Interoperation and Adaptation for Real-World Computing

Johannes Helander, Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

jvh@microsoft.com, http://research.microsoft.com/invisible, +1-425-882-8080

ABSTRACT

This paper discusses a consumer centric view of cyber physical systems and its requirements. It proposes using adaptation through observation based predictions to achieve predictable time and reliability. It proposes new directions in security research that is based on consumer electronics needs. It stipulates that interoperation is crucial and new methods for programming what happens between machines rather than within machines, are required. Finally, systems and theory must meet and verification and online model checking is proposed to achieve an increase in robustness.

BACKGROUND

Emerging consumer electronic devices hold great promise in making our lives more comfortable, aid in everyday health and independent living, as well as in producing economic output. Unfortunately the current generation of consumer devices are difficult to configure and use, they do not interoperate well with each other and the rest of the world, and expose the user to privacy and security concerns. Consumer devices interact with the physical real world either physically such as in sensing and robotics; or with human senses and instructions. Since they interact with the real world the interaction also needs to be temporally connected to that world, in other words be real-time. Since people are increasingly relying on the devices and on combinations of the devices and other computers, a higher level of reliability is also required from them. Combining the demands with low cost and high volumes can be an engineering and business challenge in addition to the open research questions.

If the consumer devices would be able to self-organize into meaningful combinations so as to fulfill human intents, however unclearly given, the devices would not just be individual gadgets but could actually enable new scenarios and functionality, bringing the consumer market to the next generation.

CHALLENGES

Making consumer devices interoperate, discover each other's capabilities, plan the right course of action, do it predictably in real time, and doing it securely poses a number of challenges:

- Making devices and other computers interoperable and auto-configurable, including real-time.
- Describing, programming, and modeling distributed processes and human behaviors
- Developing new consumer centric security models that do not require centralized authority and complex configuration steps.

BARRIERS

Current popular research approaches are unlikely to produce the solutions necessary since they are

- Stuck on programming languages and rephrasing old approaches in new vocabularies.
- Aiming too low in functionality and hardware requirements. Performance issues are important but should not dominate the agenda.
- The divide between theoreticians and practitioners. In this case modeling and verification ought to become part of the software process and runtime rather than being almost entirely separate.

More focus is needed in what happens between machines and how applications spanning multiple devices and computers can be created. It is fairly well understood how to program a single computer—doing this in a new language or with the smallest possible computing power is not ambitious enough.

Devices should also take advantage of the larger computers around them, thus it is imperative that the small devices are *not separate* from other, higher-end, computers but part of the solution.

PROPOSED APPROACHES

The goal is to create devices that automatically discover and adapt to their environment and the humans around them. This is achieved through the following:

- Layered approaches to concepts, languages, protocols, and designs.

  Extend network protocols to new horizontal layers. Separate syntax from semantics. Build everything out of clearly defined components. Each layer or component could be replaced separately. This means that instead of trying to settle on one language, there are likely to be many. XML is a good example of a two-level language. XML is just syntax. What the data means is described separately (e.g. schema, WSDL). The descriptions can be improved separately from the content. The syntax can be replaced separately from the meaning. This approach obviates the need for vertical protocol stacks, considered harmful, such as in Bluetooth, and the need and desire for a common programming language, such as Java or C#.

- Pattern descriptions for distributed processes and human behavior
  - Output: Plan distributed application execution based on a description of nodes, messaging patterns, timing and confidence requirements.
  - Input: Match observations to models that describe human behavior or environmental processes. Match instrumentation induced context history to the planned application, creating a feedback loop.

  The patterns can be viewed as a serialization of a model, where the planning and matching is the task of relating the outside world to the modeled world.

- Use probabilistic methods. An absolute algorithm or plan is not practical in a distributed system due to high level of change and chaotic environments. A probabilistic system may be more reliable. If a stochastic process or model can predict the likelihood of a failure, preventative action can be taken before the failure, rather than after the fact in the system that "cannot fail". Borrow techniques from economics, such as technical analysis, and other disciplines in creating better predictions. Real-time can be redefined in statistical terms, making the discussion of hard versus soft real-time obsolete.

- Online modeling. Combine offline modeling and verification with online models. Derive online verifier from offline model. Use observed context history to drive models and expectations.

- Indirect, physical, and natural user interfaces. Instead of a mouse and screen future devices ought to interact with humans in natural ways. This means that the computer should model the human behavior and determine the right action based on the model, triggered perhaps by touch or a nudge.

- Metadata programming and middleware.

  Create new programming models. Explicitly deal with metadata, such as component, interface, behavior, human sense interaction, etc. Analyze metadata and drive middleware based on it.

- Security based on social protocol. Federate trust rather than centralize. Current security models are akin to an onion, where the inner layers are protected from the outside. A civil society is, however, based on partial trust and partial mistrust without clear hierarchy. This disconnect between models and needs make security work badly. New models that match the needs should be developed.

OBSERVATIONS FROM ATTEMPTS

The author has made preliminary attempts at many of the issues proposed, which gives the motivation for the suggestions. Observations include: Even very simple stochastic processes are meaningful in predicting future timing requirements based on observations of the execution. XML can be efficient in small devices and is surprisingly expressive, even if admittedly ugly to look at. Mixing of multiple programming and data representation languages in one system is not a problem. Separate descriptions of various aspects of the overall system make sense as they can be described in a domain-specific way and

middleware can combine. Interoperation is actually useful and leads to real applications. Control of physical systems must not be too synchronized, rather sending a script to a robot with micro-movements locally seems better. Context histories can be used to predict behavior. Aggressive and strict componentization is an engineering necessity, including systems software.

MILESTONES

1. Define patterns, models, layers. Experiment with multiple descriptions.
2. Develop new security models. Create information models for #1.
3. Develop pattern matching, verification, and prediction mechanisms.

Bio

Johannes Helander is a researcher at Microsoft in the area of Invisible Computing. His current interests include embedded web services, communication middleware, componentized real-time systems, consumer centric security, context history based prediction and modeling. Current application areas include interoperable discovery between device classes, IT management for out-of-band and embedded devices, and multi-language support for embedded platforms.

Prior to Microsoft Johannes was at Helsinki University of Technology where he amongst other things created the first freely available Unix implementation for Mach, including simultaneous support for binaries from multiple Unix variants, and adapting Real-Time Mach to a Nokia telephone switch.

At Microsoft Johannes worked on the initial interactive TV project, firmware for 3D graphics acceleration, the first TCP direct path, a distributed mobile object system for Java; started an embedded C# runtime product group; co-designed initial smart watch software; created a component based RTOS, being used in management solutions; created the first Embedded XML Web Service implementation; designed and implemented a remote shell communicating over web services for Windows Vista; launched the Embedded WS-Management Toolkit, and prototyped a non-hierarchical trust and security system as well as an auto-adaptive distributed real-time scheduler.

Recent publications from author on embedded systems

1. Helander J., Deeply Embedded XML Communication: Towards an Interoperable and Seamless World, Proceedings of the *5th ACM international conference on Embedded software*, Jersey City, NJ, September 2005.
2. Helander J., Sigurdsson S., Self-Tuning Planned Actions: Time to Make Real-Time SOAP Real, Proceedings of the *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, Seattle, May 2005.
3. Helander, J. and Xiong, Y., Secure Web Services for Low-cost Devices, *8th IEEE International Symposium on Object-oriented Real-time distributed Computing*, Seattle, May 2005.
4. Helander, J., Exploiting Context Histories in Setting up an e-Home, *First International Workshop on Exploiting Context Histories in Smart Environments* (in conjunction with Pervasive 2005), Munich, Germany, May 2005.
5. Nosov, A, Helander, J., Management of Hardware Resources in the Datacenter Using Embedded Web Services. Microsoft whitepaper at *Windows Hardware Engineering Conference*, Redmond, WA, May 2006.
6. Preden, J. Helander J., Auto-adaptation Driven by Observed Context Histories, *Second International Workshop on Exploiting Context Histories in Smart Environments* (in conjunction with UbiComp 2006), Newport Beach, CA, September 2006.
7. Kantee, A, Helander J., Implementing Lightweight Routing for BSD TCP/IP. *EuroBSD Conference 2006*, Milan, Italy, October 2006.