WG2: Programming IoT: Models, interactions, testing, debugging

James Landay, Ben Zorn, Vikram Iyer, Jack Kolb, Peter Bodik, Steve Myers, Edward Wang, Klara Nahrstedt, James Fogarty, Rahul Bhattacharyya

State of the art

Industrial- oil fields, farms, factories

Platforms: Azure IoT, AWS

Constraints: existing machines, standards, well-defined goals

+ Stream/workflow style blocks

- Low level, hard to optimize

Home- security systems, elder care,

Platforms: Smart Things

Constraints:

+ Good interface, plug and play products

+ Simulator, community

- Hard to debug

- Siloed vertical stacks (Samsung, Apple)

Problems with current platforms

Common problems:

Explaining why something happened (need recording to explain)

Modeling “what-if” scenarios

Ensuring that the worst-case scenarios never happen

- Basic: fail-safe modes
- Harder: Understanding objects, how they interact, and what is unsafe

Formal verification

Homes

What happens when you have visitors?

Different levels of abstraction (large company, contractor, end user)

Public spaces- What happens when one person’s IoT system bothers others?

More problems

Challenges with resolving conflicts
Between devices requiring a shared resource

- Between rules that have conflicting goals
- Between applications
- Between interacting systems (a personal IoT interacts with the city IoT)
- Negotiation of data sharing between interacting IoT systems

Challenges understanding and managing data

- What data is collected and who decides what is collected?
- How does an end-user understand the cost/benefit of data collection or not

Biggest Opportunities

Need common ways to describe the system at the logical and physical level
Compiler: resolves logical to physical

Refinement levels: framework, customized solution, and end-user (like staged computation)

Challenge: person it matters to the most knows the least about programming

Requirement: different input and output modalities (for example by demonstration, in an IDE, viewed in a simulator, physically executed)

Grand Challenge: Self-driving Home Care

Goal: Create a home that can support an elderly person on its own

Who programs this?
- Large companies
- Adult child/caregiver,
- Individual under care

How do we personalize systems?

Understanding and influencing emotional state

More complicated version- person goes outside

Next Steps: Mini Grand Challenge

PL + IDE for next generation IoT systems
Hide complexity of devices, registration, enlistment, etc. as much as possible
Take existing system (e.g., Smart Things) and reimagine it with a set of clean abstractions

- Accounts for different levels of design (framework, etc.)
- Single abstraction that recognizes roles and levels of expertise, provides high-level description
- Tool chain that includes automatic optimization, safety, security, and correctness analysis
- Different modes of training/testing depending on role