PRO 04 Virgil: towards certified sensor nodes

PRO 04.1 People
- Principal Investigator Jens Palsberg
- Xiaoli Gong, Graduate Student, Computer Science, UCLA
- Kannan Goundan (funded by CENS), Graduate Student, Computer Science, UCLA
- Jens Palsberg, Professor, Computer Science, UCLA

PRO 04.2 Overview
A medical device should not crash or confuse. A device crash can be anything from inconvenient to life threatening, while confusing device behavior can lead a user to draw an incorrect medical conclusion. We envision a certification tool that can meet challenges related to space bounds, soft-real-time response, life time, and meaningful results. We aim for both fundamental advances in programming language design and static error checking, as well as progress on how to do applications programming for medical monitoring devices. Our goal is to take a major step towards design for certifiability and to bring closer the day when the FDA will use static error checking tools frequently and routinely. A medical monitoring device collects data from the body, carries out local computation, and sends data to an external computer. Together, the device and the external computer form a small sensor network with a few sensors and one base station. We are in the early stages of a NSF-funded collaboration with Majid Sarrafzadeh at UCLA whose group has built software for four monitoring devices that were then tested by doctors and patients in the UCLA Medical School. Majid’s devices monitor such things as pressure changes in the upper urinary tract, myotatic stretch reflex, neurological disorder, and diabetic foot ulcer (see Figure 1). The devices are small and the software is typically on the order of a few thousand lines of source code. Eventually we want to be able to certify the software in Majid’s four devices.

![Figure 1: A device for monitoring diabetic foot ulcer.](image)

PRO 04.3 Approach
Our goal is to develop
- a domain-specific language that will encourage design for certifiability and make certification easier, and
• domain-specific tools for certifying the four key properties of space bounds, soft-real-time response, life time, and meaningful results.

In particular, we want the tools to do static error checking, that is, certify the software without running the software. The certification tools will guarantee that certain problems cannot occur; and rigorous testing can then focus on problems that were left unaddressed by the certification tools. The certification tools will increase our confidence in medical devices and help decrease the scope, duration, and cost of the testing effort.

PRO 04.4 System(s) Description and/or Experiments
Our own language Virgil is our starting point for designing a new domain-specific language. Virgil is a statically-typed, object-oriented language in the tradition of C++, Java, and C#. Virgil is designed specifically for high-level, type-safe systems programming, including programming of device drivers for sensors, radios, timers, analog-to-digital converters, etc., and is targeted to run on tiny devices such as sensor nodes.

PRO 04.5 Accomplishments
We have written drivers in Virgil for all the Mica2 devices. The final driver that we completed was the radio device driver which unsurprisingly turned out to be a major challenge. As a result, we now have the entire base functionality of TinyOS written in Virgil. Software can now control a sensor node using Virgil code alone. We have also studied approaches to extending Virgil from being a language for programming single nodes to become a language for programming an entire network of sensor nodes. We are focusing on X10, an object-oriented language from IBM. The X10 language contains two key constructs for concurrent programming called async and finish that we believe can be valuable for programming an entire sensor network. In collaboration with two IBM researchers, we have published a paper in OOPSLA 2008 on the X10 type system that we hope will be directly applicable to our language design and software certification effort.

PRO 04.6 Future Directions
We will continue our work on porting the software in Majid’s four devices from NesC to Virgil.

We will investigate how to implement the functionality of SOS in an extension of Virgil. We will try to extend Virgil to support programming of an entire sensor network. We will implement tools for certifying key properties of Virgil programs.

PRO 04.7 Other Invited Presentations