11-12-2015

The Challenges of High-Confidence Medical Device Software

Zhihao Jiang
University of Pennsylvania, zhihaoj@seas.upenn.edu

Houssam Abbas
University of Pennsylvania, habbas@seas.upenn.edu

Kuk Jin Jang
University of Pennsylvania, jangkj@seas.upenn.edu

Rahul Mangharam
University of Pennsylvania, rahulm@seas.upenn.edu

The Challenges of High-Confidence Medical Device Software

The Challenge of High-Confidence Medical Device Software

Zhihao Jiang, Houssam Abbas, Kuk Jin Jang and Rahul Mangharam

Department of Electrical and Systems Engineering, University of Pennsylvania, USA

{zhihaoj, habbas, jangkj, rahulm}@seas.upenn.edu

1. The Promise and Challenges of Medical Devices

Medical devices play an essential role in the care of patients around the world, and can have a life-saving effect. To cite one example, an estimated 3 million people worldwide have implanted pacemakers (a heart rate adjustment device), with ~600,000 added annually. In the United States, 800,000 people have an implanted defibrillator (another heart rhythm management device), with 10,000 added monthly. Clinical trials have presented evidence that patients implanted with defibrillators have a mortality rate reduced by up to 31%.

The medical device market is worth $289 billion with $110 billion within the US. Examples include everything from adhesive bandages to drug infusion pumps, surgical robots, deep brain stimulation systems and physiological closed-loop control systems like the artificial pancreas [1] which are in development. These are safety-critical technologies combining hardware and software, each of which must be rigorously verified to be efficacious and safe.

According to the US Food and Drug Administration, in 1996, 10% of all medical device recalls were caused by software-related issues. This percentage rose to an average of 15% of recalls from 2008 to 2012. Implanted cardiac pacemakers and defibrillators have approximately 80,000-100,000 lines of software code [2] which essentially makes all sensing, control and actuation decisions autonomously within the human body, over the 5-7 year device lifetime. The primary challenge of high-confidence medical device software is to guarantee the device will never drive the patient into an unsafe condition even though we do not have complete understanding of the controlled physiology.

Fig. 1. Pacemaker operating in a closed-loop with the heart. The leads sense cardiac electrophysiological activity from inside the heart tissue (AS/VS = Atrial/Ventricular Sense event) and actuate the heart (AP/VP = Atrial/Ventricular Pacing event to maintain a desired heart rate.

1.1 Human-in-the-loop Medical Devices

Medical devices can be classified as open-loop or closed-loop. A *closed-loop device* like a pacemaker is in a feedback loop with the organ(s) it affects (see Fig. 1): it monitors certain physiological variables like heart rate, and delivers therapy, in the form of low-energy electrical pulses, to maintain a healthy heart rate. Another example is the artificial pancreas, which monitors blood glucose levels and delivers therapy, in the form of insulin, to maintain safe glucose levels. An *open loop device* on the other hand, either, (a) like a drug infusion pump, does not measure any physiological variables: the therapy it delivers is pre-programmed and non-

reactive; or (b) as in the case of a blood pressure monitor, only measures physiological signals but does not deliver therapy.

Open-loop devices are operated by professionals to ensure the safety of the patient. Closed-loop devices require very little physician intervention after the discharge visit, and hence permit a better lifestyle. Because they are constantly monitoring the physiological variables, they permit a more timely delivery of therapy. The complex run-time diagnoses needed for closed loop performance, and the intricate therapy delivered, has driven most diagnosis and therapy functions into software. This software is life-critical and verification methods should provide a high confidence in its correctness.

Validating the safety and efficacy of closed-loop software, by definition, requires that the device be connected to the organ(s) it is affecting. For example, in the case of a pacemaker, that would be the heart of a living patient. But with the advent of computer models of physiological functions, such as those encompassed by the Physiome project or presented later in this article, the *model-based design (MBD) of closed-loop medical devices* presents efficient complementary approaches that are actively researched in various disciplines of engineering and computer science. In MBD, the device (or a model thereof) is connected to a *model* of the ``controlled physiology'' it interacts with. By high confidence verification we mean that under all possible behaviors of the physiological models (known or unknown), the device will satisfy its safety properties and not adversely affect the organ.

There are two major differences between modeling physiology and modeling man-made systems: first, physiology is much more complex and less well-understood than man-made systems like cars and airplanes, and spans several scales from the molecular to the entire human body. Secondly, the variability between humans is significantly larger than that between two cars coming off the assembly line. Using the cardiac pacemaker as an example of closed-loop device, and the heart as the organ to be modeled, we present several of the challenges and early results in model-based verification.

2. Life-critical Closed-loop Software

The heart is a specialized muscle that pumps oxygenated blood to the rest of the body. It is composed of four chambers: two upper chambers called the left and right atrium, and two lower chambers called the left and right ventricle, which contract synchronously. In a healthy resting adult, the heart rate is 60 to 100 beats per minute (each ventricular contraction is a beat). The contractions of the heart are controlled by the waves of spontaneous electric depolarization that traverse it regularly. A spontaneous electric current originates in the *Sino-Atrial (SA) node* in the right atrium and propagates throughout the atria, causing them to contract. It then propagates down to the ventricles along well-defined conduction pathways, causing the ventricles to contract in turn. The SA node is thus termed the natural pacemaker of the heart.

Under certain diseased conditions, the heart rate drops below what is needed to maintain adequate blood flow to the body. This clinical condition is called *bradycardia*. When such a heart rate drop is due to abnormalities in the electrical conduction system, an implanted pacemaker

might be recommended as treatment. A pacemaker is implanted near the left collar of the patient as shown in Fig. 1, and has two leads: one connects to the right atrium, the other to the right ventricle.

The leads act as both sensors and effectors: if the pacemaker fails to sense electric activity on either lead within certain time constraints, indicative of a delayed/missed contraction, it will send an electric pulse to the corresponding chamber to provoke contraction, thus acting as an artificial pacemaker. The algorithms for detecting missed beats are complex and implemented in software which runs within the pacemaker itself. Part of the difficulty of performing that detection comes from the great variability in heart rates between patients and even within a single patient across time. Moreover, because the pacemaker is limited to sensing electrical activity through its two leads, different phenomena can manifest themselves identically to the pacemaker, thus making detection even harder.

For example, in Endless Loop Tachycardia (ELT), this ambiguity causes the pacemaker to actually *induce* dangerously elevated heart rates (tachycardia), which would not have arisen had the heart been operating on its own. This is an example of an *adverse closed-loop condition*: a dangerous situation that arises as a result of the *interaction* between device and heart. No amount of open-loop device testing and verification can reveal this condition - hence the need for closed-loop validation of medical devices, and for physiological heart models that enable early and affordable closed-loop validation.

Fig. 2. Modelling different phenomena in the heart

3. Choosing the right model for the job

Heart models of different kinds have been developed for a range of applications and Fig. 2 shows four heart modeling approaches that emphasize the electric, mechanical, cellular and fluid flow mechanisms of cardiac function. Several of these modeling approaches employ over 4 million finite elements or 100,000 ordinary differential equations to describe the dynamics and take several hours to simulate a single cardiac cycle.

Cellular models describe the generation and spread of electrical action potentials (i.e., voltages) at the molecular-cellular level [3]. At the cellular level, the flow of charged ions into and out of the cardiac cell is responsible for the change in voltage across the cell membrane. Cellular models of electrical activity are used to study how activity across ion channels affect the relation between electrical and mechanical behaviors of heart tissue, as well as to study drug therapies that affect the ion channels properties.

Anatomical models are developed using imaging technologies like MRI, and seek to re-create detailed anatomical structures like fiber orientations and the distribution and extent of scar tissue. These structures affect the heart's operation by determining muscle contraction and modifying the speed and paths of electrical conduction throughout the heart. Thus anatomical models provide a foundation for whole heart modeling efforts that we cover next. They are also used to simulate the effects of certain medical devices like stents and artificial valves.

Whole heart models use a continuum approximation of the cellular models of electrical propagation with the structure obtained from anatomical models. Researchers have developed Partial Differential Equations models of electrical activity in the *whole heart* to analyze the mechanisms of various arrhythmias. Researchers at Johns Hopkins [4] further used these models to predict the onset of arrhythmias and propose potential therapies. Such *electro-mechanical models* help evaluate the mechanical effects of different arrhythmias on blood flow.

Electrophysiological (EP) heart models help study the timing properties of the generation and propagation of electrical signals through the electrical conduction system, and can accurately diagnose most arrhythmias. Unlike the aforementioned heart models which are built from the cellular level on up, EP models are developed by conducting a clinical EP testing procedure which is a type or timing analysis of the heart rhythm across the myocardium. EP models are amenable to *model checking* [5], a powerful verification technique pioneered in the semiconductor industry. EP testing is a common method to diagnose arrhythmias: a physician inserts catheters with electrodes into the patient's heart through the veins and measures the local electrical activity around the electrodes. The physician uses the *patterns* of electrical activity and its timing characteristics to diagnose the heart's condition in terms of arrhythmias, which are derangements to normal timing patterns. In particular, electrical timing parameters of action potentials in a tissue region like conduction delay, rest period and refractory period are measured, and any abnormal conduction paths are detected.

Fig. 3. Timing of electrical behaviors of the heart are modeled by a network of (1) node and (2) path automata. Each circle is a node automaton which models the generation and blocking of electrical events. Each line is a path automata which models conduction delays between nodes.

In [6], researchers from the University of Pennsylvania developed a heart model based on clinical EP testing. Since the pacemaker only looks at the *timing* of events as input from only two locations of the heart, this EP model only seeks to model the correct timing of electrical activity in select tissue of the heart. Specialized tissue like the SA node generates electric events spontaneously and is modeled by Timed Automata known as *node automaton* as shown at Marker 1 in Fig. 3. The node automaton models the timing of signal generation, blocking and transmission in heart structures like the AV node. The rest of the tissue is abstracted as variable conduction delays as *path automata* between node automata (Marker 2 in Fig. 3). Different heart conditions can be modeled by a network of node/path automata with different topologies and parameters (Fig. 3). Moreover, pacing applied to the heart can be represented as an external activation signal to the node automata. EP models have been validated by physicians, and have been used for model checking pacemaker software.

Finally, data-driven models such as [7] fit (fractional) differential equations directly to measured heart rate without modeling the underlying mechanisms, and are used for optimal control.

4. Closed-loop model checking of device software

Short of clinical trials (which we cover later in this article), current validation practice for closed-loop medical devices focuses on open-loop testing and reviews of the design process. In such open-loop testing, a set of input sequences is fed to the device, and the device's output is

checked for correctness, typically by comparing it to a pre-defined expected output. Such testing does not evaluate the effect of the device on the organ: e.g., we can't test how the heart rate changes following a pacing by the pacemaker. Thus, we need a heart model that can interact with the device. If we use, say, a high-fidelity PDE-based model such as the electro-mechanical models described earlier to react to the device and generate input sequences, there are an infinite number of heart rhythms that such a model can generate, and testing only uses a finite subset of those. Thus, testing (whether open-loop or closed-loop with a heart model) is necessarily an incomplete technique, and safety violations of the device may be missed during testing.

The timed automata-based EP models are amenable to *model checking*, a technique that mathematically explores *all* possible executions of the heart model and device software combination against specified requirements (e.g. the pacemaker will not pace the heart beyond an upper rate limit). Model checking is widely used in the semiconductor industry to verify chip designs at various levels of abstraction, in particular at the Register Transfer Level (RTL). Violations of the requirements are returned by the model checking tool as an execution trace, which can be analyzed and used to improve the system. To capture the variability in the heart's behavior (more generally, in the physiological phenomena of interest), the heart model is *non-deterministic*: for example, rather than specifying that the conduction delay in the AV node is always 0.14 second, we allow it to be any value in the correct physiological range [0.12, 0.2] second. The model checker will *symbolically* explore all executions corresponding to all values in this range (rather than select a few) in search for requirements violations. Subtle errors in the design of safety-critical systems that often elude conventional simulation and testing techniques can be (and have been) found in this way. Because it has been proven cost-effective and integrates well with conventional design methods, model checking has been adopted as a standard procedure for the quality assurance of automotive and avionics systems, but has yet to enter the world of medical devices.

Fig. 4. Endless-loop Tachycardia (ELT) with backward conduction (red arrows) and a healthy heart condition mapped to the same input-output execution of the pacemaker (middle sequence). The heart model should have the details to resolve this ambiguity.

Endless Loop Tachycardia (ELT) is one example of a safety hazard that arises in the interaction between pacemaker and heart, shown in Fig. 4. The ELT starts with an early ventricular contraction (PVC), which is a common scenario even in a healthy person. The electrical signal travels from the ventricle to the atrium (red arrows in Fig. 4), triggering an atrial sense (AS), i.e. the pacemaker senses an event in the right atrium. As a result, the pacemaker paces the ventricle (VP) after a pre-programed delay (AVI), which triggers ventricle to atrium conduction again and this VP → AS → VP positive feedback loop persists. The ventricular rate during ELT is determined by the conduction delay from the ventricle to the atrium and the programed delay in the pacemaker, which is very fast. The healthy condition in Fig. 4 demonstrates the same input-output sequence as ELT. In it, the pacemaker paces the ventricle after each atrial event (AS) generated by the SA node (as opposed to an AS conducted from the ventricle as in ELT). This is correct pacing at an appropriate (and relatively much slower) rate than ELT, and consequently maintains adequate blood flow. Failing to distinguish these two conditions in the

heart model may introduce false-positive when checking whether ELT exists, i.e. the healthy case returned as evidence for ELT.

Fig. 5. Multi-scale modeling of the heart. The heart model at a higher level (further to the right) contains all possible inputs to the pacemaker from the heart models at previous levels.

5. Physiological Model Abstraction and Refinement

From the example above, we can see that in order to perform model checking on the closed-loop system, the heart model should not only cover all possible inputs to the pacemaker specified in the requirements, but also have enough details to resolve ambiguities of executions that may introduce false-positives and/or false-negatives. The left column of Fig. 5 shows a collection of heart models each modeling a particular heart condition. However, these models do not cover *all* possible heart conditions, thus model checking the pacemaker with each of these heart models will not guarantee absolute safety. Physiological abstraction rules (R1-R7 in Fig. 5) are defined to increase the behaviors of the original model(s), while guaranteeing new behaviors introduced can still be physiologically valid. As an example, abstraction rule R4 merges parameter ranges for heart models with the same node and path topologies. Imagine two node automata *N1, N2* can self-activate within [0.3, 0.4] second and [0.5, 0.6] second, respectively. By applying R4, the new node automaton *N3* can self-activate within [0.3, 0.6] second. *N3* covers all behaviors of *N1, N2*, plus new behaviors which are mostly physiologically valid. If the physiologically-invalid behaviors introduced into *N3* are returned by the model checker as evidence, they can be eliminated by refining *N3* back to *N1, N2*.

By systematically applying the abstraction rules on the initial set of heart models, we get an abstraction tree (Fig. 5). The root of the abstraction tree is a heart model H_{all} with only two node automata corresponding to the inputs to the pacemaker. By allowing both node automata to be able to send inputs to the pacemaker $[0, \infty]$ second after the last input, the heart model H_{all} covers all possible inputs to the pacemaker. However, H_{all} cannot distinguish the ELT condition from the healthy condition due to the lack of representation of ventricle to atrium conduction. The heart model H_{cond} (Fig. 5) models the electrical conduction between the atria and the ventricles with a path automaton, thus is the appropriate heart model to evaluate ELT. Similarly, more complex properties will lead to appropriately detailed models along the Model Abstraction Tree.

Fig. 6. Model translation framework. The pacemaker design is verified using model checking and automatically translated into code implementation. Heart models are available at all levels.

6. From Verified Models to Verified Code

During model checking, the abstract model of the pacemaker is verified against safety requirements. The abstract pacemaker model is then automatically synthesized into simulation models and into a code implementation (Fig. 6) using the UPP2SF model translation tool we developed. Each automaton in a network of timed automata is mapped to a parallel state (called parent state) in Stateflow and each location in the automaton is mapped to an exclusive state within the parent state. Along with mapping all the edges in the UPPAAL model to the Stateflow

model, the behaviors of the Stateflow model is a subset of the corresponding UPPAAL model, thus properties verified in the UPPAAL model still hold in the Stateflow model. This automatic synthesis provides rigorous traceability throughout the development process and ensures that the verified model is translated into verified code using the Stateflow model of the pacemaker with the Simulink embedded coder. Similarly the heart model is translated from timed automata to Simulink and also synthesized into a Heart-on-a-Chip for platform level testing.

Fig. 7. Model-based Clinical Trials. The synthetic cohort is generated by randomizing a parametrized model from an appropriate distribution whose bounds are inferred from clinical data. The parameters are sampled within their physiological ranges, thus generating a large synthetic cohort. Each such model instance is connected to the device and simulated.

7. Model-based Clinical Trials

The final step before the introduction of a new high-risk medical device to market is the *clinical trial*. The *randomized controlled trial* (RCT) is the ``gold standard'' for guaranteeing that a medical intervention is safe and efficacious [8]. It is in general a major effort involving patients, medical investigators, biostatisticians, ethics boards, regulators and companies, costing several millions of dollars and running for 4-6 years on average. Yet technical errors can arise at almost every step of the trial planning, jeopardizing the validity of the results. Even if the trial is well-planned, poor execution, unexpected events or even just pure chance can lead to the wrong conclusions. The applications of computer models to the medical domain presented above have largely centered on the design and verification of a given device, and have mostly eschewed matters related to the clinical trial. There is however now an opportunity to use these computer models to assist in the planning and conduct of RCTs, as will be presented in this section. (Another application of modeling to trials is the UVA/PADOVA diabetes model, which replaces *animal* trials for evaluating diabetes control algorithms [1].

Suppose that a manufacturer of medical devices is designing a new implantable defibrillator for the treatment of certain abnormal cardiac rhythms, or *arrhythmias*. Both the hardware and software are tested by the company to ensure they satisfy their specifications. The device may then be implanted and tested on animals. But up to this point, the effect of the device on humans has not been observed. The RCT compares the efficacy and safety of the device on two groups of patients: the *treatment group* which is implanted with the new investigational device, and the *control group* which is on standard medical care, e.g. one or more devices already on the market [8]. Both treatment and control groups are monitored for a pre-determined amount of time, at the end of which the rate of treated arrhythmias is evaluated in each group. The results are analyzed to determine whether the difference in rates between the groups, if any, is *significant*, i.e. is unlikely to be due to chance alone.

We can design a *Model-Based Clinical Trial* (MBCT) to test a number of assumptions made by the trial investigators, before the trial starts (Fig. 7). In an MBCT, we start by modeling the physiological phenomenon of interest, in this case the spread of electrical activity in the human heart. The model should be valid of course (i.e., not produce too many non-physiological signals). For an MBCT the model must also be *rich*: it should be capable of simulating a large variety of arrhythmias that are targeted by the new defibrillator. Once such a parametrized model is

created, we can generate a large number of model instances by sampling the parameter space from appropriate distributions, which may be inferred from previous trials' data. This constitutes our synthetic cohort.

We can now analyze the effect of the device in ways not possible or impractical with a clinical trial, so as to guide the RCT investigators when designing the trial's protocol. For example, we may vary the distribution of arrhythmias in our synthetic cohort and analyze how this affects the device's performance. This is equivalent to running multiple trials on different populations in which the arrhythmias appear in different proportions. The investigators can then use these results to confirm or revise their confidence in the superiority of the new device to standard medical care across populations. We can also study the sensitivity of the trial's outcome to device settings: by running the trial multiple times with the same synthetic cohort but with different device settings every time, we get solid estimates for how different settings affect the MBCT's outcome. This in turn can inform the investigators whether they need to correct for different settings when drawing the trial protocol and when analyzing the results. Further, by breaking down the MBCT results by arrhythmia (or other interesting criteria) the investigators can make informed decisions, before trial start, on which classes of arrhythmias are most or least susceptible to treatment by the device. This in turn helps refine the eligibility criteria and focus the trial's efforts on certain classes of patients.

This and other experiments possible in an MBCT increase the probability of success of an RCT by providing early, fast and rigorous testing of various assumptions and hypotheses made by the trial investigators.

8. Conclusion

This article has surveyed the challenges of bringing new medical devices and their software to market from early verification to late-stage clinical trials, and gives an outlook to how modeling and formal methods can play a role in facilitating this process. *Complexity, limited observability and variability* stand out as three major challenges. The *complexity* of the physiological phenomena that the device is meant to control stems partially from the multi-scale nature of human physiology, where an organ's operation is affected by both molecular factors and patient's lifestyle. Moreover, the devices have limited *observability* on these complex physiological phenomena because increased observability usually requires increased invasiveness of the surgical procedures, with all the attending risks. Coupled together, complexity and limited observability imply that the detection and therapy algorithms of devices must deal, safely and reliably, with a lot of uncertainty. By explicitly allowing for non-determinism in the systems they study, formal methods are well-suited for dealing with uncertainty. The major challenge for formal methods is the development of appropriate physiological models and abstraction-and-refinement frameworks for addressing the complexity of the phenomena, and abstraction trees take a step in this direction in the domain of electrophysiology. *Variability* arises as the third major challenge: how to verify that a device works well in a population of patients that varies greatly in its characteristics and medical history? Clinical trials remain the standard and legally accepted way to answer that question. While models cannot substitute for observations in a human patient, they promise to alleviate the burden of conducting trials by early and rigorous testing of their assumptions. These

applications usher a new era of exciting research challenges at the intersection of computer science, statistics and medicine.

References

[1] M. Ghorbani and P. Bogdan. A cyber-physical system approach to artificial pancreas design. In 2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pages 1-10, 2013.

[2] Paul L. Jones. Senior Systems/Software Engineer, Office of Science and Engineering Laboratories, US FDA. Personal communication, 2010.

[3] A.J. Pullan, L.K. Cheng, and M.L. Buist. Mathematically Modelling the Electrical Activity of the Heart: From Cell to Body Surface and Back Again. World Scientific, 2005.

[4] Natalia A. Trayanova and Patrick M. Boyle. Advances in modeling ventricular arrhythmias: from mechanisms to the clinic. Wiley Interdisciplinary Reviews: Systems Biology and Medicine, 6(2):209-224, 2014.

[5] E.M. Clarke, O. Grumberg, and D. Peled. Model Checking. MIT Press, 1999.

[6] Zhihao Jiang, Miroslav Pajic, and Rahul Mangharam. Cyber-Physical Modeling of Implantable Cardiac Medical Devices. Proceedings of the IEEE,100(1):122-137, Jan. 2012.

[7] Paul Bogdan, Siddharth Jain, Kartikeya Goyal, and Radu Marculescu. Implantable pacemakers control and optimization via fractional calculus approaches: A cyber-physical systems perspective. In Proceedings of the 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems, ICCPS '12, pages 23-32, 2012.

[8] L. M. Friedman and C. D. Furberg and D. L. DeMets. Fundamentals of Clinical Trials. Springer, 2010.

Abstract

Keywords

D.2.4 Software/Program Verification < D.2 Software Engineering < D Software/Software Engineering, B.1.m.a Emerging technologies < B.1.m Miscellaneous < B.1 Control Structures and Microprogramming < B Hardware

Author Bio:

Zhihao Jiang (Student Member ‘10) received his Bachelor’s degree from UESTC China in 2008 and Master in Robotics degree from University of Pennsylvania in 2010. He is currently a doctoral student in the Department of Computer & Information Science at University of Pennsylvania. His research interests are in model-based design and certification of Cyber-Physical System, in particular closed-loop medical devices.

Houssam Abbas (M'15) received the B.S.Eng degree from the American University of Beirut, Beirut, Lebanon in 2004 and the M.S. and Ph.D. degrees from Arizona State University, Tempe, AZ in 2006 and 2015 respectively. He was a Computer-Aided Design (CAD) engineer with Intel's System-On-a-Chip (SoC) Verification group in Chandler, AZ from 2006 to 2014, working on functional verification and the verification of low-power designs. He is a postdoctoral fellow in the Department of Electrical and Systems Engineering at the University of Pennsylvania. His research interests are in the verification, control and conformance testing of Cyber-Physical Systems, in particular hybrid systems. Current research focuses on the verification of medical devices, verification and control of autonomous vehicles, and anytime computation and control.

Rahul Mangharam (M'02) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2000, 2002, and 2008 respectively. He is an Associate Professor in the Dept. of Electrical & Systems Engineering and Dept. of Computer & Information Science at the University of Pennsylvania. He is the Director of the Real-Time and Embedded Systems Lab. His current interests are in real-time scheduling and control algorithms for networked embedded systems with applications in autonomous systems, medical devices, energy-efficient buildings and wireless control networks. Dr. Mangharam has received numerous awards, including the National Science Foundation (NSF) CAREER Award in 2014, the 2013 IEEE Benjamin Franklin Key Award, the 2012 Intel Early Faculty Career Award and was selected by the National Academy of Engineering for the 2012 US Frontiers of Engineering.