Rapid Evidence-based Development of Mobile Medical IoT Apps[†]

Priyanka Bagade[*]
Intel Corporation, Portland, Oregon
Email: pbagade@asu.edu

Ayan Banerjee and Sandeep K.S. Gupta
IMPACT Lab, CIDSE, Arizona State University, Tempe, Arizona
Email: {abanerj3,sandeep.gupta}@asu.edu

***Abstract*—Mobile medical apps (MMAs) work in close loop with human physiology through sensing and control. As such it is essential for them to achieve intended functionality, without having harmful effects on human physiology, affecting the availability of the service and compromising the privacy of health data. However for a mobile app manufacturer, generating evidences regarding safety, sustainability and security (S3) of MMAs can be time consuming. To accelerate the development of S3 assured MMAs, we propose Health-Dev β tool that takes high level description of MMAs and automatically generates validated code and evidences of safety, security, and sustainability. Using the mobile artificial pancreas medical control application we show that Health-Dev β tool can generate code that satisfies requirements and reduce development time by a factor of 1.8.**

I. Introduction

The Internet of Things (IoT) is a network of systems connected to the Internet in which various embedded systems (wearable sensors/actuators) communicate with each other to exchange information. One of the example of IoT is Mobile Medical Apps (MMAs) used to provide pervasive health. According to industry surveys, by 2018, more than 1.7 billion smartphone and tablet users will have downloaded an MMA [1]. Such widespread adoption of smartphone based medical apps is opening new avenues for innovation, bringing MMAs to the forefront of low cost healthcare delivery. These apps often control physiology and work on sensitive data, thus it is necessary to have evidences of their correct functioning before actual deployment. The key challenges in ensuring correct working of MMAs are maintaining privacy of health data, long term operation of wearable sensors and ensuring no physical harm to the user [2]. Providing such evidences for safety, security and sustainability (S3) properties can increase development time and may require higher skills set for the developer. Hence, although ensuring S3 is essential, it often acts as a hindrance to innovation. In this paper, we propose Health-Dev β tool which takes high level models of MMA and rapidly generates S3 satisfied code as well as evidences. We also show examples where using Health-Dev β reduces development time by almost 1.8 times.

In an MMA (Figure 1), a wearable sensor can collect physiological data, and communicate it to the smartphone through the wireless channel. The smartphone in turn can communicate the data to a cloud server, where it can be stored in an electronic health record (EHR). The smartphone can also control safety critical devices such as drug infusion pump, thus forming a cyber-physical system. The control inputs computed by the MMA directly affects the human physiology. In such cases, the control algorithms have to be verified for patient safety. Due to lack of in depth knowledge about human physiology, smartphone app developers might fail to consider these safety concerns. Further, since the smartphone is not a dedicated healthcare platform, non-medical apps such as games, can affect patient safety by compromising the availability of the medical app (e.g. via rapid battery depletion). These apps often obtain physiological data from the wearable sensors which work on low power. Thus to have the sensor availability for long time, sustainable sensor design is needed. All communication between sensors, smartphone and cloud takes place through the wireless channel which is prone to security attacks. Thus the wirelessly communicated data should be encrypted. However, current app development and certification methods for MMAs do not adequately address these safety, sustainability and security issues.

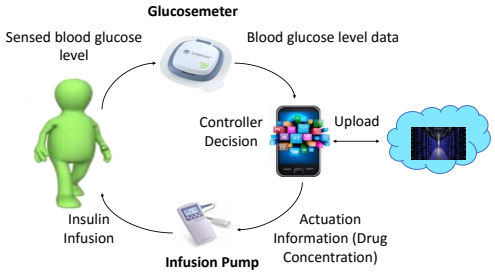

Fig. 1. System model for mobile medical applications. Artificial pancreas mobile application example.

Recently there has been efforts taken for data security evidence generation in MMAs with companies such as Happtique [3], however most of these approaches are subjective. Moreover, the proposed regulation system had failed to effectively detect serious security flaws in apps such as password stored in plain text. This gives rise to a need of objectively evaluating S3 requirements of MMAs before market approval.

In this paper, we hypothesize that a model based app development approach will enable us to objectively evaluate S3 requirement of MMAs and possibly generate trustworthy (S3 assured) code for sensitive operations such as actuator

[†]We are thankful to Yi Zhang and Paul Jones at FDA for their insights in regulatory challenges of MMAs. The work is partly funded by NSF IIS 1116385 and NIH NIBIB EB019202.

[*]This work was done when Priyanka was at ASU.

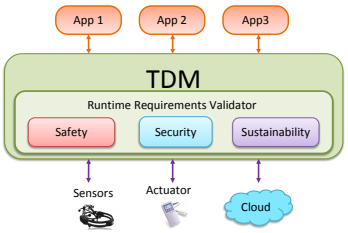

Fig. 2. Conceptual operating model for MMAs. The Trustworthy Data Manager monitors interfacing calls from different applications and verifies them against requirements.

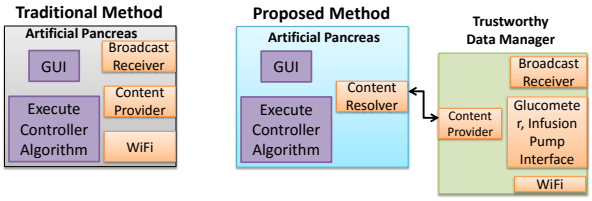

Fig. 3. Change in MMAs data sharing methodology. For access to sensing, actuation, and communication services the developer should issue inter-process calls to the TDM.

control, sensor interface and operation, and cloud communication. If the application design conforms to a standard model then automated tools such as hybrid automata based model checking techniques can be used for evaluating patient safety due to interaction between human physiology and app software as shown in our previous work [4]. Sensor design optimizer can be used to obtain sustainable design for long term availability. Moreover, security enabled software for interfacing the smartphone with sensors, actuators, and the cloud services can be generated automatically. To enable use of automated verification tools, we propose an App model, an operating model for the MMAs. It isolates the MMA sensor or actuator interface, data communication and data storage from the core mobile device graphical user interface (GUI) and processing algorithms. An app model should not limit the functionalities of the smartphone that can be exploited by the developers, nor it can cause a performance degradation of the app in terms of response time or memory and battery usage. The main contributions of the paper are as follows:

App Model: The proposed app model can enable safe and secure interactions of MMAs with sensors, cloud and other apps through a trustworthy entity. It can contain multiple apps working individually or sharing data with each other under accurate permissions.

Trustworthy Framework for development of MMAs: We propose Health-Dev β tool, which can be used by a developer to either generate evidences for trustworthiness of MMAs or critical parts of the code. In our manifestation, the safety verification of smartphone controller algorithm is done using formal modeling with spatio-temporal hybrid automata (STHA) [5], sustainable sensor design for a defined time, hardware and software constraints is obtained by using an optimizer algorithm, and the security enabled data communication over wireless channel is done by generating the interfacing components with automated code generator, Health-Dev β.

Finally, we evaluate improvements in development time with Health-dev β as opposed to manually coding using software development effort estimation tools, COCOMO [6].

II. Related Works

The existing frameworks for mobile health app development can be classified into three categories as: a) API support, b) programming abstractions, and c) automated code generation. The mobile app development with APIs is popular due its ease of programming [7]–[9], however, these do not verify S3 requirements. UPHIAC [10] and PRISM [11] frameworks provide health data security with APIs to interface with smartphone sensors and cloud for data storage. However these frameworks do not check for safety and sustainability requirements of health apps. Programming abstractions for sensors are being widely used to allow developers to write minimal sensor code to include wearables in health apps [12]–[16]. Considering the limited energy availability on wearables, energy efficiency techniques are supported by Reflex [14], LittleRock [15] and Turducken [16]. However these methods require sensor programming from the developer. Thus to reduce health apps development time and to get bug-free code, automated sensor code generators were proposed [17]–[20]. Still they require developers to write code for smartphone apps and its vulnerable components such as interfacing with sensors, actuators and cloud. This led to fully automated code generators for sensors and smartphone apps [21], [22]. However they only consider system safety and do not consider sensor sustainability and data security. This paper discusses an automated code generator for health apps which can ensure their trustworthiness while reducing developer burden.

III. App Model

We hypothesize that S3 assured MMAs should have an operating model, an app model as shown in Figure 2. In this new model of applications, the smartphone software itself is in charge of only the graphical and algorithmic aspects of the application. Every instance of data communication to the sensor, cloud, data storage in the smartphone, control inputs to the actuator, interaction with the user should be through a Trustworthy Data Manager (TDM). The app model enables the development of MMAs in the form of a suite, where participating apps are certified by the regulatory agencies against safety, sustainability and security, defined as follows:

Safety: Any form of control input from MMA is safety assured by checking it with hybrid automata based model.

Sustainability: Any sensor communicating with TDM supports long term availability with the sustainable design.

Security: Any form of data communication is privacy ensured by TDM. Also, data collected by the different applications are kept in secured databases (similar in principal to application sandboxing) shared with certified apps only.

Change in Development Paradigm: We consider a smartphone based wearable closed loop blood glucose level control

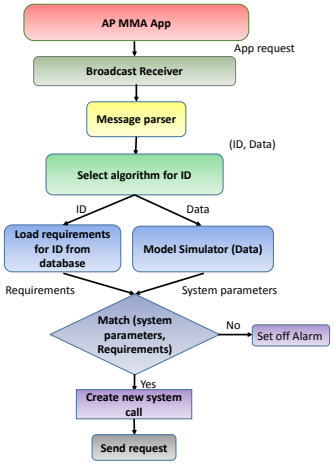

Fig. 4. Flowchart of the TDM runtime environment for validating the Apps operation against requirements.

application to explain the app model. The artificial pancreas application takes glucose levels from a continuous glucose monitor (CGM), computes the next infusion level according to a control algorithm, and sends actuation information to an infusion pump through the wireless channel.

Figure 3 shows change in app development method with the app model as compared to the traditional method. Using the traditional method, to develop Artificial pancreas app, developer needs to implement GUI, the control algorithm, CGM sensor code, security algorithms at both smartphone as well as sensor end. Considering the low power working and low level implementation language requirements, developing sensor code can be a challenging task. Developer is also responsible for writing methods to access and share data with other apps using content provider and resolver respectively, receiving broadcast messages, accessing WiFi and interfacing with cloud, CGM sensor and infusion pump.

On the other hand, to develop artificial pancreas (AP) app following the proposed app model, developer only develops the smartphone software to display and execute the control algorithm. The Health-Dev β tool takes care of developing sensor code and generating TDM app. TDM interfaces an app with sensor, other apps and cloud in S3 assured manner. Thus, the broadcast receiver, WiFi access, sensor interface required for data sharing and accessing are implemented at TDM end. Developer needs to implement only one content resolver through which it can access data from TDM. The AP app queries TDM to get data from CGM and sends data to the infusion pump instead of directly communicating with them.

IV. Trustworthy Data Manager

The TDM runtime process monitoring and validation functionality is not only cognizant of user preferences but also has enough embedded intelligence so that it can detect anomalous behavior of an app as shown in Figure 4. An app can register with a TDM by first establishing a Binder IPC channel with the TDM. Through the Binder channel it passes the UUID to TDM. The app can have its Multi-Tier Model encrypted using the UUID so that no other app except the TDM can access its models. TDM has a broadcast listener, which continuously listens for app requests. Once it receives a request from MMA, it will invoke a message parser which separates the requests into two parts: ID and Data. Corresponding to every ID there is an algorithm for runtime requirement checking. The algorithm will load requirements for the specific ID from the database. These requirements can be the safety regulations of mobile medical apps, the sensor sustainability requirements and the HIPAA security requirements. Simultaneously a model simulator will take the data as input and simulate a model for certain time to generate system parameter. Models may include Finite State Automata based representation of secure data management schemes, hybrid automata models of safety assessment of control algorithms and optimization algorithm to check sustainable sensor configuration. The TDM will be equipped with simulators that can execute the models and estimate the expected operating condition or state of the app for current phone context (CPC). These simulators can be reachability analysis on hybrid automata for MMA safety verification [5], sustainable sensor configuration using optimization algorithm [23]. If the output of the simulator matches the requirements, the TDM will create new system call to issue the initial app request. If the requirement is not matched, the TDM will set off an alarm.

The runtime validation functionality of the TDM can be expressed using functions $f_{safety}, f_{sustainability}$ and $f_{security}$ as shown in Equations 1, 2 and 3.

$$f_{safety} = \ < M_1(CPC, V_h), Th > \tag{1}$$

$$f_{sustainability} = \ < M_2(CPC, P), B_{min} > \tag{2}$$

$$f_{security} = \ < M_3(CPC, \tau_{M_3}), True\ or\ False > \tag{3}$$

Here, M_1, M_2 and M_3 are models used by TDM to validate the app request against safety, sustainability and security. V_h is the physiological parameter controlled by the app and Th is the safety threshold value. P is the power required for the sensor for specified operation in the system call whereas B_{min} is the minimum threshold value for the sensor battery level. Data access request τ_{M_3}, is checked against M_3 to see if the request is valid or not for a given CPC.

A more accurate estimation of the actual state of the app can be obtained by tracing the system calls made by the app. The app makes different types of system calls such as Broadcast Provider for radio transmission or Content Provider for accessing the sensors and actuators. All these system calls goes through the TDM app in our proposed method. and hence the TDM can trace the state of the app from these system calls. These two states can be compared in runtime. If there is a match then the app behaves as the proposed model and hence its system call is trustworthy and the TDM allows the requested operation. If the app operation does not match the model then a default operation is performed. This default operation has to be specified by the App and is kind of a fail safe mode of the application. If this fail safe mode is ever reached an exception will be raised by the TDM.

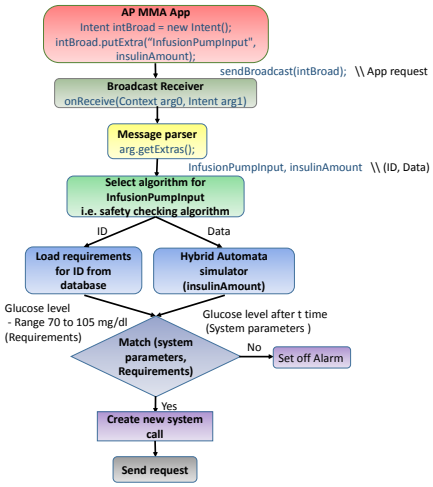

Fig. 5. Example flowchart for TDM runtime environment for artificial pancreas application.

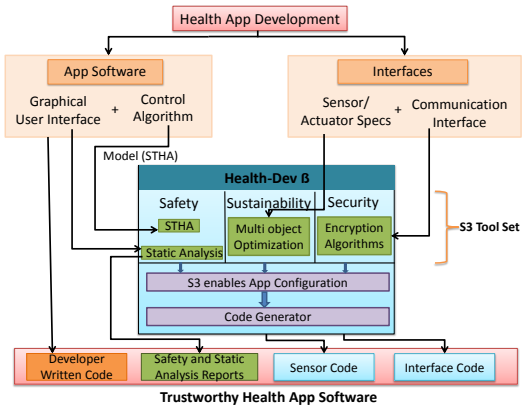

Fig. 6. Architecture for development of S3 assured MMAs. The developer writes the graphical UI and algorithms and also provides models for safety, security, and sustainability. For all interfaces it sends IPC calls to TDM. The TDM then uses the S3 assurance toolset to verify requirements.

Artificial Pancreas Example TDM: Figure 5 shows the MMA request processing by TDM for artificial pancreas app.Artificial Pancreas MMA sends the message of change in insulin concentration to TDM using Intent message. Broadcast listener implemented in TDM receives this message and invokes a message parser which separates the requests into two parts, ID and Data. For artificial pancreas, the ID is infusion pump input and data is amount of insulin to be infused. As for this example the request is for infusion pump to change drug concentration, the algorithm for checking if the requested insulin amount is safe or not is called. The algorithm loads requirements for the specific ID from the database i.e. glucose safe range. Simultaneously a model simulator takes the data as input and simulate a model for certain time to generate system parameter. For safety verification hybrid automata model for artificial pancreas is used. The simulator takes amount of insulin to be infused as input, simulated hybrid automata for that input and outputs blood sugar level after certain amount of time. Now this output matches the requirements i.e. the sugar level within required range, the TDM creates new system call to issue the initial app request. If the requirement is not matched, the TDM will set off an alarm.

V. Framework for S3 assured MMA Development

We propose a framework, *Health-Dev* β, to allow the developer to implement MMAs following the app model in Figure 2 and automatically ensuring safety from interactions, sustainability in sensor and security of data. It also generates TDM for interfacing with sensors, actuators, and cloud.

We envision that the *Health-Dev* β tool for trustworthy development of MMAs should have the architecture as shown in Figure 6. A developer can use such an architecture to either verify whether the developed MMA satisfies trustworthiness requirements or automatically generate critical parts of the code that can impose S3 vulnerabilities. The developer can provide a high level design of his app as input to the architecture. The high level specification considered in this paper is Architectural Analysis and Description Language (AADL) since it is industry standard and is generic enough to describe computational methods of smartphone apps as well as physiological aspects of human body [24], [25]. The high level design can be further optimized for sensor sustainability constraints using the optimizer algorithm. The algorithm gives designs, which are sustainable under certain time constraints [23]. For the controller app, control algorithm on smartphone end can be theoretically verified [5]. In this case, the developer should specify a hybrid automata representation of the application design in the Health-Dev β tool input. In our previous work, we have shown how AADL can be effectively used to represent hybrid automata [24]. The hybrid automata represented by the developer can then be extracted from the model and can be analyzed using the reachability analysis methods to determine patient safety.

The rest of the design can be used by the automatic code generator to generate sensor interface and communication code for the MMA. In our previous work, we have developed Health-Dev [26], which can convert an AADL design into sensor and smartphone implementations. In addition to using the standard software primitives, Health-Dev can be extended to have security plug-in, the TDM app, sensor design optimizer and app safety verifier which can be added on top of interfacing code to ensure S3 properties.

The STHA reachability analysis [5], sustainable optimized sensor design [23] and the security primitives generate a certification report stating their findings. The code and the certification report can then be reviewed by an expert personnel in the field to certify the application. If the MMA fails certification, then the developer can redesign and again use the same architecture.

A. Sustainable Design for Wearable Sensors

Wearable sensors typically scavenge energy from light or human body heat. The availability of scavenging sources is unpredictable. Thus, to have sustainable working of the sensors, we form an optimization problem to achieve energy neutrality. A system is energy neutral when the storage device

level remains same before and after the computation i.e. energy needed for the computation is solely obtained from a scavenging source.

In this problem formulation, we optimize the data sending frequency of the sensor depending on the availability of scavenging source. Let us consider that sending frequency is f_c when scavenging source is available and f_d when scavenging source is unavailable. The sensor battery will be charged only when the scavenging source is available. Thus, to achieve energy neutrality, we need to find values of f_c and f_d such that the execution should require only the energy obtained from scavenged source. Further, even the scavenging source is available, the sensor battery is unable to store energy beyond the battery capacity. Given these constraints, our aim is to minimize f_c and f_d. Equation 4 shows the simplified optimization formulation used to find optimal sensor design [23].

Find f_c and f_d that optimizes
(Power consumption should not go above scavenged power)
such that
Solar Energy: obtained when the source is available, (4)
Energy Neutrality: battery level = initial battery level,
Storage Constraint: battery level ≥ 0,
Battery Capacity Constraint: Stored Energy $\leq$ Battery Capacity.

The optimizer algorithm gives a set of time constrained sustainable designs. Developer can chose any one of the designs depending on availability and configuration settings.

B. Safety Verification in Smartphone Controller Design

The safety verification ensures that the side-effects of interactions between human physiology and sensors are within accepted limit. These interactions are typically governed by smartphone control algorithm when sensor acts as an actuator. In such a scenario, developer provides the hybrid automata which embeds the control algorithm while specifying high level design. Reachability analysis is then performed on the specified hybrid automata while considering optimized time constrained designs as initial set.

One of the safety verification methods is using spatio-temporal hybrid automata (STHA) [5] and related reachability analysis algorithm. STHA considers effect of interactions over time as well as over space to give more accurate results as compared to only time-based analysis. It considers the interactions of the form,

$$A_i \frac{\partial V}{\partial t} = B_i \frac{\partial^2 V}{\partial x^2} + C_i V + u_i, \quad (5)$$

This technique can be easily integrated with Health-Dev tool which can provide evidence for safety of the medical app.

C. Automated Implementation of Security-enabled MMAs

We have developed an automated model-based code generator, Health-Dev [26]. It takes requirements of smartphone and physiological sensors in the form of models in AADL and generates downloadable code for them. The automation in code generation helps to reduce manual implementation errors in developing wireless health monitoring apps. Health-Dev also provides graphical user interface to input requirements for the user who does not have any programming knowledge. It supports the use of physiological signal processing algorithms to process health data.

The current Health-Dev input model allows to specify various components such as sensing, computation and communication. In sensing, type of sensors, motes, sampling frequency, sending frequency can be specified. The computation component includes processing of data on both sensors and smartphone which uses various physiological algorithms from the database such as Fast Fourier Transform (FFT), peak detection, mean etc. Developer can add new algorithm as well as modify the inputs. Further, communication protocol between sensors and smartphone, as well as smartphone and cloud can be specified. It also includes specification of energy management techniques such as duty cycle, radio power level. We have extended the specification of communication component to include security protocol from a security algorithm database which can also be extended with new protocols.

TDM app generation: The code generator also generates a TDM app which ensures the secure wireless communication. It includes the security algorithm specified by the user in the input. The code generator maintains a code template for generating TDM app. On getting the specification of the security algorithm, it pulls the algorithm from database and inserts in the code. It contains encryption-decryption algorithms such as PEES [27], Advanced Encryption Algorithm (AES) to secure the physiological data over vulnerable wireless channel. In TDM code generation, code generator uses the Application Programming Interfaces (APIs) to allow communication between external wireless mote and an Android smartphone via Bluetooth. It consists of two components, Bluetooth API and sensor handler. Bluetooth API ensures connection establishment and data communication between mote and smartphone while sensor handler acts as a manager and registers all user assigned sensor, different algorithms associated with each sensor and handling of data received from the particular sensor. For each of the wireless communication calls, the generator appends the security protocol.

VI. Evaluation of Health-Dev β Tool

In this section, we evaluate Health-Dev to determine the overhead during runtime and the development effort reduction.

A. Overhead analysis of TDM runtime application

To check the overhead of runtime validation in TDM app, we have implemented the safety and sustainability models to get their execution time. Runtime security validation uses FSM which checks for the program flow with no complex computation involved thus it is assumed to no time. For safety analysis, pharmacokinetic model [28] for the artificial pancreas example discussed in Section III is implemented as Android service. It took 10.894 seconds to check for the given drug concentration value if the blood glucose level will remain within safety thresholds. For sustainability, the objective function in the optimization problem formulation (Equation 4) is implemented as Android service to check for the given sensor configuration change if the sensor will satisfy energy neutrality constraint. The execution time for the sustainability model was 0.03 seconds.

B. Effort Reduction in Developing MMAs

Generating S3 evidences for MMAs might increase the burden on development end. Thus to validate that Health-Dev β tool doesn't hinder the development process but accelerates it by automated code generation, we estimated and compared efforts required with manual implementation. For that, we used Constructive Cost Model (COCOMO) [6], a cost-effort estimation tool. As an example, we considered development of PetPeeves [29] app which requires developing user interface, ECG sensor code and communication interface between smartphone and sensor. Manual implementation of PetPeeves app required 2553 lines of code (LOC) without any S3 assured evidences. With Health-Dev β, the communication and sensor code is automatically generated with TDM app, leaving developer to write only 1678 LOC including sensor specifications. After executing the model, we got effort required with Health-Dev β tool as 6.9 developer-month, whereas for manual implementation it came out as 12.1 developer-month which is almost 1.8 times more. The developer cost is also increased by 1.8 times for manual implementation. This shows that the proposed tool not only saves the development efforts but guarantees correct working of the software by generating evidences of S3 requirements.

VII. Conclusions

In this paper, we focus on developing evidence-based medical apps for smartphones. The paper proposes an automated code generator, Health-Dev β for medical apps which generates evidences of safety, sustainability and security of the apps without hampering the development efforts. The COCOMO [6] effort estimation tool shows that the time required for developing MMAs with Health-Dev β is nearly 1.8 times less than manual implementation.

References

[1] FDA. Mobile medical applications. http://www.fda.gov/medicaldevices/productsandmedicalprocedures /connectedhealth/mobilemedicalapplications/default.htm.

[2] S. K. S. Gupta, T. Mukherjee, and K. K. Venkatasubramanian, *Body Area Networks: Safety, Security, and Sustainability*. New York, NY, USA: Cambridge University Press, 2013.

[3] P. L. Dolan. http://exclusive.multibriefs.com/content/health-app-certification-program-halted.

[4] P. Bagade, A. Banerjee, and S. Gupta, "Evidence-based development approach for safe, sustainable and secure mobile medical app," in *Wearable Electronics Sensors*, ser. Smart Sensors, Measurement and Instrumentation, S. C. Mukhopadhyay, Ed. Springer International Publishing, 2015, vol. 15, pp. 135–174.

[5] A. Banerjee and S. K. S. Gupta, "Spatio-temporal hybrid automata for safe cyber-physical systems: A medical case study," in *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on*, April 2013, pp. 71–80.

[6] "Cocomo model," http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html.

[7] X. Chen, A. Waluyo, I. Pek, and W.-S. Yeoh, "Mobile middleware for wireless body area network," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 31 2010-sept. 4 2010, pp. 5504 –5507.

[8] B. Kaufmann and L. Buechley, "Amarino: a toolkit for the rapid prototyping of mobile ubiquitous computing," in *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*. ACM, 2010, pp. 291–298.

[9] I. for Android, "Android development tools," https://www.sparkfun.com/products/retired/10748.

[10] T. Laakko, J. Leppänen, J. Lähteenmäki, A. Nummiaho *et al.*, "Mobile health and wellness application framework," *Methods Inf Med*, vol. 47, no. 3, pp. 217–222, 2008.

[11] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "Prism: platform for remote sensing using smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 63–76.

[12] W. Brunette, R. Sodt, R. Chaudhri, M. Goel, M. Falcone, J. Van Orden, and G. Borriello, "Open data kit sensors: a sensor integration framework for android at the application-level," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 351–364.

[13] F. X. Lin, A. Rahmati, and L. Zhong, "Dandelion: a framework for transparently programming phone-centered wireless body sensor applications for health," in *Wireless Health 2010*. ACM, 2010, pp. 74–83.

[14] F. X. Lin, Z. Wang, R. LiKamWa, and L. Zhong, "Reflex: using low-power processors in smartphones without knowing them," *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1, pp. 13–24, 2012.

[15] B. Priyantha, D. Lymberopoulos, and J. Liu, "Enabling energy efficient continuous sensing on mobile phones with littlerock," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010, pp. 420–421.

[16] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins, "Turducken: hierarchical power management for mobile devices," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 261–274.

[17] J. B. Lim, B. Jang, S. Yoon, M. L. Sichitiu, and A. G. Dean, "Raptex: Rapid prototyping tool for embedded communication systems," *ACM Trans. Sen. Netw.*, vol. 7, pp. 7:1–7:40, August 2010.

[18] E. Cheong, E. A. Lee, and Y. Zhao, "Viptos: a graphical development and simulation environment for tinyos-based wireless sensor networks," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ser. SenSys '05. New York, NY, USA: ACM, 2005, pp. 302–302.

[19] M. Mozumdar, F. Gregoretti, L. Lavagno, L. Vanzago, and S. Olivieri, "A framework for modeling, simulation and automatic code generation of sensor network application," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08. 5th Annual IEEE Communications Society Conference on*, june 2008, pp. 515 –522.

[20] B. Kim, L. T. Phan, O. Sokolsky, and I. Lee, "Platform-dependent code generation for embedded real-time software," in *Compilers, Architecture and Synthesis for Embedded Systems (CASES), International Conference on*. IEEE, 2013, pp. 1–10.

[21] M. Paschou, E. Sakkopoulos, and A. Tsakalidis, "easyhealthapps: e-health apps dynamic generation for smartphones & tablets," *Journal of medical systems*, vol. 37, no. 3, pp. 1–12, 2013.

[22] S. Procter and J. Hatcliff, "An architecturally-integrated, systems-based hazard analysis for medical applications," in *Formal Methods and Models for Codesign (MEMOCODE), 2014 Twelfth ACM/IEEE International Conference on*. IEEE, 2014, pp. 124–133.

[23] P. Bagade, A. Banerjee, and S. K. S. Gupta, "Optimal design for symbiotic wearable wireless sensors," in *Wearable and Implantable Body Sensor Networks (BSN), 2014 11th International Conference on*. IEEE, 2014, pp. 132–137.

[24] A. Banerjee and S. K. S. Gupta, "Your mobility can be injurious to your health: Analyzing pervasive health monitoring systems under dynamic context changes," in *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*. IEEE, 2012, pp. 39–47.

[25] ——, "Analysis of smart mobile applications for healthcare under dynamic context changes," *Mobile Computing, IEEE Transactions on*, vol. 14, no. 5, pp. 904–919, May 2015.

[26] A. Banerjee, S. Verma, P. Bagade, and S. K. S. Gupta, "Health-dev: Model based development pervasive health monitoring systems," in *Wearable and Implantable Body Sensor Networks (BSN), 2012 Nineth International Conference on*, may 2012, pp. 85 –90.

[27] A. Banerjee, S. K. S. Gupta, and K. K. Venkatasubramanian, "Pees: physiology-based end-to-end security for mhealth." in *Wireless Health*. Citeseer, 2013, p. 2.

[28] D. Wada and D. Ward, "The hybrid model: a new pharmacokinetic model for computer-controlled infusion pumps," *Biomedical Engineering, IEEE Transactions on*, vol. 41, no. 2, pp. 134 –142, feb. 1994.

[29] J. Milazzo, P. Bagade, A. Banerjee, and S. K. S. Gupta, "bhealthy: A physiological feedback-based mobile wellness application suite," in *Proceedings of the conference on Wireless Health*. ACM, 2013.