A Holistic Tool for Developing a Wireless Home-Based Health Monitoring System

Priyanka Bagade, Ayan Banerjee, Sunit Verma, Joseph Milazzo, and Sandeep K.S. Gupta

About the Authors

Priyanka Bagade is a PhD candidate in the Computer Science Department at Arizona State University. E-mail: pbagade@asu.edu

Ayan Banerjee, PhD, is postdoctoral fellow at the IMPACT (Intelligent Mobile and Pervasive Applications and Computing Technologies) Lab, Arizona State University. E-mail: abanerj3@asu.edu

Sunit Verma is a master's candidate in the Computer Science Department at Arizona State University. E-mail: sverma14@asu.edu

Early detection and diagnosis of potentially fatal physiological conditions such as heart attacks, require continuous monitoring of patient health following transfer from hospital to home. In response to this need, wireless home-based health monitoring systems (WHMS) are being proposed[1-5] as a low-cost solution. A WHMS (Figure 1) consists of physiological sensors that store, process, and communicate physiological data through a wireless communication network to a local manager (LM) such as a smartphone, which in turn uses cloud services for diagnosis.

Such WHMS should satisfy strict safety, security, reliability, and long-term real-time operation requirements, as mandated by regulatory agencies such as the U.S. Food and Drug Administration (FDA) and by policies such as the Health Insurance Portability and Accountability Act (HIPAA). Development of a WHMS that satisfies these requirements is a challenging task, and several tools have been proposed for this purpose.[9-16]

However, such tools are often lacking in several aspects of the WHMS life cycle, including design, development, and maintenance. The principal challenge in developing a holistic, wireless, home-based health monitoring tool comes from its potential diverse set of users. A WHMS life-cycle management tool should be usable by the following set of users each having their own needs and knowledge base:

a) Physicians have expert knowledge on diagnosis of a patient using WHMS. In this regard, they might need monitoring and display of physiological signals, usage of signal processing algorithms for diagnosis, and storage of medical records. However, we do not expect them to have knowledge of intricate engineering processes in a WHMS, such as architectural details of sensor hardware and programming languages.

b) Developers possess the required technical skills to implement WHMS hardware and software. They need to perform platform

Figure 1. Home-Based Health Monitoring System

specific coding, manage the computing resources in sensors, and ensure that the design meets the requirements. However, they might not have detailed knowledge about the diagnostic needs of a physician.

c) Medical device regulators set forth the safety, security, reliability, and operational standards of WHMS. They might need automated tools for verifying whether a prototype meets the standard requirements. But at present they rely only on documentation from manufacturers for the engineering details and from physicians for intended use.

d) A common user can be a patient or a health-concerned person who might wish to monitor his/her physiological health. Common users possess a high-level understanding of a WHMS, but they may lack both technical and medical requirements to implement WHMS.

To satisfy the different needs of the users, a WHMS life cycle management tool should have several key features, one of the most important being the capability to guarantee that the WHMS implementation satisfies software safety requirements set forth by regulatory agencies. Safety checks on implementations are currently done using separate tool chains such as CodeSonar,[24] which requires domain-specific knowledge from manufacturers and regulators, thus necessitating a learning period.

To quicken the process of safety verification, a WHMS tool can incorporate automatic safety checks of implementations. Hence, the capability to provide relevant levels of abstractions to the right set of users is necessary for a holistic WHMS tool. In this regard, a model-based approach, representing WHMSes using different levels of abstractions, is well suited for capturing user-specific requirements as well as for providing a foundation for in-depth analysis.

In this paper, we first identify the key features that should be supported by a holistic WHMS life cycle management tool. Based on these features, we classify existing tools and identify drawbacks. Finally, we analyze HealthDev,[17] an example of a model-based approach towards a holistic WHMS life cycle management tool that is in development to illustrate the key features discussed.

WHMS Life Cycle Management Tool Requirements

A WHMS life cycle management tool should consider its design, development, and maintenance in an automated manner as shown in Figure 2.

a) Design

A WHMS tool should enable the following design options for users:

Choice of Hardware:

Sensor: A large variety of wearable physiological sensors are available in the market. The WHMS tool should allow the user to choose sensors from this heterogeneous set, depending on their requirements. Most of the available WHMS development tools support only TinyOS-based sensors,[9-11,13] while a few also support C-based language.[12,15,17]

LM device: The LM device can be a smartphone or a personal computer. Tools generally focus on gathering and storing data on personal computers.[12-13] They have limited capability of developing interactive applications for the user. The WHMS tools will be usable if they support customizable design of interactive applications. Some tools such as Mobile Middleware support this functionality[13] (Table 1).

Cloud services: Computation power and memory capacity of LM devices, such as smartphones are limited. Thus, for heavy computation and a large amount of data storage cloud services are preferred. The tool should provide interfaces for developing apps that can interact with cloud services.

Network protocols: New network protocols such as Bluetooth low energy profile, are targeted towards embedded devices. The WHMS tool should allow the user to exploit all

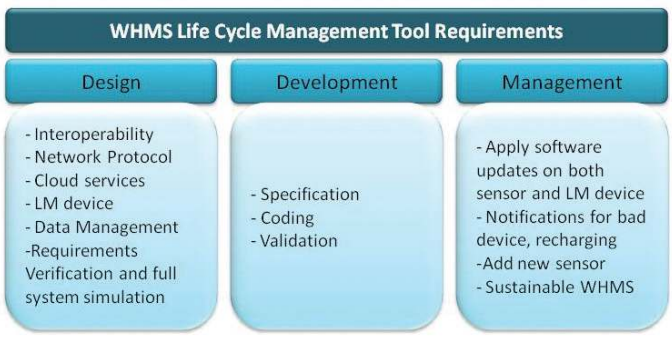

Figure 2. WHMS life cycle management tool requirements

Joseph Milazzo is a master's candidate in the Computer Science Department at Arizona State University. E-mail: jvmilazz@asu.edu

Sandeep K.S. Gupta, PhD, is professor of computer science and director of the IMPACT Lab, Arizona State University. E-mail: sandeep.guptaasu.edu

the functionalities of these new protocols and ensure interoperability among them. Most of the existing WHMS development tools support only the ZigBee network protocol,[9-13,15] and do not consider communication with LM devices,[9-11,14,16] which generally use Bluetooth or Wi-Fi (Table 1).

Interoperability: Often a WHMS needs a heterogeneous set of sensors to match the monitoring or diagnosis need of a patient. These sensors may have different data formats and communication protocols or customized Operating Systems (OS). Tools such as SPINE [2,12] and RapTex[15] support WHMS development with sensors having different OS. The framework by Mozumdar[9] et al. facilitates development for heterogeneous sensors supporting C-based programming dialects. A common drawback is that almost all the tools ignore LM design[9-11,14-16] and therefore do not support interoperability of the sensors with the LM.

Data management: Managing health data storage on either LM devices or cloud servers, in a privacy-ensured manner is important. Although current WHMS development tool supports data storage in sensors or LMs, they have to be integrated with secure storage services such as Microsoft Health Vault.[18]

Requirements verification and full system simulation: Software errors account for a considerable percentage of medical device failures, as documented in the U.S. Food and Drug Administration (FDA) MAUDE database.[7] WHMS design should have strict safety, security, and reliability requirements, as mandated by the FDA[7] and Health Insurance Portability and Accountability Act respectively.

A WHMS tool should not only provide capability of automated safety verification of the design (e.g., BAND-AiDe[11] analyzes a WHMS design with respect to safety requirements), but also allow users to choose from a wide variety of security protocols, such as elliptic curve cryptography[23] or physiological signal-based cryptography.[8]

Furthermore, the WHMS tool must support full system simulation before implementation. An important factor in WHMS is the cyber

Properties / Tools	Specification Input	Verification	Validation	Simulation	Code Generation	Supported Platform	LM Device	Users	Programming knowledge required	Physiological algorithm support	Supported Network protocol	Software Type	Health monitoring application development
Frame-work for model-ing , simulation and code generation[9]	State flow diagram (MATLAB)	Yes	No	Yes	Auto (Sensor)	TinyOS, MANTIS	No	Developer	No	No	ZigBee	Proprietary (MATLAB)	No
TOSDev[10]	GUI/ wiring diagram	No	No	No	Manual (Sensor)	TinyOS	No	Developer	Yes	No	ZigBee	Open source (wxWidget)	No
VipTOS[11]	GUI/ wiring diagram	No	No	Network Simulation	Auto (Sensor)	TinyOS	No	Developer	Yes	No (Network specific algorithms)	ZigBee	Open source (Ptolemy, TinyOS)	No (Designed for Wireless Sensor Network only)
SPINE 2[12]	No	No	No	No	Manual (APIs - sensor and base station)	C-based Dialect sensors	Personal computer (PC)	Developer	Yes	No	ZigBee	Open source (Java, TinyOS)	Yes (activity monitor)
Mobile middlware[13]	No	No	No	No	Manual (APIs - sensor and base station)	TinyOS Windows phone (C#)	Windows smart phone	Developer	Yes	No	ZigBee	Proprietary (Visual Basic for C#)	Only for windows phone & TinyOS based sensors
ANDES[14]	AADL	Yes	No	No	No	NA	NA	Developer	Yes	NA	NA	Open source (AADL)	No
RapTex[15]	GUI/ wiring diagram	Only for network	No	Network Simulation	Auto (Sensor)	C-based Dialect sensors	No	Developer	Yes	NA	ZigBee	Open source (Java, TinyOS)	NA
BAN-AiDe[16]	AADL	Yes	No	No	NA	NA	NA	Developer & Manufacturer	Yes	Yes	NA	Open source (AADL)	Yes
Health-Dev[17]	GUI / AADL	Yes	Yes	No	Auto (sensor and smart phone)	C-based dialect sensors	Android phone, PC	Patient, Developer & Manufacturer	No	Yes	ZigBee & Bluetooth	Open source (AADL, Java)	Yes

NA: Not Applicable
AADL : Architectural Analysis and Design Languages

Table 1. Comparison of Existing Tools

physical interactions of the sensors with the human body on thermal effects and drug diffusion dynamics. These interactions may induce serious safety violations[11] and must be analyzed before implementation. Although several tools, such as the one by Mozumdar et al.,[9] VipTOS[11] and RapTex[15] simulate the overall system before actual implementation, they ignore cyber-physical interactions.

b) Development

Figure 3 shows the different phases of WHMS development:

Specification: The design specification interface for a WHMS has to be usable by a varied set of users. Model-based specification is typically easy to use and can also be detailed for specific needs of the users. This approach is widely used by the researchers to develop health monitoring systems.[9-11, 15-17]

Implementation: Automated implementation is useful for rapid prototyping of WHMS and for incorporating requirements guaranteed in the implemented software. Available tools support the following:

1. Code generation using Application Programming Interface[12,13] (API)
2. Automatic code generation[9,11,15]

Automated code generation can abstract the engineering details from the user. This might be useful for physicians and common users. However, for a developer to optimize an implementation or for a regulator to check its correctness, access to the source code is necessary. To satisfy different needs, we hypothesize that the WHMS tool should support automated code generation and customization of the source code.

Validation: The code generated by the WHMS tool should be validated for correct operation using techniques such as unit testing and black box testing. For this, the WHMS tool should support automated test case generation.

c) Maintenance of WHMS

Because WHMS are meant to be pervasive, the maintenance operation should be minimal and seamless. In this regard, several approaches are applicable.

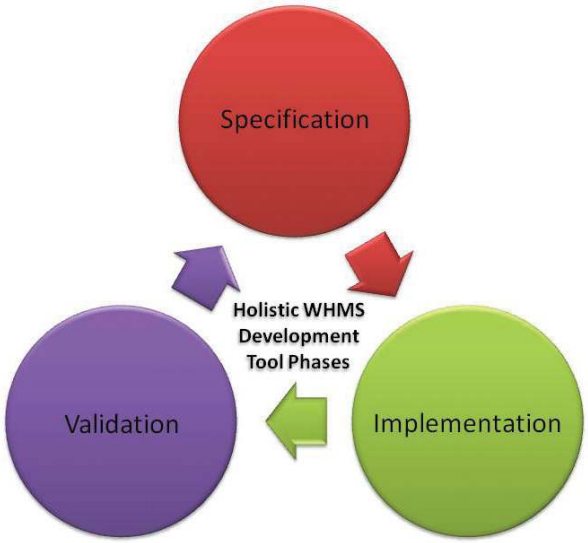

Figure 3. WHMS Development Tool Phases

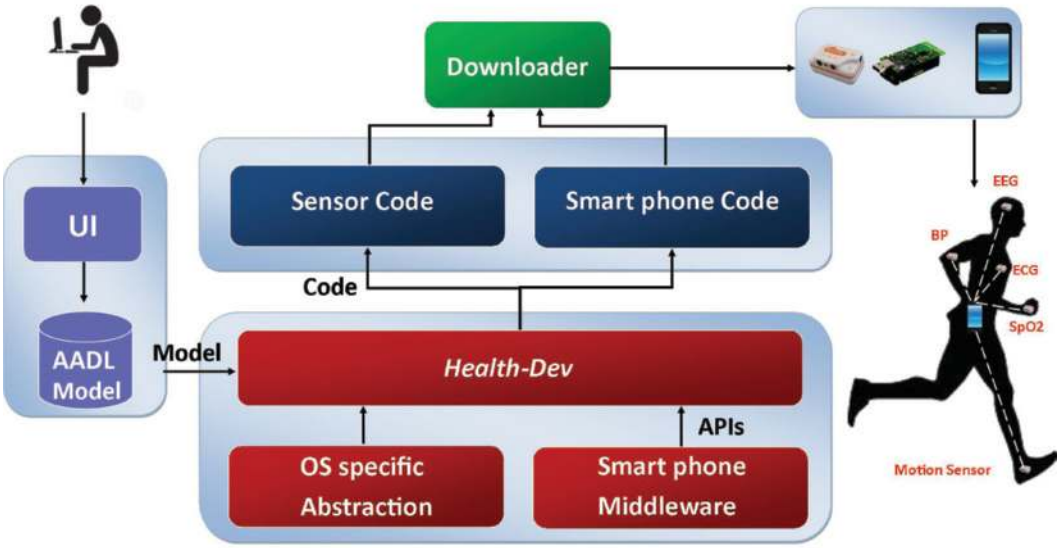

Figure 4. Health-Dev System Model

Software updates to LM and sensors: On-the-fly reprogramming of sensors is an active area of research and is currently achievable for very limited types of sensors.[19-20] In addition, a WHMS tool should support software updates to the LM without service interruption.

Notifications: When a sensor malfunctions, requires replacement or simply recharging, a user needs to be alerted through simple notifications.

Seamless addition of new sensors: Addition of new sensors into WHMS should be as simple as possible. The user should not have to re-configure the WHMS.

Sustainable WHMS: In order to minimize service interruptions due to lack of power, WHMS components should use energy management techniques and scavenge energy from the human body and the environment.

Table 1 shows a comparison of existing tools with respect to the lifecycle requirements of WHMS. It shows that no single tool supports the required phases of the WHMS life cycle. In the following section, we discuss the advantages and extensions of Health-Dev as a holistic WHMS tool.

Health-Dev—A Model-Based WHMS Life Cycle Management Tool

An example of such a tool, and still in development, is Health-Dev[17]—a model-based approach for WHMS design, development, and maintenance that fulfills the key requirements discussed in Section 2. As shown in Figure 4, Heath-Dev takes design requirements through an intuitive architectural analysis and design language (AADL)-based UI (Figure 5), and automatically generates downloadable codes for the sensors as well as for the LM device such as a smartphone or a personal computer. It also provides an interface with AADL for finer control or adding new functionalities. The tool facilitates model-based abstractions for physicians and common users through UI, developers, and device regulators through AADL.

Health-Dev supports code generation for a heterogeneous set of sensors having different

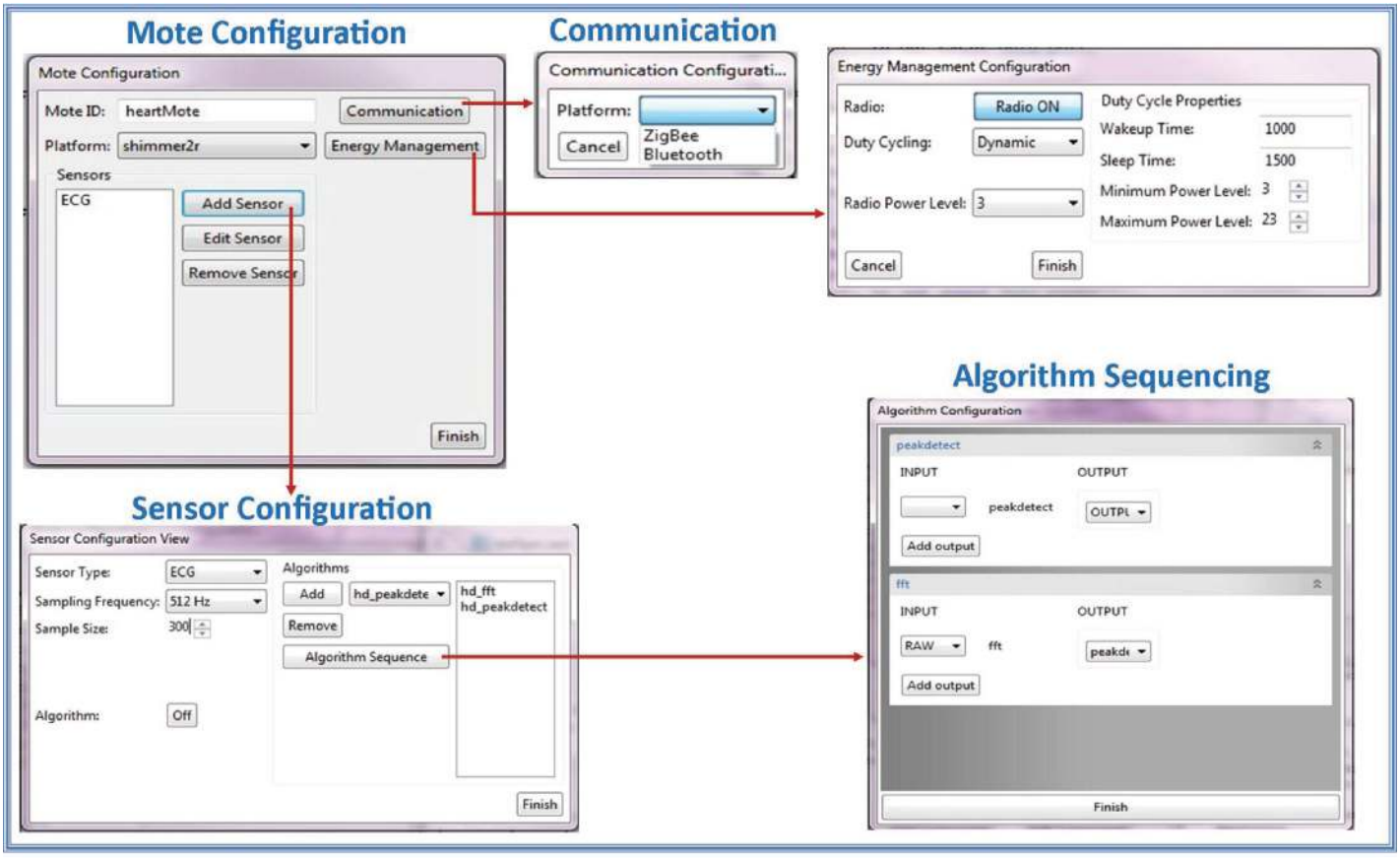

Figure 5. Health-Dev User Interface

OS (TinyOS or Arduino) and Android-based LM. It supports frequently used network protocols such as ZigBee and Bluetooth. Table 2 shows that Health-Dev uses Safe TinyOS[24] to verify the generated code does not undergo memory related errors such as an array of out-of-bound, pointer deference, or race conditions. Furthermore, the tool generates a more optimized code as compared to manual implementation that has been verified by TinyOS static code analyzer, cXprop.[24]

Any WHMS life-cycle tool should support physiological signal processing algorithms to be used on both the sensor and smartphone sides for developing smart context-aware applications. Using these physiological algorithms, the model-based tool can be easily extended to include security algorithms such as PSKA.[8] Sustainable sensor operation is implemented by incorporating radio power control algorithms that maintain reliable communication. Health-Dev also enables the development of interactive applications in smartphones.

Figure 6 shows an example of using the tool as a physiological data provider for an Android-based health app, PETPEEVES,[22] which monitors user's exercise habits and estimates the number of calories burned. As an input, it uses heart rate, which can be obtained from the LM software of the WHMS developed using Health-Dev for electrocardiogram (ECG) monitoring. In addition, the implementation generated by the tool is validated in Ayushman,[21] an experimental test bed developed at the IMPACT (Intelligent Mobile and Pervasive Applications and Computing Technologies) Lab at Arizona State University.

The WHMS life-cycle tool Health-Dev, presented here as an example, should be extended in the following aspects to get closer to a holistic WHMS tool: It should provide support for full system simulation before implementation; since it has AADL as specification framework, analytical tools such as BAND-AiDe[16] can be easily integrated for simulation support; and its functionality should be enhanced by adding cloud services to upload health data for physician's reference.

A verification module has to be integrated with the tool to ensure the correctness of the generated code. TUnit,[25] the unit testing framework, can be used to validate Health-Dev generated TinyOS-based sensor code. To support maintenance requirements of WHMS, over the air programming should also be added to support software patch updates in the existing system without rebuilding it. In order to enable sustainable WHMS, design support for controlling renewable energy sources is necessary in such a tool.

Conclusion

A WHMS tool should provide a developer, regulator, physician, and common user with the necessary functionalities for rapid prototyping of safe home-based health monitoring systems with required data processing capabilities. This paper identifies the key requirements for a holistic WHMS life cycle management tool that considers the design, development, and

Tasks	Safe TinyOS	cXprop
FFT*	+ 32.3%	- 8.2%
FFT+	+ 35.06%	- 9.36%
Peak Detection*	+ 32%	- 8.8%
Peak Detection+	+ 35%	- 9.78%
FIR*	+ 31.4%	- 8.16%
FIR+	+ 40.7%	- 10.5%
Differential Encoding*	+ 38.02%	- 9.1%
Differential Encoding+	+ 41.90%	- 11.1%
Out of range*	+ 34.2%	- 8.88%
Out of range+	+ 39.2%	- 9.97%
Statistics*	+ 31.2%	- 8%
Statistics+	+ 33.4%	- 8.29%

* Health-Dev genererated code
\+ BSNBench Code (manually written)

Table 2. Percentage Change in Code Size for Regular TinyOS (Unsafe)

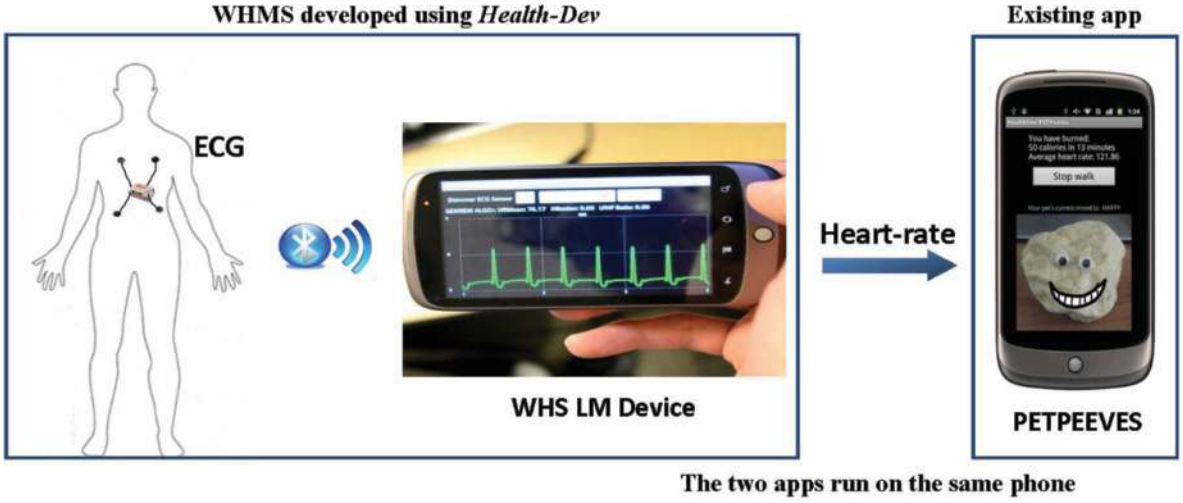

Figure 6. Health-Dev as a Physiological Data Provider to Existing App

maintenance of WHMS. It surveys currently available WHMS development tools and classifies them according to focus areas.

The classification reveals significant gaps in existing tools and requirements. To address these gaps, the paper discusses the advantages and extensions of Health-Dev as an example and potential solution for a holistic WHMS development tool that provides a usable model-based WHMS specification interface to the user and automatically generates software. It incorporates static analysis of the software to eliminate memory- related errors and race conditions, and ensures safe execution.

Acknowledgment

This research is funded by in part by NSF grant CNS-0831544 and IIS-1116385.

References

1. **Baker CR, Armijo K, Belka S et al.** Wireless Sensor Networks for Home Health Care. *Advanced Information Networking and Applications Workshops, 2007,* AINAW '07. 21st International Conference. 2007;2:832-837.

2. **Milenkovic A, Otto C, Jovanov E.** Wireless Sensor Networks for Personal Health Monitoring: Issues and an Implementation. *Comp Comm.* 2006;29(21):2521-2533.

3. **Stankovic JA, et al.** Wireless Sensor Networks for In-Home Healthcare: Potential and challenges. High Conf Med Device Software and Sys (HCMDSS) Workshop. 2005.

4. **Huo H, Xu Y, Yan H, Mubeen S, Zhang H.** An Elderly Health Care System Using Wireless Sensor Networks at Home. *Sensor Tech and Apps.* 2009;158-163.

5. **Ko J, Lu C, Srivastava MB, Stankovic JA, Terzis A, Welsh M.** Wireless Sensor Networks for Healthcare. *Proceedings of the IEEE.* 2010; 98(11):1947-1960.

6. **Peterson J.** Congress Passes Bill Allowing FDA to Regulate Smartphone Apps. Available at: http://news.yahoo.com/congress-passes-bill-allowing-fda-regulate-smartphone-apps-002106746.html. Accessed Feb. 2, 2013.

7. **FDA.** MAUDE Adverse Event Report. Available at: www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/detail.cfm?mdrfoi__id=2287727. Accessed Feb. 2, 2013.

8. **Venkatasubramanian KK, Banerjee A, Gupta SKS.** PSKA: Usable and Secure Key Agreement Scheme for Body Area Networks. *Information Technology in Biomedicine, IEEE Transactions,* vol.14, no.1, pp.60-68, Jan. 2010

9. **Mozumdar M, Gregoretti F, Lavagno L, Vanzago L, Olivieri S.** A Framework for Modeling, Simulation and Automatic Code Generation of Sensor Network Application. *IEEE Sensor, Mesh and Ad Hoc Communications and Networks.* SECON 2008:515–522.

10. **McCartney WP, Sridhar N.** Tosdev: A Rapid Development Environment for TinyOS. 4th Int'l Conf. on Embedded Networked Sensor Systems, *SenSys* 2006:387–388.

11. **Cheong E, Lee EA, Zhao Y,** Viptos: A Graphical Development and Simulation Environment for TinyOS-Based Wireless Sensor Networks. 3rd Int'l Conf Embedded Networked Sensor Systems, *SenSys* 2005:302–302.

12. **Fortino G, Guerrieri A, Bellifemine F, Giannantonio G,** SPINE2: Developing BSN Applications on Heterogeneous Sensor Nodes. Industrial Embedded Systems, SIES, IEEE International Symposium 2009: 128–131.

13. **Chen X, Waluyo A, Pek I, Yeoh W-S,** Mobile Middleware for Wireless Body Area Network. Engineering in Medicine and Biology Society (EMBC), Annual International Conference of the IEEE, 2010:5504–5507.

14. **Prasad V, Yang T, Jayachandran P, Li Z, Son SH et al.** ANDES: An Analysis-Based Design Tool for Wireless Sensor Networks. Real-Time Systems Symposium, RTSS, 28th IEEE International, 2007:203–213.

15. **Lim LB, Jang B, Yoon S, Sichitiu ML, Dean AG,** RaPTEX: Rapid Prototyping Tool for Embedded Communication Systems. ACM Trans. Sen. Netw. 2010:7.

16. **Banerjee A, Kandula S, Mukherjee T, Gupta SKS,** BANDAiDe: A Tool for Cyber-Physical Oriented Analysis and Design of Body Area Networks and Devices. ACM Trans on Embedded Comp Sys. In Press.

17. **Banerjee A, Verma S, Bagade P, Gupta SKS,** Health-Dev: Model Based Development Pervasive Health Monitoring Systems. Wearable and Implantable Body Sensor Networks (BSN), Ninth International Conference. 2012:85-90.

18. **Sunyaev A, Chornyi D, Mauro C, Kremar H.** Evaluation Framework for Personal Health Records: Microsoft HealthVault vs. Google Health. System Sciences (HICSS), 43rd Hawaii International Conference 2010;1-10.

19. **Hagedorn A, Starobinski D, Trachtenberg A.** Rateless Deluge: Over-the-Air Programming of Wireless Sensor Networks Using Random Linear Codes. Proceedings of the 7th international conference on Information processing in sensor networks (IPSN '08). 2008:457-466.

20. **Parthasarathy R, Peterson N, WenZhan S, Hurson A, Shirazi BA.** Over the Air Programming on Imote2-Based Sensor Networks. System Sciences (HICSS), 43rd Hawaii International Conference. 2010:1-9.

21. **Venkatasubramanian K, Deng G, Mukherjee T, Quintero J, Annamalai V, Gupta SKS.** Ayushman: A Wireless Sensor Network-Based Health Monitoring Infrastructure and Testbed. *Distributed Comp Sensor Sys.* 2005:3560.

22. **Verma S, Milazzo J, Xie Y, Bagade P, Banerjee A, Gupta SKS.** Model-based Wireless Health System Design Tool. Proc of 3rd Conf in Wireless Health. 2012.

23. **Seroussi G.** Elliptic Curve Cryptography. Inf Theory Netw Workshop. 1999;41.

24. **Cooprider N, Archer W, Eide E, Gay D, Regehr J,** Efficient Memory Safety for Tinyos. Proc of the 5th Int Conf on Embedded networked sensor systems, *SenSys.* 2007:205–218.

25. **Rincon Research Corporation.** Tinyos2.x Automated Unit Testing/Tunit. Available at: www.lavalampmotemasters.com/. Accessed Feb. 10, 2013.