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 · Siobhn Clarke · Christin Groba · Brian Dougherty · Chris Thompson · Douglas C. Schmidt

the date of receipt and acceptance should be inserted later

Abstract *New and planned mobile Internet devices, such as Apple's iPhone and Motorola's Droid, have powerful processors and a variety of sensors that 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 paper presents a sampling of key R&D challenges facing developers of mobile cyber-physical applications that integrate with Internet services and summarizes emerging solutions that 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 an arbitrary mobile software architecture. 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 Systems, 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 Google Android-based phones (such as the Motorola Droid), have become incredibly popular. For example, Apple has sold over 33.8 million iPhones. 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 12 phones in 26 different countries within a year.

The broad dissemination of these mobile Internet devices, their accelerated processing power, and pervasive cellular connections make them ideal platforms for building novel *mobile cyber-physical applications* that process and react to data from external stimuli and make decisions that also impact the physical world [31]. For example, cyber-physical applications can sense environmental stimuli from a device's ambient light sensors, accelerometers, GPS sensors, audio recorders, and imaging systems, *e.g.*, a device's accelerometers can detect traffic accidents and dispatch first-responders to aid victims [32, 15]. Cyber-physical applications can also use a device's WiFi and cellular networking connections to relay data back to Internet services for further processing. When cyber-physical applications are combined with Internet services they can leverage context information the user environment, as well as social network information derived from the user's contacts, Facebook account, and other social databases.

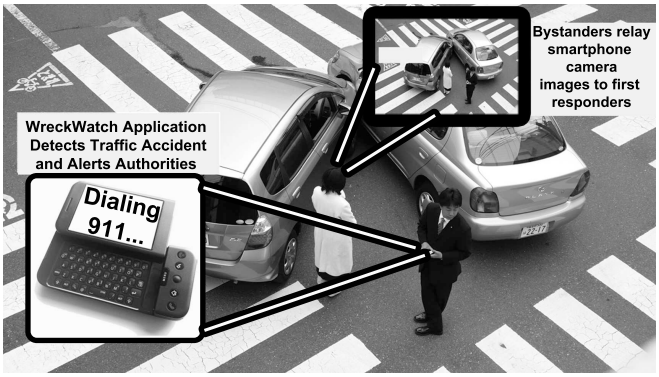

Fig. 1 Mobile Cyber-physical Application to Detect and Report Traffic Accidents

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 [28], track and analyze CO_2 emissions [10], detect traffic accidents and provide situational awareness services to first responders [15] (shown in Figure 1), measure traffic and derive road quality [27,23], and monitor cardiac patients [17].

Developing cyber-physical applications and Internet services for the new generation of mobile Internet devices is an important emerging area that has not yet received extensive coverage in the R&D community. The new devices are significantly more sophisticated than prior mobile phones and handheld computing devices, which enables developers to create more sophisticated applications. In particular, the current generation of mobile Internet devices contain more processing power and memory, as well as complex sensors and sensor integration platforms, that can be used to build cyber-physical applications.

Building cyber-physical applications atop mobile Internet devices offers a range of benefits compared to developing specialized hardware and software solutions with equivalent functionality. Maintenance of customized hardware and software solutions, such as wireless sensor networks, has historically been a key issue that must be addressed [19]. Not only must sensors be kept in working order, but they must also be supplied with 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 also been required in traditional custom hardware solutions to communicate data back to base stations for compute-intensive processing [20]. Mobile cyber-physical applications can communicate with Internet services using standard IP networking to transmit data for aggregation and receive processed results.

Another promising area for cyber-physical applications built on mobile Internet devices is their ability to travel with their owners and take measurements at multiple locations throughout the day. Conventional sensor network nodes are often stationary due of the tremendous power cost of movement. Moreover, monitoring human-centered phenomena, such as traffic congestion, is far easier when the sensors travel with mobile Internet device users.

Open R&D challenges. Despite the benefits of building mobile cyber-physical applications on top of mobile Internet devices and Internet services, there are a number of open R&D challenges limiting their development and deployment. This paper presents a sampling of 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 life-cycle. It is hard to predict how cyber-physical software architectures will consume battery power early in the development life-cycle, which makes it expensive and time-consuming to develop applications that can run for extended periods of time on a mobile Internet device.
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. Moreover, overprovisioning for occasional peak loads wastes resources for common usage conditions.
3. **Addressing platform variations**, which complicates cyber-physical application development. Developing a configurable cyber-physical software product for a wide range of targets is hard due to the variations between target platforms, the difficulty of optimizing the software for each platform, and the complexity of determining that non-functional constraints are met.
4. **Integrating external sensors** to exploit the benefits of co-existing conventional sensor solutions and emerging cyber-physical applications. The different resource constraints and device capabilities of mobile Internet devices and traditional sensor platforms complicate the seamless integration and collective evolution of heterogeneous sensor networks.

This paper summarizes efforts by ourselves and others to address these challenges.

Paper organization. The remainder of this paper is organized as follows: Section 2 summarizes a motivating example of a cyber-physical application and sup-

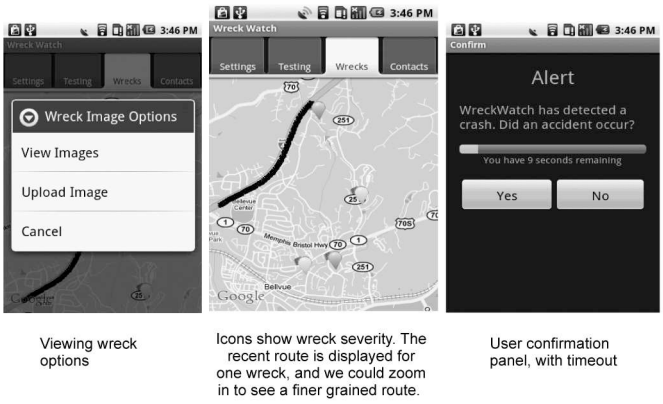

Fig. 2 The WreckWatch Traffic Accident Detection Application for Mobile Internet Devices

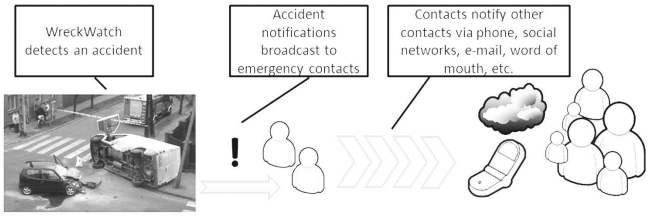

Fig. 3 The Wreckwatch Communication Paradigm

porting Internet services for detecting traffic accidents we developed; Section 3 explores a sampling of key R&D challenges and solutions based on our motivating example; Section 4 describes other emerging R&D opportunities in mobile cyber-physical systems and Internet services; and Section 5 presents concluding remarks.

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 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 paper since the range of challenges we faced to develop it are representative of gaps in current R&D efforts to develop mobile cyber-physical applications and supporting Internet services.

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 that can travel with drivers 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 be 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 proceeds with the accident report.

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) so it can automatically place emergency calls to 911 and dynamically provision an accident hot-line for friends and family of the accident victims. WreckWatch can also be configured to automatically send text messages to a list of emergency contacts with the emergency hotline 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 devices camera to take pictures of the accident and share them via the central server with first responders, as shown in figure 3. Video can also be captured and made available to first responders. WreckWatch's ability to use networks of bystanders to submit imagery of accidents helps improve 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 architec-

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

ture and Internet services that support it, *e.g.*, careful design is required to ensure it does not consume too much power, overconsume network bandwidth, or overwhelm central servers. This section describes a sampling of these key R&D challenges and presents promising solution approaches that we and others are developing to address these R&D gaps. We selected these challenges based on our experience developing WreckWatch and other mobile cyber-physical applications and supporting Internet services described in Section 4.

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

Context. Current mobile Internet devices have significantly increased processing capabilities, but this computational power can easily be used by cyber-physical software at the expense of increased power consumption. Whereas simple applications written for previous generation devices could easily consume power slow enough for devices to function for days between charges, current mobile cyber-physical applications use so many sensor that device batteries can be exhausted quickly, *e.g.*, the Apple iPhone specification lists the maximum battery life with continuous 3G data connection usage at 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 was able to completely drain the battery of an HTC G-1 developer phone in under two hours. Clearly, for a cyber-physical traffic accident detection application that is designed to be always on, this rate of power drain was a major defect.

On software platforms that support multi-tasking, such as Google Android and Palm Pre devices, cyber-physical software may be required to share power, computing, and sensor resources with multiple other applications. The cyber-physical software must not only draw power slowly enough for the device to remain charged for a full day but it must also do so while the user places phone calls, browses the web, and checks email. It is critical that cyber-physical software be designed so that it does not become such a significant power burden on a device that owners are unwilling to run it.

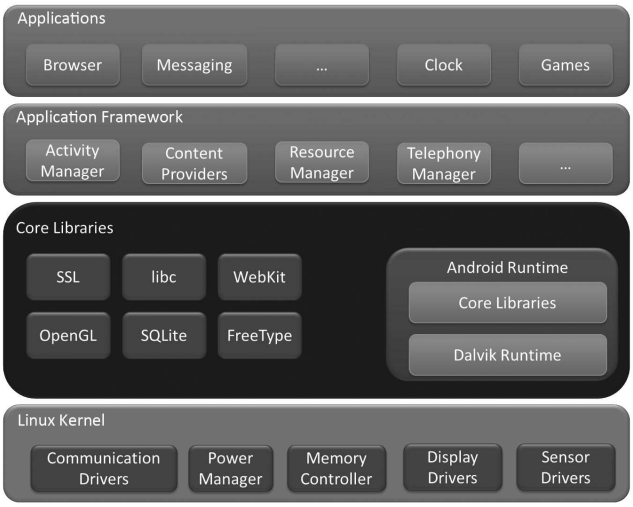

Fig. 4 Layering of Middlware and OS Abstractions

Open problems. A major challenge for developers of mobile cyber-physical applications is that it is hard 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. This inability to predict power consumption during the design stage was problematic since design changes late in the development process are more costly.

Many hard-to-predict platform factors play a role in determining how a particular software design consumes power. Middleware and OS decisions on task scheduling and memory utilization can affect a software architecture consumes power [11]. Networking implementation details, such as design decisions in the MAC layer of the OS [1], can also play an important role. Moreover, diversity in hardware (such as variation in sensors) can consumer power at different rates across devices. Using GPS on one device may be much more costly than on another.

Conventional cyber-physical systems with custom hardware and software typically use lightweight OS and middleware layers, such as TinyOS [18], 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 applicatons 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. Figure 4 shows Android's layers of middleware and net-

work stack abstractions and serves to show the difficulty associated with predicting 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 comiting to a specific architecture.

For example, the *System Power Optimization Tool* (SPOT) [32] is an MDE tool that models mobile software architectures and generates emulation code. SPOT utilizes a visual modeling environment, based on the Eclipse IDE, to allow developers to model the high-impact aspects of their designs before any implementation is performed. 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 overhead of implementing them by hand. It also allows developers to perform continuous integration testing [13] by substituting actual cyber-physical application logic for generated emulation code as the application logic is developed. Using this continuous integration process, developers can increase the overall accuracy of their models as development progresses.

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 addresses the problems associated with seeing through multiple layers of abstraction to predict power consumption. 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 the computational ablities of mobile Internet devices have improved significantly, many cyber-physical data processing tasks, such as image processing, are not suitable for a mobile application. Timely completion of tasks, such as large-scale data processing activities is not possible on mobile Internet devices due to their limited amount of memory and processing power compared to server infrastructure. For example, data aggregation of terabytes of information or image manipulation on thousands of multi-megapixel files, are beyond the capabilities of a mobile Internet device.

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 cluster [29]. Data harvested by cyber-physical applications from device sensors is sent to these Internet servcies that aggregate and process the data before sending the results back to the individual devices, *e.g.*, Microsoft's Bing Maps aggregates user-submitted photographs and combines them using its Photosynth technology to create 3D maps viewable on mobile phones.

For example, WreckWatch uses a centralized Internet services to perform 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 both 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 confront 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 significant excess capacity built into the computing platform, however, a significant amount of 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. Un-

planned for occurrences, such as professional sporting events or bad 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 developers to provision infrastructure that can provide the critical low latency response time needed by a cyber-physical system, 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 [22]. Power consumption and cooling are only expected to become more important and expensive for data centers in coming years [4].

Emerging solution → Cloud computing and resource auto-scaling. Cloud computing is a new paradigm that uses virtualization [5] to allow the dynamic provisioning of OS images in a data center. Developers have traditionally needed to purchase 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, however, are often inefficient platforms for Internet services that support mobile cyber-physical applications. For example, data processing loads to support mobile cyber-physical applications have periods of increased workload that are not always foreseen and fluctuate significantly. Additional OS images must be deployed in the cloud to handle these periods of increased activity. When the workload subsides, however, the additional OS images sit idle, wasting valuable resources, such as power, and leads to higher costs.

Recently, computing clouds such Amazon's Elastic Compute Cloud (EC2) have 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 allow 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 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.

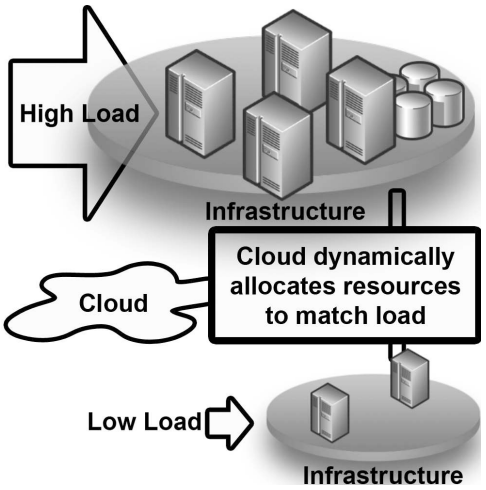

Fig. 5 Cloud Computing Can Dynamically Scale Resource Allocation to Meet Load

3.3 Challenge: Addressing Platform Variations

Context. Unlike the desktop and server operating system market, it is unlikely that one smartphone operating system vendor will dominate. 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 market with ≈30%. Developers will therefore need to develop 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, Android has released Android 1.5 and Android 2.0. Moreover, there are now five different Android devices by Motorola and HTC. Moreover, we have begun the process of determining how to reimplement WreckWatch on 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, on Android, WreckWatch can

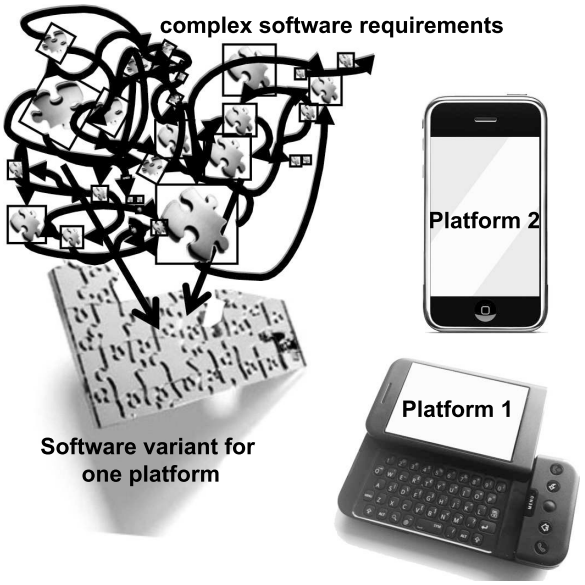

Fig. 6 Complexities of Targeting Multiple Platforms

run as a background service in parallel with other applications. In contrast, WreckWatch cannot run concurrently 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 the development complexity.

Even within a single OS platform there can be variations across versions and devices that add development complexity. For example, the latest 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.

Moreover, 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 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* [16]. A feature model uses a unit of abstraction, called a feature, which may represent an increment of product functionality or point of variability. A feature model uses a tree-like structure to specify the constraints on configuration.

A configurable mobile sensor software platform can be created using SPL principles [36]. The SPL's 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].

A key attribute of SPL feature models is that they can be transformed into mathematical representations, such as constraint satisfaction problems [3] or satisfiability problems (SAT) [21]. Once 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 allows developers to produce precisely crafted cyber-physical software configurations for each target platform that would 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, 21] that can improve battery life and reduce cost for mobile cyber-physical application software. Moreover, in some research endeavors, these SPL optimization techniques have been shown to produce good results for dynamic mobile software configuration at runtime, which could help to 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. Some current approaches for this use heuristic methods, such as Filtered Cartesian Flattening [35], to derive configurations and drastically reduce solving time. These types of heuristic techniques can be used to aid developers when the complexity of the cyber-physical system's resource or other non-functional constraints cannot be tackled by existing CSP or SAT techniques.

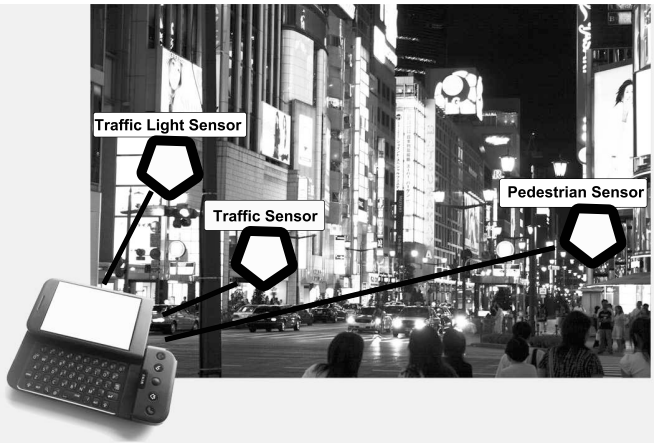

Fig. 7 WreckWatch Integration with Heterogeneous External Sensor Network

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 example for 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 that is 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 followed 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 [9]. 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][25].

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 a service that hides 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 sensor. First, services abstract from technical details and provide ready-made building blocks that can be quickly combined to build new applications [26]. 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) [8] 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 instance, a service on the mobile Internet device can dynamically send a message into a wireless sensor network to discover available sensor platforms and services they provide. 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 [14] and HTTP-based service bindings [25]. In addition, Moritz et al. [24] propose adaptations and enhancements to limit the number of exchanged DPWS message for service discovery and meta data exchange. Models such as these will improve the integration capabilities of heterogeneous sensor networks.

4 Emerging R&D Opportunities and Challenges

The R&D challenges and solutions 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.

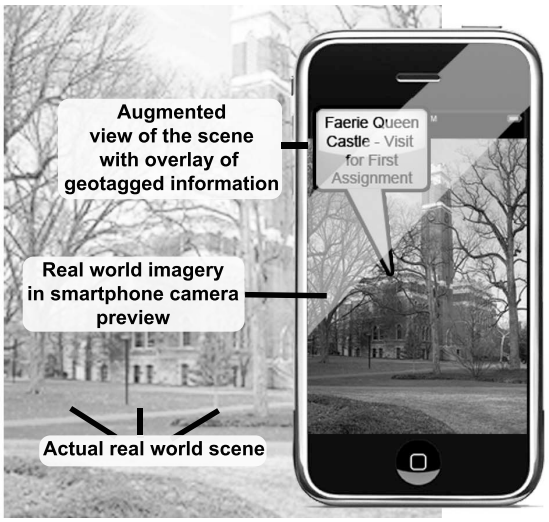

Fig. 8 An Augmented Reality Teaching Platform

4.1 Augmented Reality

Augmented reality (AR) is an emerging new area for mobile cyber-physical systems and supporting Internet services [12]. Previously, the ability to combine virtual information with real world images was 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 this 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 the field. English classes could remediate literary works in virtual worlds that could be affected by real world actions.

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

Additional R&D is therefore needed to investigate strategies for handling the lower accuracy of commodity sensors. Fetching geotagged datasets from an Internet service 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 the mobile cyber-physical system.

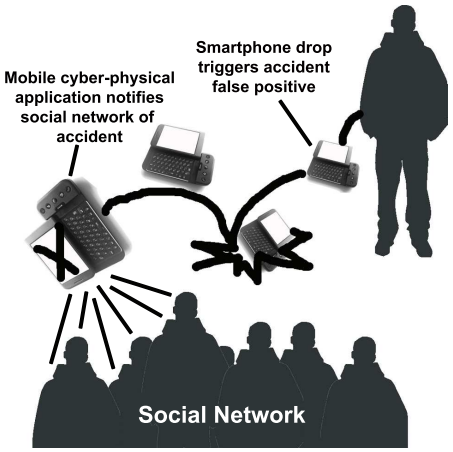

Fig. 9 Accident False Positive Dissemination to Social Network

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. Moreover, the latest mobile Internet device middleware platforms, such as the Palm Pre's Web OS, 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.

Clearly, interacting with a user's social network offers significant possibilities for mobile cyber-physical applications. At the same time, however, care must be taken to ensure that any automated interactions with the social network do not harm user reputations or cause emotional damage to friends and family. As shown in Figure 4.2, for example, in WreckWatch, there is the potential for accident false positives to be detected and notifications sent to emergency contacts. WreckWatch takes great care to try and minimize the chance that inccorrect accident reports are sent out 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 a user's emergency contacts is not physically catastrophic, it is certainly damaging. 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 a negative emotional impact.

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 systems 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.

In current work, we are 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 system 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. In the case of our Parkinson's monitoring application or other applications that use multiple external sensors networked through USB, Bluetooth, or other means, processing and disseminating data in real-time becomes much more challenging. Developers of cyber-

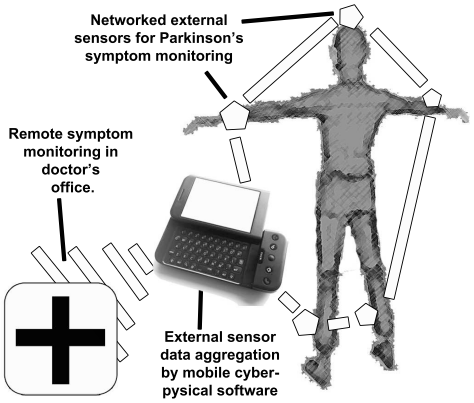

Fig. 10 Mobile Cyber-physical System for Real-time Monitoring of Parkinson's Disease Symptoms

physical application must 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

As mobile Internet devices continue to proliferate, the benefits of using them as the foundation for novel new cyber-physical applications is growing. Many types of cyber-physical applications are easier to implement on top of mobile Internet devices compared with conventional large deployments of customized hardware and software. Achieving this vision of building complex mobile cyber-physical applications that leverage supporting Internet services requires solutions to key 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. **Platform variations make it hard to run cyber-physical applications on a variety of devices.** By using SPL optimization techniques, however, cyber-physical software variations that provide better power consumption, cost, or other critical properties can be derived to overcome this challenge.
2. **Using cloud computing and auto-scaling to dynamically allocate computation resources to Internet services that support mobile cyber-physical systems is an efficient methodology for maintaining system performance.** Applying auto-scaling techniques also provides reductions in system cost and power consumption in comparison to statically provisioned hardware setups.
3. **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.
4. **Individual mobile devices are prone to unexpected unavailability.** Fluctuating environmental conditions, geographical areas of limited coverage, and a battery exhaustion can cause mobile devices to become unavailable unexpectedly, making additional R&D into handling failures important.

Mobile cyber-physical applications supported by Internet services for data aggregation and processing offer an exciting new paradigm for distributed computing. The computational potential of utilizing such devices will continue to increase as devices become cheaper, more widely available, and more powerful. Our goal in this paper was to present the R&D challenges we found most pressing throughout the development of our mobile cyber-physical applications. There are clearly other challenges that are faced when developing mobile cyber-physical applications and supporting Internet services that we could not cover in this paper. We look forward to working with others in 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. DPWS. Devices Profile for Web Services (DPWS), 2006. http://schemas.xmlsoap.org/ws/2006/02/devprof/.
9. 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.
10. 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.
11. D. Gay, P. Levis, and D. Culler. Software Design Patterns for TinyOS. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):22, 2007.
12. 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.
13. 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.
14. 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.
15. W. Jones. Forecasting Traffic Flow. *IEEE Spectrum*, 38(1):90–91, 2001.
16. 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.
17. 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.
18. 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.
19. 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.
20. S. Lindsey and C. Raghavendra. PEGASIS: Power-efficient Gathering in Sensor Information Systems. In *IEEE Aerospace Conference Proceedings, 2002*, volume 3, 2002.
21. 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.
22. J. Mitchell-Jackson. *Energy Needs in an Internet Economy: a Closer Look at Data Centers*. PhD thesis, Citeseer, 2001.
23. 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.
24. 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.
25. 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.
26. A. Rezgui and M. Eltoweissy. Service-oriented Sensor-Actuator Networks: Promises, Challenges, and the Road Ahead. *Computer Communications*, 30(13):2627–2648, 2007.
27. G. Rose. Mobile Phones as Traffic Probes: Practices, Prospects, and Issues. *Transport Reviews*, 26(3):275–291, 2006.
28. 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.
29. M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, pages 1–7. ACM New York, NY, USA, 1996.
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.