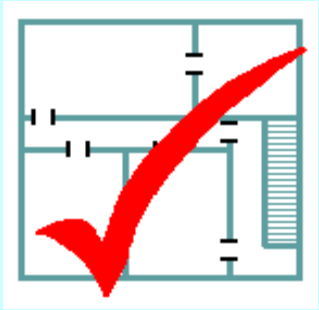
Suppressing Ubicomp Skirmishes

Dr David Greaves

Lecturer
University of Cambridge
Computer Laboratory

and

Chief Scientist
Tenison EDA



A Vision of Evolution for UbiComp

1. A myriad of devices connected to the network.

2. All devices are connected and share a common, all-pervasive, middleware.

3. Mixed reality interaction is the norm.

A Device: A collection of Pebbles and a Canned App

Let’s look at what a modern TV set contains:

- 1. The following separate devices, each of which can be individually useful in a networked home:

  - RF Tuner
  - Colour Display
  - Ni-Cam Audio Decoder
  - Power Amplifier
  - Surround Sound Decoder
  - IR Receiver
  - Teletext Decoder
  - MPEG Decoder
  - Programming Memory
  - Front Panel User Interface

- 2. A canned application that joins the components.

Device API Evolution

1. Early, wire protocol RPC APIs:
   - Sun RPC, CORBA, HAVI, MOSTNET

2. Evolvable XML APIs with Reflection:
   - UPnP, (SNMP), XMLRPC, SOAP, WSDL

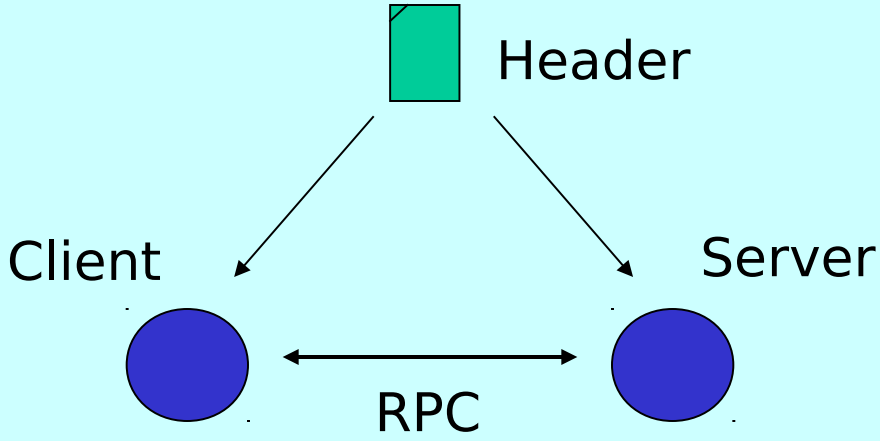
3. Self-Assembling Directory Services
   - UPnP, LDAP, SCP, SSDP, INS, …

4. Namespaces and Ontologies
   - OWL, OWL-S, DAML, RDF

5. Code Reflection

Binary Wire Coding

- First generation RPC uses binary coding
  - eg. Sun RPC, Java RMI and Corba

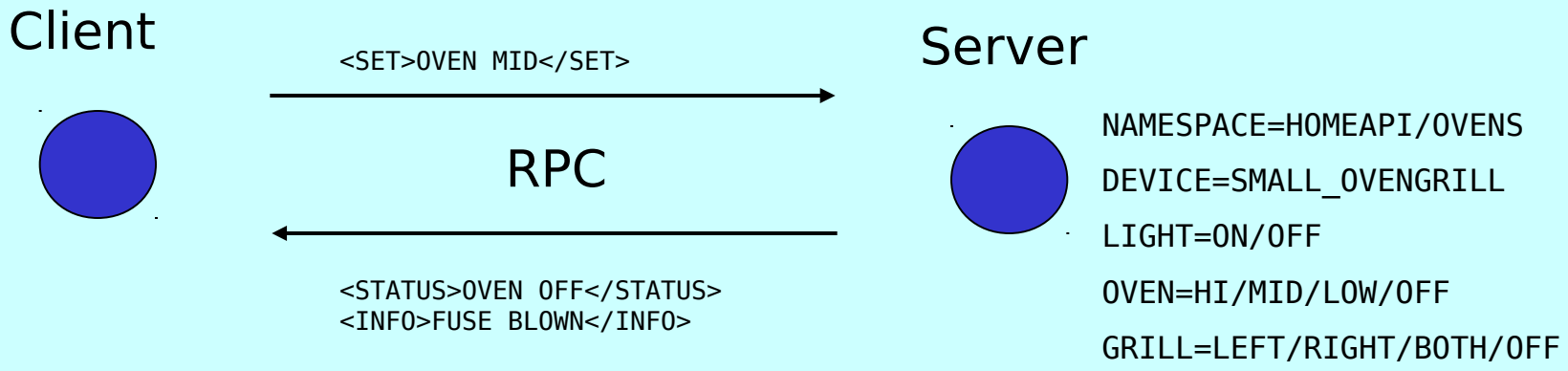


Any change in the shared header file makes interoperation impossible.

Is this good for long-lifetime components ?

XML Wire Coding

- Current Generation uses XML Coding
  - eg. SOAP, XMLRPC, Semantic Web



ASCII English is Human Readable.

New fields are not fatal.

Automated Directories

- Devices register in an ad hoc database
  - eg. UPnP's SSDP and our own Oxygen system
- Devices can be found by service offered
  - eg. A colour printer on floor 3 west.
- Retrieval by conjunction of predicates

Few successful deployments.

Unexpected behaviours.

Load balancing/path finding unsupported.

Ontology Mapping

Ontologies can be mapped:

- Orange Jam
- Living-Room
- Front-Door-Bell
- StartUP

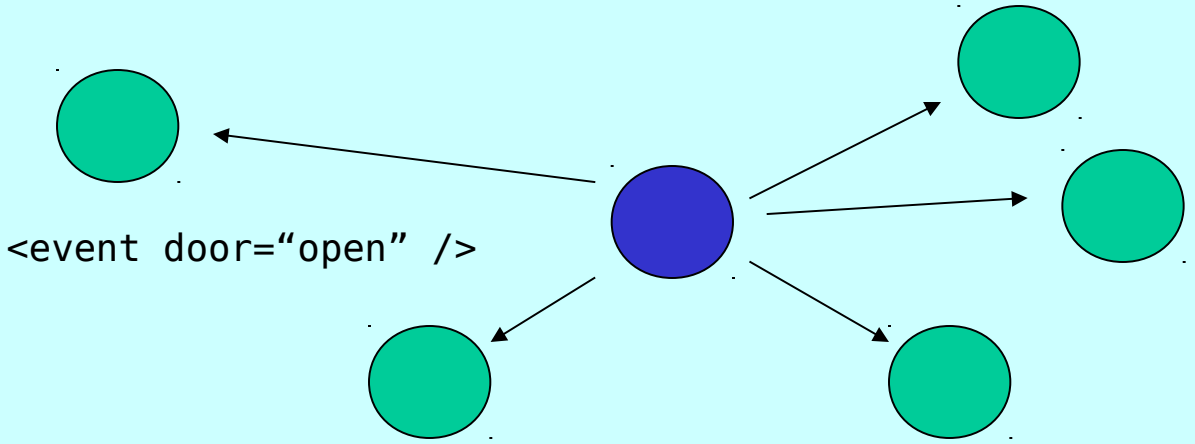
- Marmalade
- Lounge
- FrontDoorBell
- SwitchOn

Ontologies can be nested:

Claret isa RedWine isa Wine isa Drink isa Food

Ontology Web Language Service (OWL-S)

Asynchronous Eventing



- Asynchronous notification of events
- Publish/Subscribe or broadcast over LAN

Does not block source if destination is dead or slow.

Advanced federation and relaying architectures exist.

Controllers Vs The Controlled

- API Reflection is now a Mature Technology
- It will be further deployed (?)
  - X-by-wire, Field Busses, Sensor Networks, CAN.
  - EDDL, XDDL, Embedded Systems

- Code Reflection has seen virtually no work!
  - Canned Applications
  - Self-hydrating applications
  - User scripting

Code Reflection

- A device must expose the proactive behaviour of its canned application(s)
  - Actual source code (constrained language)
  - Proof carrying actual source code
  - Summary of behaviour
    - E.G. I will not send control messages when I am in standby mode.
    - E.G. I am always off between 1:00 and 5:00.

- Device is banned from full operation unless proof obligations are met.

User Scripting

- Record both Simpson's shows tonight and charge to Pam's pay-per-view account.
- Create a video call to Peter of best quality.
- Whenever the doorbell is pressed during darkness turn on the porch light for 10 minutes.
- Whenever Lulu comes home, play introduction to Thriller on all loudspeakers downstairs.

David Greaves - EP6 @ Savoy Place

Rules of the Domain

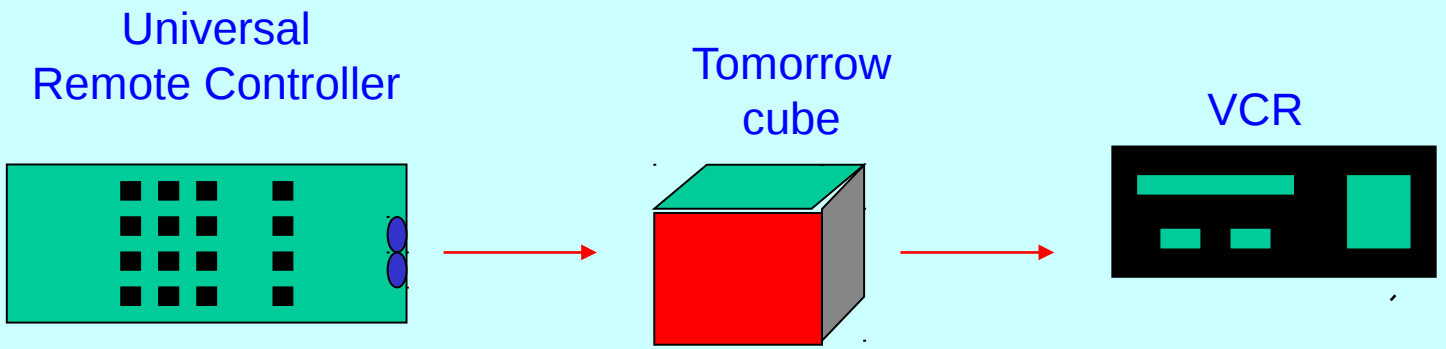
- No rule should issue a command under the same circumstances where another rule issues the counter-rule.

- Jonny is not allowed to spend more than 2 pounds per day on pay-per-listen.

- Fire Alarm must mute all music sources.

- The front gates must always be remotely openable by some method or other.

MultiView/Tangible Scripting



- A cube ‘modifies’ an IR controller ray.
- Multiple Views with Inspectors/Editors must be supported for each script.
  - eg. Voice, Text, Tangible, Gesture.

An Approach: *Push Logic*

- All applications and rules are held in Push Logic.
- The world is made of disjoint ‘Domains’ that devices and rules enter and leave.
- Devices and rules enter on a first-come basis and so take priority over subsequent disagreeing arrivals
- Push Logic rules can be model-checked in tens of milliseconds on embedded processors.

Push Logic Semantics

- Every application is encoded as/converted to a list of state transition rules.
- Rules are re-hydrated to use only static bindings.
- Every time of day mentioned is a partition, forming disjoint temporal extents.
- All rules of behaviour are expressed to safety or liveness form.
- A symbolic model checker finds conflicts and satisfaction.
- All rules are interpreted in Herbrand form so as to tolerate arbitrary state space growth.

Progress So Far

- A large number of Pebbles created
- Shared codebase with MIT Oxygen
- Various experiments in multi-view script manipulation
- Four or five declarative programming approaches implemented.

Research Challenges

- Can embedded RAM afford XML protocols ?
- Can embedded processors run the model-check phase efficiently ?
- How much application code can easily be ported to a *formal-proof-amenable* form ?
  - eg. Can we implement a full TiVo PVR ?
- How to modify conventional APIs to hold less state (eg. Posix Sockets) ?
- How to merge domains on the fly ?
  - eg. Two sets of train carriages become a single train.

Conclusions

- Still vast divides between
  - ESL/SystemC hardware modelling
  - Micro-controllers programmed in assembler
  - X-by-wire and safety critical
  - Sensor networks
  - Workstation middleware

- Now no fundamental obstacle to closing this divide.

The End

- David.Greaves@cl.cam.ac.uk
- www.cl.cam.ac.uk/Research/SRG/HAN/Pebbles
- The Pebbles, AutoHAN and Oxygen O2S Teams