform and ensure that non-functional constraints are met.
4. **Integrating external sensors** to exploit the benefits of combining conventional sensor solutions and emerging cyber-physical applications is hard due to different resource constraints and device capabilities of mobile Internet devices and traditional sensor platforms.

This article summarizes efforts by ourselves and others to address these challenges and is organized as follows: Section 2 summarizes a motivating example of a cyber-physical application and supporting Internet services for detecting traffic accidents we developed; Section 3 explores key R&D challenges and solutions based on our motivating example; Section 4 describes other emerging R&D opportunities in mobile cyber-physical applications and Internet services; and Section 5 presents concluding remarks and lessons learned.

2 Motivating Example: WreckWatch

To motivate the capabilities available to mobile cyber-physical applications built on Internet devices, this section describes the structure and functionality of *WreckWatch*, which is multi-tier cyber-physical application

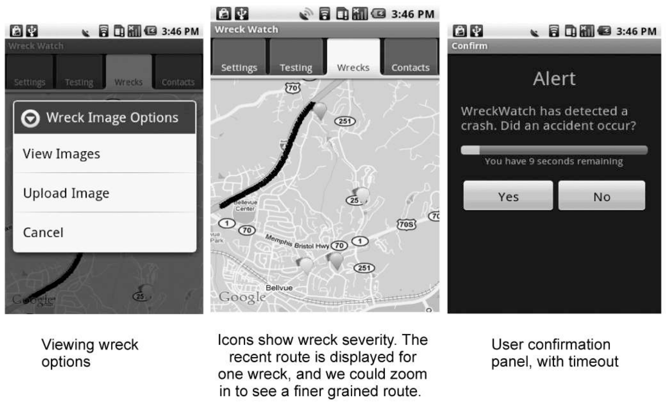
Wreck Image Options
View Images
Upload Image
Cancel
Settings
Testing
Wrecks
Contacts
Alert
WreckWatch has detected a crash. Did an accident occur?
You have 9 seconds remaining
Yes
No
Viewing wreck options
Icons show wreck severity. The recent route is displayed for one wreck, and we could zoom in to see a finer grained route.
User confirmation panel, with timeout

Fig. 2 The WreckWatch Traffic Accident Detection Application for Mobile Internet Devices

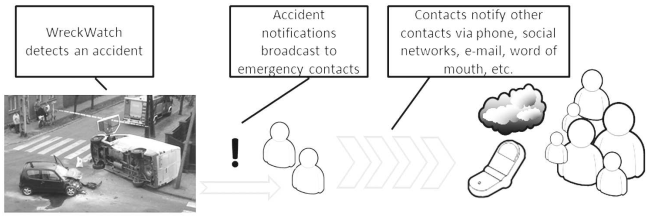
WreckWatch detects an accident
Accident notifications broadcast to emergency contacts
Contacts notify other contacts via phone, social networks, e-mail, word of mouth, etc.

Fig. 3 The WreckWatch Communication Paradigm

for detecting traffic accidents. WreckWatch is one of numerous open-source[1] sensor mobile cyber-physical applications we developed on the Google Android and iPhone platforms. We use WreckWatch as a motivating example in this article since the range of challenges we faced developing it apply to many mobile cyber-physical applications and supporting Internet services, as well.

WreckWatch is based on the premises that mobile Internet devices now contain sufficiently sophisticated sensors and networking capabilities that software applications can be built on top of them to serve as portable *black boxes*. These black boxes can travel with drivers and/or passengers to help detect traffic accidents and provide critical situational awareness information to first responders. Unlike existing traffic accident detection systems, such as OnStar, WreckWatch is not tethered to a particular vehicle and can travel seamlessly with its owner.

WreckWatch runs as a background service on Google Android and polls the accelerometer and GPS for current speed and acceleration information. At speeds above a predefined threshold, WreckWatch starts feeding speed and deceleration information into a mathematical accident prediction model. If the model indicates that the current pattern of deceleration and speed is indicative of a traffic accident, WreckWatch reports the accident to a central accident response server.

As shown in Figure 2, WreckWatch does not immediately report the accident to the central server. Instead, a dialog is presented to the user asking if an accident has actually occurred so users can cancel an accident report for a false positive. If the user does not respond to the dialog before a predetermined timeout, WreckWatch submits an accident report.

[1] WreckWatch and our other applications for mobile Internet device sensor networks are available as open-source from `code.google.com/p/vuphone`.

WreckWatch uses a phone-based client and a central Internet service to disseminate accident information to first responders, emergency contacts derived from social data, and other motorists using a variety of voice and data channels. Reported accidents are plotted by the Internet service on Google Maps and made available to first responders and other motorists via the WreckWatch client application. The central accident reporting service uses the Asterisk Private Branch Exchange (PBX) to automatically place emergency calls to 911 and dynamically provision an accident hot-line for friends and family of the accident victims. WreckWatch can also use the emergency hotline to automatically send text messages to a list of emergency contacts when wrecks occur.

Motorists can use WreckWatch's multimedia upload capabilities to provide first responders with detailed visual and audio information about wrecks. Likewise, accident bystanders can use their device cameras to take pictures or videos of the accident and share them via the central server with first responders, as shown in Figure 3. WreckWatch's ability to use networks of bystanders to submit imagery of accidents exemplifies its cyber-physical capabilities.

3 Overview of R&D Challenges and Solutions

The capabilities of WreckWatch described in Section 2 incur a number of demands on the software architecture and Internet services that support it. For example, careful design is required to ensure it does not consume too much power, overconsume network bandwidth, or overwhelm central servers. This section describes key R&D challenges and presents promising solution approaches that we and others are developing to address these challenges. We selected these challenges based on our experience developing WreckWatch and other mobile cyber-physical applications and supporting Internet services described in Section 4. Although the solutions are paired with individual challenge problems, many common themes, such as the use of model-driven engineering, crosscut the solutions.

3.1 **Challenge**: Optimizing Power Consumption of Mobile Cyber-physical Software Early in the Lifecycle.

Context. Although the latest mobile Internet devices have powerful processors (*e.g.*, the Nexus One contains a 1 Ghz processor), cyber-physical application software can quickly use this computational power at the expense of increased power consumption. Whereas simple applications written for previous generation devices consumed power slowly enough for devices to function for days between charges, current mobile cyber-physical applications use so many sensors that device batteries can be exhausted within hours (*e.g.*, the Apple iPhone maximum battery life with continuous 3G data connection usage is ∼5 hours). When a mobile cyber-physical application combines heavy processor usage with power drain from a combination of sensors and data transfer, battery life can be very short.

For example, WreckWatch runs continuously as a background service on Google Android. In some of our initial implementations, the highest possible update rate provided by Android was used to receive GPS location updates. The combination of processor usage for our accident prediction model and GPS polling completely drained the battery of an HTC G-1 phone in under two hours. This rate of power drain presents a major problem for a cyber-physical traffic accident detection application designed to run continuously.

On software platforms that support multi-tasking, such as Google Android and Palm Pre devices, cyber-physical application software may be required to share power, computing, and sensor resources with multiple other applications. The cyber-physical application software must draw power slowly enough for the device to remain charged all day, while simultaneously allowing users to place phone calls, browses the Web, and check email. It is therefore critical that cyber-physical application software be designed so that it does not become such a significant power burden on a device that owners are unwilling to run it.

Open problems. It is hard for developers of mobile cyber-physical applications to predict the power consumption of a software architecture early in the development process. Our experience with WreckWatch showed that the sensor software must be implemented, deployed, and tested on the target hardware to determine its power consumption characteristics. The inability to predict power consumption during the design stage was problematic since design changes late in the development process are more costly.

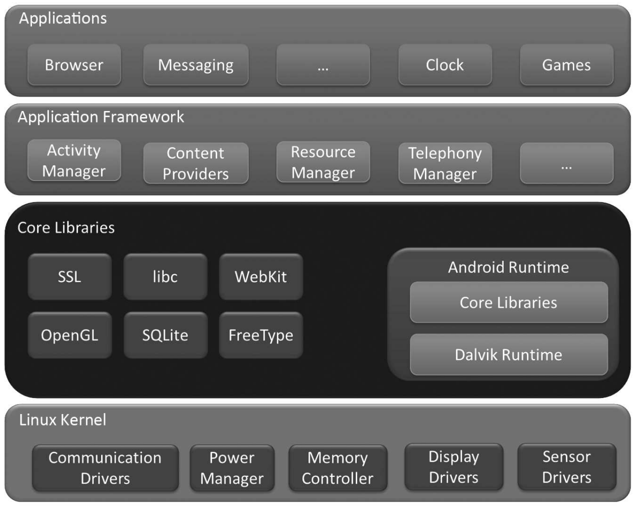

Fig. 4 Layering of Android Middleware and OS Abstractions

Many hard-to-predict platform factors play a role in determining how a particular software design consumes power. Middleware and OS task scheduling and memory utilization strategies can affect how a software architecture consumes power [12]. Likewise, networking implementation details, such as design decisions in the MAC layer of the OS [1], also play an important role. Moreover, diversity in hardware (such as variation in sensors) can consume power at different rates across devices, *e.g.*, using GPS on one device may be much more costly than on another.

Conventional cyber-physical applications with custom hardware and software typically use lightweight OS and middleware layers, such as TinyOS [19], that provide low-level programming APIs that tightly-couple the software to the hardware. This minimalistic approach complicates software development, but allows for more control over how power is consumed. The increased control over how power is consumed makes it easier for developers to forecast power consumption.

In contrast, cyber-physical applications built on mobile Internet devices are perched atop an intricate set of OS and middleware layers that expose high-level APIs to developers and simplify software development. For example, Figure 4 shows the depth of Android's layers of middleware and network stack abstractions, which makes it hard to predict the power consumption of each layer. For most applications, such as games or user productivity applications that are not concerned with power consumption, these higher-level APIs are ideal. For cyber-physical applications that must conservatively consume power and always be on, these intermediate layers of abstraction make managing power consumption harder.

Emerging solution → Model-driven power consumption analysis. Model-driven engineering (MDE) tools [30] help specify high-level cyber-physical software architectures rapidly and then generate architecture emulation code to run on target devices and to obtain rough estimates of power consumption. By utilizing MDE tools along with device- or platform-specific code generation, it becomes possible to address the challenge of predicting mobile cyber-physical application power consumption early in the development cycle. These MDE tools allow developers to analyze and evaluate potential designs on a physical device before commiting to a specific architecture.

The key benefits that an MDE-based power analysis approach provides are (1) the ability to generate emulation or simulation code for an architecture before it is implemented and (2) the capability to refine the generated code as the system becomes more precisely understood. Code generation is critical since it allows developers to rapidly test power consumption characteristics of an architecture before committing to the cost of implementing a design. Refinement is another important property because it allows the power consumption estimates to become more precise as the development process progresses. Increased precision in the power consumption estimates allows developers to not only tune high-level architectural properties but more fine-grained implementation decisions.

For example, the *System Power Optimization Tool* (SPOT) [32] is an MDE tool that models mobile software architectures and generates emulation code. The SPOT visual modeling environment (based on the Eclipse IDE) allows developers to model the high-impact aspects of their designs before commiting to a particular implementation. Designers can specify sensor, CPU, networking, and OpenGL utilization. SPOT then generates Java code for Android devices that allows developers to run and analyze their designs without the tedium of manual implementation. It also allows developers to perform continuous integration testing [14], which generated emulation code is incrementally replaced by actual cyber-physical application logic as the software evolves. This model-driven continuous integration process helps application developers increase the accuracy of their models throughout the software lifecycle.

SPOT provides developers with a rough idea of how their design will perform as early and with as little overhead as possible. This MDE tool also helps pierce through multiple layers of abstraction to predict power consumption accurately. Since SPOT produces actual device code, speculation of how these layers will affect power consumption is unnecessary because middleware interaction is accounted for in the resulting data.

3.2 **Challenge**: Avoiding Costly Overprovisioning.

Context. Although mobile Internet device processing power has improved significantly, some cyber-physical data processing tasks (such as high-speed image processing) are still not suitable for a mobile application. Likewise, timely completion of complex tasks (such as location-based searches) is not possible on mobile Internet devices due to their limited memory, processing speed, and power compared to server-based infrastructure. For example, data aggregation of terabytes of information or image manipulation on thousands of multi-megapixel files (*e.g.*, aggregating camera images to create 3D maps) are beyond the capabilities of today's mobile Internet devices.

One approach to handling tasks that cannot be accomplished by mobile applications is to use Internet services to aggregate and process data for the mobile cyber-physical applications. These Internet services run on supporting servers in a cloud. Data harvested by cyber-physical applications from mobile device sensors can be sent to these Internet services, which aggregate and process the data before sending the results back to the devices.

For example, WreckWatch uses a centralized Internet service to provide enhanced emergency response services for the mobile Internet devices. WreckWatch's Internet service can collect and disseminate images and video from an accident for emergency response teams. WreckWatch's Internet service also provides more computationally taxing functions, such as the ability to dynamically provision emergency response VoIP hotlines through its integrated Asterisk PBX. These features of WreckWatch are only possible through the use of client-side accident detection and imaging code in the mobile cyber-physical application and server-side media aggregation and PBX functionality.

Open problems. Using Internet services to support mobile cyber-physical applications requires developers to address the challenging problem of determining how to efficiently provision servers to run the services. Conventional approaches to server provisioning, such as worst-case capacity planning, over-engineer computing platforms to ensure quality-of-service (QoS) requirements are met during peak load conditions. Due to the excess capacity built into the computing platform, however, many computing resources are idle under non-peak load conditions. With mobile cyber-physical applications, processing load may change dramatically during the day as users become stationary or go to sleep.

For example, WreckWatch's peak loads are during rush hour traffic periods when more cars are on the road and more accidents occur. At night or when users have finished their morning commutes to work, the supporting Internet service is substantially less loaded. Unplanned occurrences, such as incliment weather, may also cause spikes in the processing load of the Internet service that are far above the average. This wide variation in processing load makes it hard for operators to provision infrastructure that provide the low response time needed by cyber-physical applications, but that also does not require costly overprovisioning.

As power consumption becomes an increasingly important issue, service providers will not be able to overprovision as easily due to regulation and higher power costs. In 2003, it was estimated that data centers consumed 22TWh of power [23]. Power consumption and cooling are expected to become more important and expensive for data centers in the future [4].

Emerging solution → Cloud computing and resource auto-scaling. Cloud computing uses virtualization [5] to allow dynamic provisioning of OS images in a data center. Operators have traditionally purchased individual hardware platforms for each OS image. With cloud computing, virtual OS images are co-located on the same hardware, allowing more efficient use of hardware. These flexible OS image allocation techniques can deploy Internet services into production environments much faster and often reduce initial deployment cost.

Manually configured cloud computing environments are often inefficient platforms for Internet services that support mobile cyber-physical applications. For example, these applications have periods of increased workload that are not always foreseen and which can fluctuate significantly. Additional OS images must therefore be deployed in the cloud to handle these periods of increased activity. When the workload subsides, however, the additional OS images are idle, wasting valuable resources (such as power) and increasing operation costs.

Computing clouds, such Amazon's Elastic Compute Cloud (EC2), have recently introduced automated cloud scaling [33]. EC2 uses auto-scaling to respond to fluctuations in the computational needs of the Internet service utilizing the cloud. For example, if a traffic accident occurs, WreckWatch's Internet service could see drastically increased loads.

Figure 3.2 shows how automated cloud scaling allows on-demand deployment of additional computational resources to handle increased workloads. The type of OS image and resources deployed can also be tailored for particular application needs. For example, if a supporting Internet service requires substantially increased processing power—but only marginally increased memory availability—then an OS instance with precisely the needed resources can be provisioned. After the workload returns to the normal state, the additional resources are

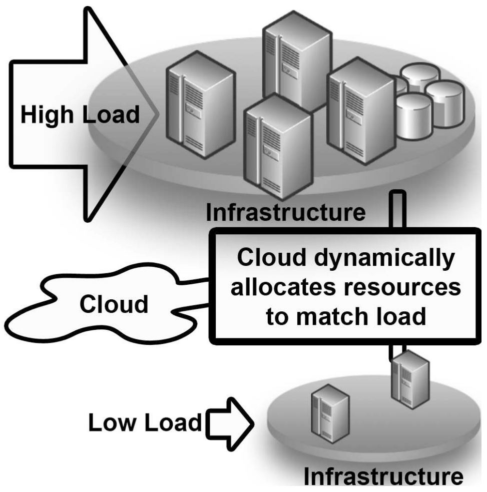

Fig. 5 Cloud Computing Can Dynamically Scale Resource Allocation to Meet Load

released. As a result, the size of the cloud remains appropriate for the current workload, regardless of unforeseen fluctuations, thereby helping to minimize power consumption and operational cost.

3.3 **Challenge**: Addressing Platform Variations

Context. Unlike the desktop and server operating system market, there may not be a dominante smartphone operating system vendor. Gartner estimates that Windows Mobile, Blackberry OS, iPhone OS, and Google Android will each have roughly ≈13% of the market in 2012. Symbian is expected to have the largest share of the international market with ≈30%. Many developers and organizations will therefore likely create and maintain cyber-physical applications that are targeted for multiple mobile Internet device operating systems and versions.

For example, multiple versions of Google Android were released during the development of WreckWatch. Our development efforts initially targeted Android 1.0 and HTC's G1, which was the only Android hardware available at the time. Since the initial implementation was finished, Google has released Android 1.5 and Android 2.0 and there are now at least five different Android devices by Motorola and HTC. We have also begun the process porting WreckWatch to the iPhone.

Open problems. As shown in Figure 3.3, there is significant complexity involved in managing the variability of cyber-physical software and determining the appropriate software configuration for a given mobile platform. For example, WreckWatch can run as a background service on Android in parallel with other applications. In contrast, WreckWatch cannot run concur-

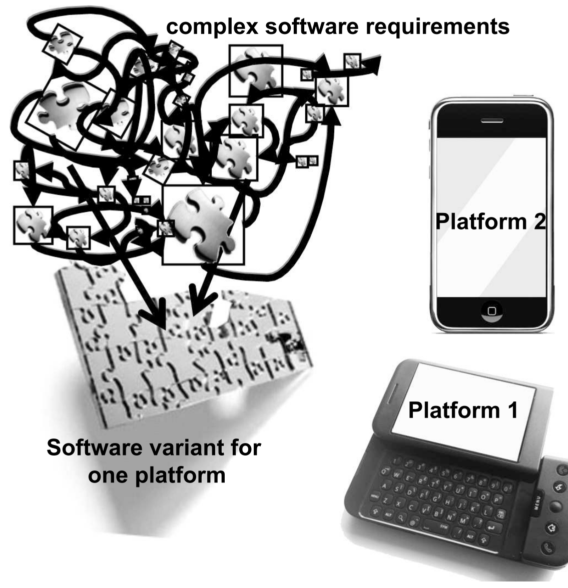
complex software requirements
Platform 2
Platform 1
Software variant for one platform

Fig. 6 Complexities of Targeting Multiple Platforms

rently with other applications on the current version of the iPhone and must be redesigned as a modal application. Each additional variation in platform design increases development complexity.

Even within a single OS platform there can be variations across versions and devices that add development complexity. For example, the 2.0 release of Android provides a Bluetooth API that can be used by a cyber-physical application to communicate with external sensors, whereas prior versions did not. The 3.0 release of iPhone added the ability to have notifications asynchronously delivered to applications that were not running. This notification API makes notifying WreckWatch client users of new accidents easier than on prior versions of the iPhone OS.

Although mobile Internet devices have signficant processing capabilities, certain resources (such as battery power) are still limited. It is therefore essential to optimize the configuration of a mobile cyber-physical application for each individual capability set of a type of device. Adding this resource optimization consideration into the configuration problem makes it even harder to manage and develop multiple software versions. The optimization process must also ensure that any non-functional constraints on memory consumption or other resources are met by the cyber-physical application software's configuration.

Emerging solution → Mobile cyber-physical application software product-lines. Software product-lines (SPL) [6] are a promising approach for dealing with the complexity of managing a mobile cyber-physical application targeted for multiple mobile Internet device platforms. An SPL is a software platform designed with points of variability so it can be rapidly reconfigured for different requirement sets. A critical component of an SPL is a model of the points of variability and the rules governing their configuration.

A common approach to modeling SPL variability is called *feature modeling* [17]. A feature model uses a unit of abstraction, called a feature, that represents an increment of product functionality or point of variability. Feature models use a tree-like structure to specify the constraints on their configurations.

A configurable mobile sensor software platform can be created using SPL principles [36]. An SPL feature model provides a roadmap that explicitly captures the complex rules needed to reconfigure the software for multiple target OS, middleware, and hardware sets. This model helps prevent developers from making hard-to-diagnose configuration mistakes and decreases development time for new platforms [36].

SPL feature models can be transformed into mathematical representations, such as constraint satisfaction problems (CSP) [3] or satisfiability problems (SAT) [22]. After an SPL feature model is in one of these mathematical formats, optimized software configurations can be derived that minimize cost, power, or other critical properties [36]. This type of configuration optimization helps developers automatically generate precisely-crafted cyber-physical application software configurations for each target platform, which would otherwise be hard to discover manually.

New techniques for optimizing SPL configuration using constraint and SAT solvers can produce good results for deriving highly optimized software designs [3, 22] that can improve battery life and reduce cost for mobile cyber-physical application software. In some research endeavors, these SPL optimization techniques have been shown to produce good results for dynamic mobile software configuration at runtime. For example, WreckWatch can use SPL configurations to help rapidly setup highly optimized mobile cyber-physical software deployments.

In some situations, such as when resource constraints on memory or power are added, deriving SPL software configurations using CSP or SAT techniques can be time consuming. For example, attempting to derive configurations of the WreckWatch software that fit within the memory limits of less powerful mobile Internet devices is challenging. Applying emerging heuristic methods, such as Filtered Cartesian Flattening [35], to derive configurations can drastically reduce solving time. These types of heuristic techniques can be used to aid developers when the complexity of the cyber-physical application's resources or other non-functional

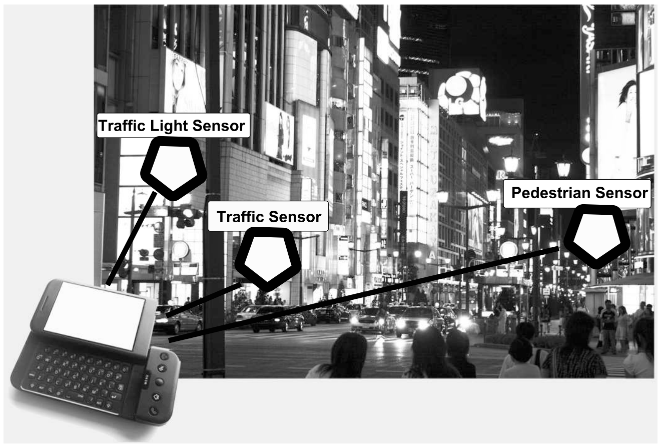

Fig. 7 WreckWatch Integration with Heterogeneous External Sensor Network

constraints cannot be tackled by existing CSP or SAT techniques.

3.4 **Challenge**: Integrating External Sensors

Context. With the emergence of pervasive and ubiquitous computing, everyday objects and activities will contain embedded sensing, computing, and communication capabilities. These smart devices will increasingly interact in networks to jointly perform computational tasks. Conventional device networks dedicated to a single application will have to open up, connect, and interoperate with each other to allow multiple and new applications to use their services. Devices may even be required to discover each other dynamically and interact in an *ad hoc* fashion. The integration of mobile Internet devices with conventional sensor networks is one instance of these network-of-networks scenarios.

For example, Figure 7 shows how WreckWatch can dynamically connect to a sensor network embedded in the road or road-side units to deliver detailed information on the road condition that led to the accident. It can further establish an *ad hoc* communication link to sensors attached to passengers to collect and integrate health data that would allow for remote assessment of the required medical aid.

Open problems. Most embedded devices have a custom-made software and hardware platform designed for a specific purpose. Mobile Internet devices are made to stay online while on the move. These devices are equipped with sufficient memory, processing, and communication units to check emails, browse the Web, and make phone calls.

Conventional sensor platforms are low-cost devices deployed in a high density to monitor environmental phenomena or to track objects. Compared to mobile Internet devices, conventional sensor platforms are far more restricted in terms of their storage and processing capabilities, communication range, and power supply. Moreover, the operating system for conventional sensor platforms differs considerably from mobile Internet devices since conventional sensor platforms can be recharged less often and controlling energy consumption is a major concern.

The different device and network capabilities result in incompatibility issues that make a seamless integration between cyber-physical applications and external sensor networks a significant research challenge. Incompatible communication links prevent today's mobile Internet devices from exchanging IP-based messages with conventional sensor platforms via a low-power radio connection. Incompatibility stemming from device heterogeneity is traditionally overcome by proprietary communication interfaces and gateway concepts.

Proprietary interfaces complicate the development of new applications, however, because they require in-depth knowledge of technical details [10]. Moreover, proprietary interfaces cannot be reused when new devices with different features are added. Application-level gateways have been introduced to compensate for the lack of a common language understood by all devices and also to translate between the different message formats. Moreover, communication via such an application-layer gateway introduces an additional level of indirection which bears extra configuration cost and hampers system evolution [34][26].

Emerging solution → A service-oriented device architecture (SODA) [7] based on mature Internet technology is a promising integration approach for heterogeneous sensor networks. Physical devices in a SODA can be modeled as services that hide device-specific implementation details behind well-defined, open or standardized interfaces. A service consumer may access and control a wide range of physical devices via their service interfaces without being affected by the diversity of the underlying device-specific hardware, firmware, and software.

There are two benefits of device-centric, service-oriented architectures when integrating external sensors. First, services abstract from technical details and provide ready-made building blocks that can be quickly combined to build new applications [27]. Second, when physical devices become available, corresponding services can be announced through which other devices can find out about their capabilities [2].

Web services realize a service-oriented architecture and have been successfully deployed as an integration media for business systems and distributed applications. They comprise a number of standards to define the description, registry, and communication of services. Web services are poorly suited for embedded devices, however, since they are too resource-intensive. The Device Profile for Web Services (DPWS) [9] helps address this drawback by defining a minimal set of implementation constraints to enable secure Web service messaging, discovery, description, and eventing on resource-constrained devices. For example, the DPWS restricts the size and complexity of messages, provides an asynchronous publish-subscribe mechanism, and allows for dynamic service discovery.

These features make DPWS an ideal candidate on which to base a solution for the seamless integration between mobile Internet devices and conventional sensor networks. For example, WreckWatch could dynamically send a message into a wireless sensor network at a traffic intersection to discover available sensor platforms and services. The mobile Internet device can then invoke all service that match its requirements and aggregate the returned data with local sensor readings. The use of XML as message exchange format and the transmission via the Internet Protocol allow for a communication independent from any device-specific low-level interfaces.

Additional R&D is needed to enhance DPWS since it does not fully address the constraints of conventional wireless sensor networks. In particular, DPWS uses UDP-/IP and TCP/IP for transmission, which is not natively supported in low-power IEEE 802.15.4 based radio networks. IP support in wireless sensor networks is a prerequisite for using DPWS and the 6LoWPAN working group explores encapsulation and compression mechanisms to receive and send IP packets over IEEE 802.15.4 based networks.

Since DPWS uses XML as message exchange format allowing for a standardized data exchange, its verbosity may require several radio packets to transmit a single message. Design choices are being explored to minimize the cost of providing structured data and functionality description, such as compression and tag compacting techniques [15] and HTTP-based service bindings [26]. In addition, Moritz et al. [25] propose adaptations and enhancements to limit the number of exchanged DPWS message for service discovery and meta data exchange. These types of models will improve the integration capabilities of heterogeneous sensor networks.

4 Emerging R&D Opportunities and Challenges

The R&D challenges and solutions addressed in Section 3 were based on our WreckWatch application described in Section 2. We are also creating other mobile cyber-physical applications and supporting Internet services that are in earlier stages of development. This section describes the R&D challenges that have emerged in our ongoing work on these applications, but are not yet as well formulated as the challenges and solutions presented in Section 3.

4.1 Augmented Reality

Augmented reality (AR) [13] is an emerging new area for mobile cyber-physical applications and supporting Internet services. The ability to combine virtual information with real world images was historically restricted to expensive instrumentation, such as heads up displays for flight avionics or luxury automobiles. Recent advances in the area of augmented reality allow the creation of portable versions of these interfaces using smartphones.

Mobile cyber-physical AR systems use GPS receivers, accelerometers, and compasses precisely capture the orientation and actions of smartphone users and deduce what the user is looking at. Virtual geotagged information is then obtained, typically from an Internet service, and overlaid across a smartphone's camera display. Overlaying information on the display allows the camera preview to serve as a looking glass that blends virtual and real world imagery.

We are developing an AR system for creating Augmented Reality Teaching Spaces (ARTS) in collaboration with educators in the Humanities. The goal of the ARTS project is to produce an AR platform that allows teachers to use an Internet service to publish geotagged information that students can see overlaid across real-world imagery in a smartphone camera display. Figure 4.1 shows how this platform will be used to fuse assignment information with real imagery from the Vanderbilt campus. For example, biology, anatomy, geology, or archeology instructors could mark up demonstrations with information that students can access in the laboratory or in the field. English classes could remediate literary works in virtual worlds that could be affected by real world actions.

New research challenges are already being uncovered in our development of AR projects. Interpreting what the user is looking at based on compass and GPS data requires precise estimations and fast fetching of large geotagged datasets. Many existing cyber-physical applications use custom hardware with high accuracy sensors. We have found that commodity smartphones have significant jitter in their sensor readings, requiring the use of complex data filtering.

Additional R&D is therefore needed to study strategies for handling the lower accuracy of commodity sensors. Fetching geotagged datasets from an Internet ser-

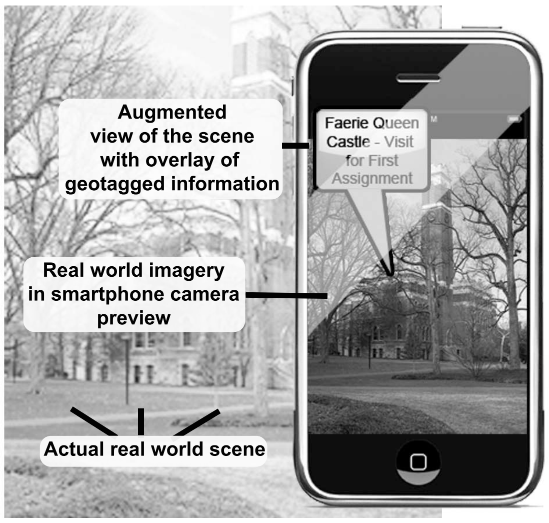
Augmented view of the scene with overlay of geotagged information
Faerie Queen Castle - Visit for First Assignment
Real world imagery in smartphone camera preview
Actual real world scene

Fig. 8 An Augmented Reality Teaching Platform

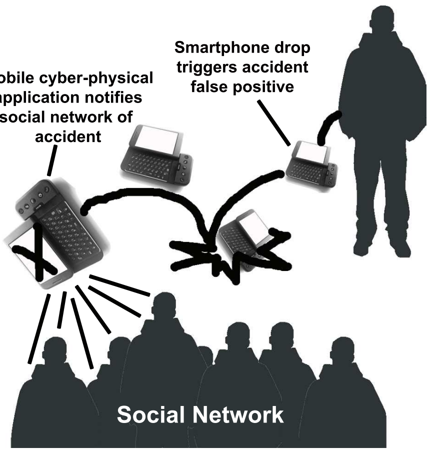
Smartphone drop triggers accident false positive
Mobile cyber-physical application notifies social network of accident
Social Network

Fig. 9 Accident False Positive Dissemination to Social Network

vice fast enough to provide real-time AR is challenging with varying cellular connectivity and bandwidth. We see the need to develop approaches for exposing more physical heading information, such as location and speed, to the supporting Internet services to more intelligently deduce what data to send to mobile cyber-physical applications.

4.2 Interaction with Social Networks.

Social networking platforms, such as Facebook and Twitter, provide Internet services that can be used by mobile cyber-physical applications to glean key social data about users. Recent mobile Internet device middleware platforms (such as the Palm Pre's Web OS) also offer libraries that simplify access to these social networking services. This type of social data can improve cyber-physical applications in various ways, such as WreckWatch's ability to notify friends and family when accidents occur.

Interacting with the social networks of users clearly offers significant possibilities for mobile cyber-physical applications. Care must be taken, however, to ensure that automated interactions with social networks do not harm user reputations or cause emotional damage to friends and family. For example, WreckWatch has the potential to detect accident false positives and send notifications to emergency contacts, as shown in Figure 4.2. WreckWatch is carefully designed to minimize these inccorrect accident reports, but cannot guarantee that mistakes will not be made.

Verification has been used to ensure physical safety properties, such as that a plane will not crash due to an unforeseen software state. Likewise, it is becoming important to investigate how verification can be used in the context of notifications to social networks. Although sending a notification of a non-existent accident from WreckWatch to user emergency contacts is not physically catastrophic, it is certainly problematic. Additional R&D is therefore needed to investigate techniques for verifying correct interactions with social networks when the messages that are being sent have significant potential for producing negative impacts.

4.3 Patient Diagnosis

Typical cyber-physical application for health care use expensive proprietary hardware that it is not feasible for a patient to take home. Mobile cyber-physical applications that can monitor patient health using onboard sensors or connected external sensors can be produced and delivered to patients much more affordably. Moreover, these mobile cyber-physical health systems can use standard IP networking to send data back to Internet services that aggregate information for doctors.

We are currently investigating the use of smartphone accelerometers and networked Bluetooth accelerometers to provide continual real-time monitoring of the symptoms of Parkinson's disease. As shown in Figure 4.3, the mobile cyber-physical application that we are developing will collect tremor characteristics from patients and then relay this information to an Internet service. Doctors will then use this service to see trends in symptoms over the course of a day and adjust medication dosages more precisely.

Collecting data from onboard smartphone sensors is relatively easy for a mobile cyber-physical application. Processing and disseminating data in real-time becomes much more challenging for our Parkinson's monitoring application or other applications that use mul-

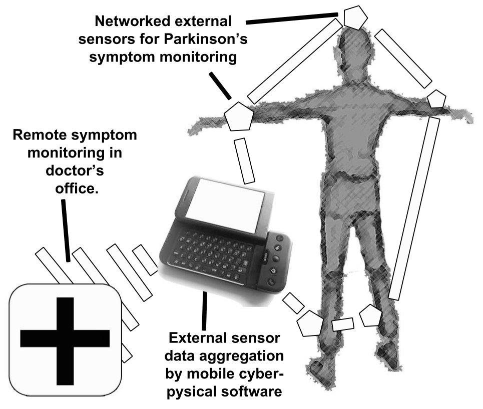
Networked external sensors for Parkinson's symptom monitoring
Remote symptom monitoring in doctor's office.
External sensor data aggregation by mobile cyber-pysical software

Fig. 10 Mobile Cyber-physical Application for Real-time Monitoring of Parkinson's Disease Symptoms

tiple external sensors networked through USB, Bluetooth, or other means. Developers of cyber-physical applications must therefore determine appropriate architectures that can buffer data when cellular connections are unavailable, yet not overrun device memory.

It is possible to perform some onboard processing on the phone to reduce the amount of data that must be transmitted from the phone to the Internet service or buffered, but these approaches require carefully balancing processing load, data accuracy, and timeliness of results. Additional R&D is therefore needed to develop the software patterns and architectures to manage large streams of external sensor data that must be processed on by a mobile cyber-physical application and then sent to a supporting Internet service.

5 Concluding Remarks

The benefits of using mobile Internet devices as the foundation for novel cyber-physical applications is growing as these devices continue to proliferate. Many types of cyber-physical applications are easier to implement atop mobile Internet devices compared with conventional large-scale deployments of customized hardware and software. Achieving this vision of building complex mobile cyber-physical applications that leverage supporting Internet services requires solutions to hard R&D challenges, including optimizing power consumption and devising software architectures that leverage the increased power of these devices.

Our work developing mobile cyber-physical applications in the context of WreckWatch and related projects yielded the following lessons:

1. **Many components of the solutions are highly related.** For example, MDE tools can better control and understand computing clouds, drive the configuration of an SPL, and help predict power consumption. Unified MDE approaches, such as the NAOMI platform [8], that tie many of these solution components may be required to analyze mobile cyber-physical application properties that span devices and services.
2. **Analysis of properties, such as safety, that span a combination of devices and services is difficult.** New MDE approaches for analyzing these systems of cyber-physical applications and internet services will be needed.
3. **Factoring social/human properties of systems into system analysis is not well understood.** New R&D is therefore needed to evaluate the ramifications of interacting with social networks.
4. **It is hard to integrate mobile Internet devices with conventional sensor networks.** Solving the incompatibility issues caused by device heterogeneity with a service-oriented device architecture is a promising direction to increase the integration capabilities of heterogeneous sensor platforms.
5. **Individual mobile devices are prone to unexpected unavailability.** Fluctuating environmental conditions, geographical areas of limited coverage, and battery exhaustion can cause mobile devices to become unavailable unexpectedly, which motivates additional R&D on handling intermittent failures.

The goal of this article was to present R&D challenges we found most pressing when developing and operating our mobile cyber-physical applications and supporting Internet services. Many other challenges must also be addressed when developing these applications and services. We look forward to working with the R&D community to identify and resolve these challenges.

References

1. K. Arisha, M. Youssef, and M. Younis. Energy-aware TDMA-based MAC for Sensor Networks. *IEEE IMPACCT*, pages 21–40, 2002.
2. D. Barisic, M. Krogmann, G. Stromberg, and P. Schramm. Making Embedded Software Development More Efficient with SOA. In *International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, pages 941–946. IEEE Computer Society, 2007.
3. D. Benavides, P. Trinidad, and A. Ruiz-Cortés. Automated Reasoning on Feature Models. In *LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, volume 3520, pages 491–503. Springer, 2005.
4. P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The Case for Power Management in Web Servers. *Power Aware Computing*, pages 261–289, 2002.
5. R. Buyya, C. Yeo, and S. Venugopal. Market-oriented Cloud Computing: Vision, Hype, and Reality for Delivering

IT Services as Computing Utilities. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08, IEEE CS Press, Los Alamitos, CA, USA)*, 2008.

6. P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, Boston, USA, 2002.
7. S. de Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker. SODA: Service-Oriented Device Architecture. *IEEE Pervasive Computing*, 5(3):94–96, 2006.
8. T. Denton, E. Jones, S. Srinivasan, K. Owens, and R. Buskens. NAOMI-An Experimental Platform for Multi-modeling. In *Proceedings of MODELS*, pages 143–157, Toulouse, France, October 2008.
9. DPWS. Devices Profile for Web Services (DPWS), 2006. http://schemas.xmlsoap.org/ws/2006/02/devprof/.
10. C.-L. Fok, G.-C. Roman, and C. Lu. Enhanced Coordination in Sensor Networks through Flexible Service Provisioning. In J. Field and V. T. Vasconcelos, editors, *Coordination Models and Languages (COORDINATION)*, volume 5521 of *Lecture Notes in Computer Science*, pages 66–85. Springer, 2009.
11. J. Froehlich, T. Dillahunt, P. Klasnja, J. Mankoff, S. Consolvo, B. Harrison, and J. Landay. UbiGreen: Investigating a Mobile Tool for Tracking and Supporting Green Transportation Habits. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1043–1052. ACM, 2009.
12. D. Gay, P. Levis, and D. Culler. Software Design Patterns for TinyOS. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):22, 2007.
13. A. Henrysson and M. Ollila. UMAR: Ubiquitous mobile augmented reality. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, page 45. ACM, 2004.
14. J. Hill, D. C. Schmidt, J. Slaby, and A. Porter. CiCUTS: Combining System Execution Modeling Tools with Continuous Integration Environments. In *Proceeedings of 15th Annual IEEE International Conference and Workshops on the Engineering of Computer Based Systems (ECBS)*, Belfast, Northern Ireland, March 2008.
15. N. Hoeller, C. Reinke, J. Neumann, S. Groppe, D. Boeckmann, and V. Linnemann. Efficient XML Usage Within Wireless Sensor Networks. In *International Conference on Wireless Internet (WICON)*, pages 1–10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.
16. W. Jones. Forecasting Traffic Flow. *IEEE Spectrum*, 38(1):90–91, 2001.
17. K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh. FORM: A Feature-Oriented Reuse Method with Domain-specific Reference Architectures. *Annals of Software Engineering*, 5(0):143–168, January 1998.
18. P. Leijdekkers and V. Gay. Personal Heart Monitoring and Rehabilitation System Using Smart Phones. In *Proceedings of the International Conference on Mobile Business*, page 29. Citeseer, 2006.
19. P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler. The Emergence of Networking Abstractions and Techniques in TinyOS. In *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, 2004.
20. P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A Self-regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, volume 246, 2004.
21. S. Lindsey and C. Raghavendra. PEGASIS: Power-efficient Gathering in Sensor Information Systems. In *IEEE Aerospace Conference Proceedings, 2002*, volume 3, 2002.
22. M. Mannion. Using First-order Logic for Product Line Model Validation. *Proceedings of the Second International Conference on Software Product Lines*, 2379:176–187, 2002.
23. J. Mitchell-Jackson. *Energy Needs in an Internet Economy: a Closer Look at Data Centers*. PhD thesis, Citeseer, 2001.
24. P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM New York, NY, USA, 2008.
25. G. Moritz, E. Zeeb, S. Prüter, F. Golatowski, D. Timmermann, and R. Stoll. Devices Profile for Web Services in Wireless Sensor Networks: Adaptations and Enhancements. In *International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2009.
26. N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao. Tiny Web Services: Design and Implementation of Interoperable and Evolvable Sensor Networks. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 253–266. ACM, 2008.
27. A. Rezgui and M. Eltoweissy. Service-oriented Sensor-Actuator Networks: Promises, Challenges, and the Road Ahead. *Computer Communications*, 30(13):2627–2648, 2007.
28. G. Rose. Mobile Phones as Traffic Probes: Practices, Prospects, and Issues. *Transport Reviews*, 26(3):275–291, 2006.
29. T. Saponas, J. Lester, J. Froehlich, J. Fogarty, and J. Landay. iLearn on the iPhone: Real-Time Human Activity Classification on Commodity Mobile Phones. *University of Washington CSE Tech Report UW-CSE-08-04-02*, 2008.
30. D. C. Schmidt. Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.
31. J. Sztipanovits. Composition of Cyber-Physical Systems. In *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*, pages 3–6, 2007.
32. C. Thompson, J. White, B. Dougherty, and D. Schmidt. Optimizing Mobile Application Performance with Model-Driven Engineering. In *Proceedings of the 7th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, 2009.
33. J. Varia. Cloud architectures. *White Paper of Amazon, jineshvaria. s3. amazonaws. com/public/ cloudarchitectures-varia. pdf*, 2008.
34. D. Villa, F. J. Villanueva, F. Moya, F. Rincón, J. Barba, and J. C. López. Web Services for Deeply Embedded Extra Low-Cost Devices. In *International Conference on Advances in Grid and Pervasive Computing*, pages 400–409. Springer, 2009.
35. J. White, , B. Dougherty, and D. C. Schmidt. Filtered Cartesian Flattening: An Approximation Technique for Optimally Selecting Features while Adhering to Resource Constraints. *Workshop on Analysis of Software Product-Lines at the International Conference on Software Product-lines*, October 2008.
36. W. Zhang and S. Jarzabek. Reuse Without Compromising Performance: Industrial Experience from RPG Software Product Line for Mobile Devices. *Lecture notes in computer science*, 3714:57, 2005.