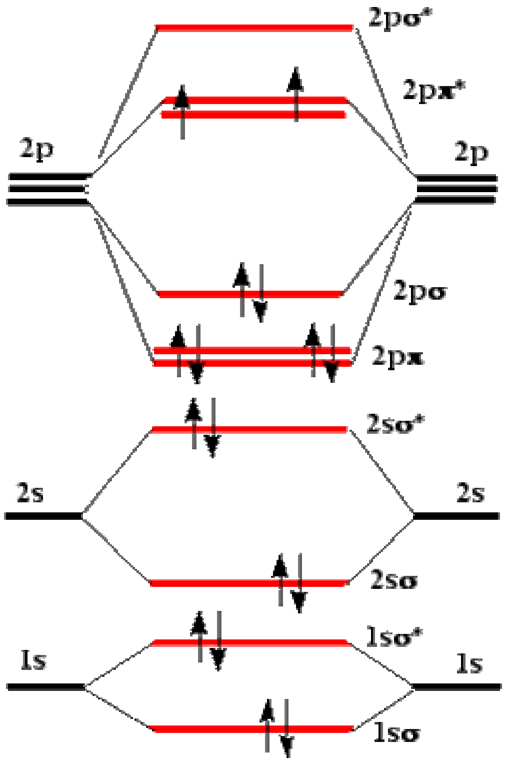
MIT PROJECT OXYGEN
PERVASIVE, HUMAN-CENTERED COMPUTING

Distributed Pervasive Applications:

easing the pain

A framework for dynamic assembly of Oxygen applications

27 January, 2003



Steve Ward

LCS, MIT

L C S

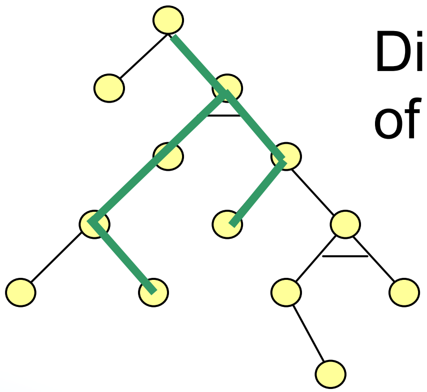
O_2S

Some local history.

TeleConference(Anant, Steve)

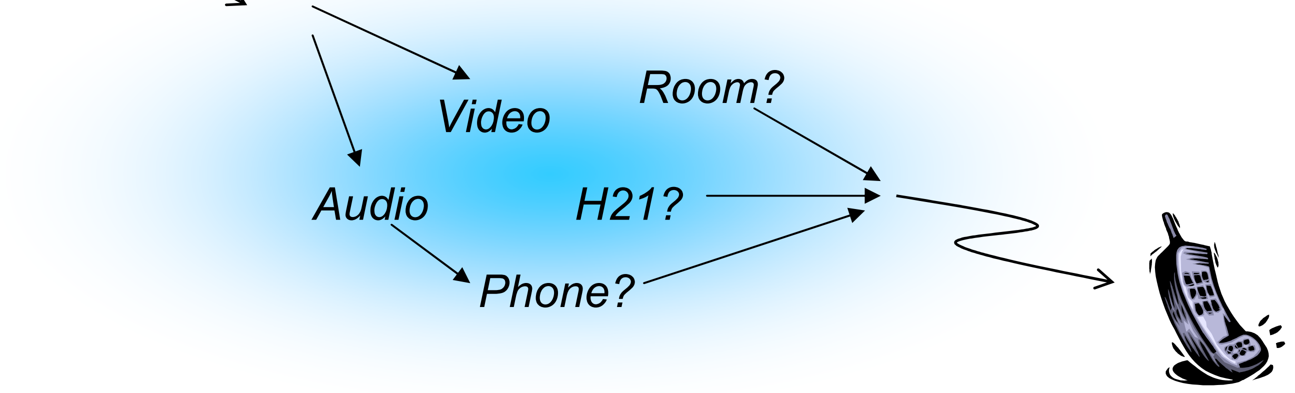
Goal-oriented Software Architecture:

Standardized GOALS as commodities



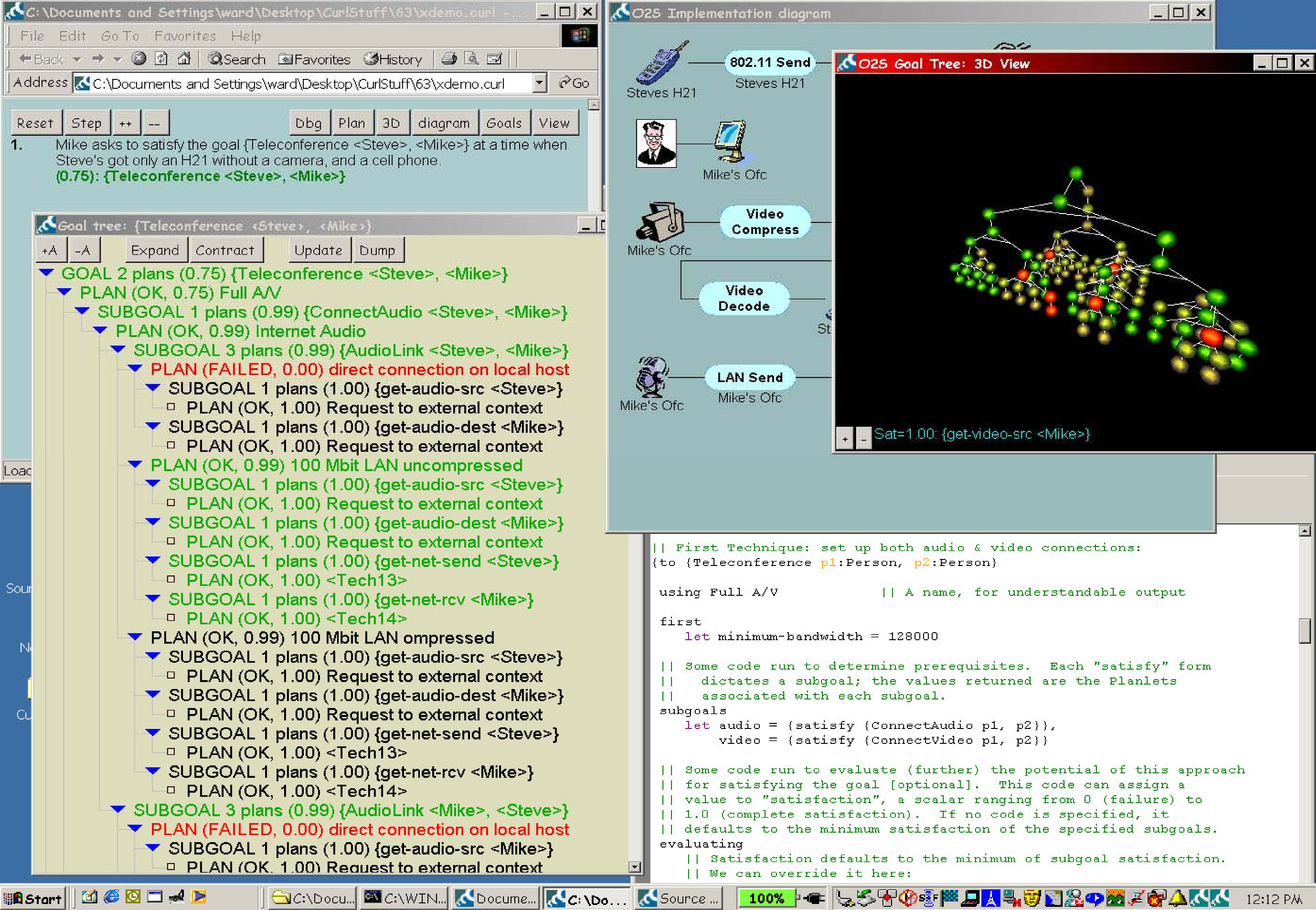
Distributed database of TECHNIQUES

Achieving goals by pursuit of sub-goals



Steve

O_2S

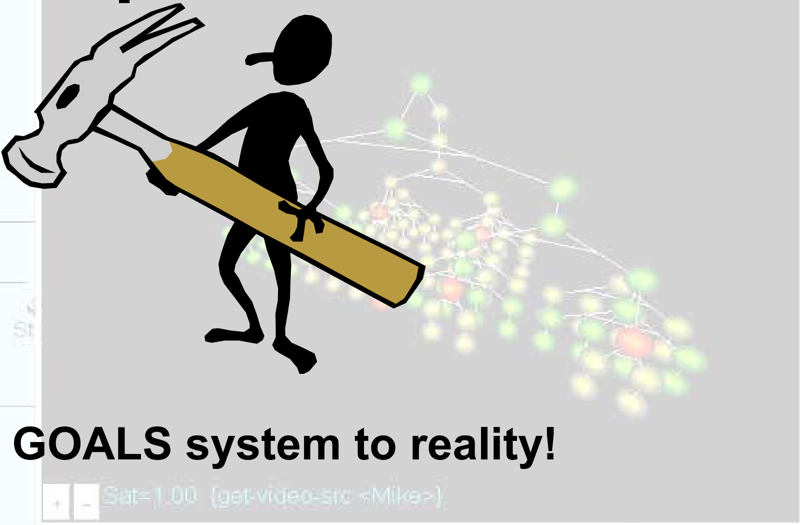

C:\Documents and Settings\ward\Desktop\CurlStuff\63\xdemo.curl - ...
File Edit Go To Favorites Help
Back Search Favorites History
Address C:\Documents and Settings\ward\Desktop\CurlStuff\63\xdemo.curl
Go
Reset Step ++ -- Dbg Plan 3D diagram Goals View
Goal tree: {Teleconference <Steve>, <Mike>}
+A -A Expand Contract Update Dump
O2S Implementation diagram
Steves H21
802.11 Send
Steves H21
Mike's Ofc
Mike's Ofc
Video Compress
Video Decode
St
Mike's Ofc
LAN Send
Mike's Ofc
O2S Goal Tree: 3D View
+ -
Sat=1.00: {get-video-src <Mike>}
Load
Sour
N
Cu
Start
C:\Docu...
C:\WIN...
Docume...
C:\Do...
Source ...
100%
12:12 PM

1. Mike asks to satisfy the goal {Teleconference <Steve>, <Mike>} at a time when Steve's got only an H21 without a camera, and a cell phone.
(0.75): {Teleconference <Steve>, <Mike>}

```
GOAL 2 plans (0.75) {Teleconference <Steve>, <Mike>}
  PLAN (OK, 0.75) Full A/V
    SUBGOAL 1 plans (0.99) {ConnectAudio <Steve>, <Mike>}
      PLAN (OK, 0.99) Internet Audio
        SUBGOAL 3 plans (0.99) {AudioLink <Steve>, <Mike>}
          PLAN (FAILED, 0.00) direct connection on local host
            SUBGOAL 1 plans (1.00) {get-audio-src <Steve>}
              PLAN (OK, 1.00) Request to external context
            SUBGOAL 1 plans (1.00) {get-audio-dest <Mike>}
              PLAN (OK, 1.00) Request to external context
          PLAN (OK, 0.99) 100 Mbit LAN uncompressed
            SUBGOAL 1 plans (1.00) {get-audio-src <Steve>}
              PLAN (OK, 1.00) Request to external context
            SUBGOAL 1 plans (1.00) {get-audio-dest <Mike>}
              PLAN (OK, 1.00) Request to external context
            SUBGOAL 1 plans (1.00) {get-net-send <Steve>}
              PLAN (OK, 1.00) <Tech13>
            SUBGOAL 1 plans (1.00) {get-net-rcv <Mike>}
              PLAN (OK, 1.00) <Tech14>
          PLAN (OK, 0.99) 100 Mbit LAN ompressed
            SUBGOAL 1 plans (1.00) {get-audio-src <Steve>}
              PLAN (OK, 1.00) Request to external context
            SUBGOAL 1 plans (1.00) {get-audio-dest <Mike>}
              PLAN (OK, 1.00) Request to external context
            SUBGOAL 1 plans (1.00) {get-net-send <Steve>}
              PLAN (OK, 1.00) <Tech13>
            SUBGOAL 1 plans (1.00) {get-net-rcv <Mike>}
              PLAN (OK, 1.00) <Tech14>
        SUBGOAL 3 plans (0.99) {AudioLink <Mike>, <Steve>}
          PLAN (FAILED, 0.00) direct connection on local host
            SUBGOAL 1 plans (1.00) {get-audio-src <Mike>}
              PLAN (OK, 1.00) Request to external context
```

```
|| First Technique: set up both audio & video connections:
{to {Teleconference p1:Person, p2:Person}

 using Full A/V            || A name, for understandable output

 first
    let minimum-bandwidth = 128000

 || Some code run to determine prerequisites.  Each "satisfy" form
 ||   dictates a subgoal; the values returned are the Planlets
 ||   associated with each subgoal.
 subgoals
    let audio = {satisfy {ConnectAudio p1, p2}},
        video = {satisfy {ConnectVideo p1, p2}}

 || Some code run to evaluate (further) the potential of this approach
 || for satisfying the goal [optional].  This code can assign a
 || value to "satisfaction", a scalar ranging from 0 (failure) to
 || 1.0 (complete satisfaction).  If no code is specified, it
 || defaults to the minimum satisfaction of the specified subgoals.
 evaluating
    || Satisfaction defaults to the minimum of subgoal satisfaction.
    || We can override it here:
```

Steps toward Goals…



We've built a (more) real version…

… but that's not today's topic.

Challenge:

Connecting our GOALS system to reality!

- **Diversity of devices, hosts, failure modes**
- **Lack of notification guarantees to drive planning process**
- **Unbearable debugging environment**
- **Maze of platform, OS, language dependencies**

- **NEEDED:**
  - **Coherent target model for planning**
  - **Robust, platform- and language-independent implementations**

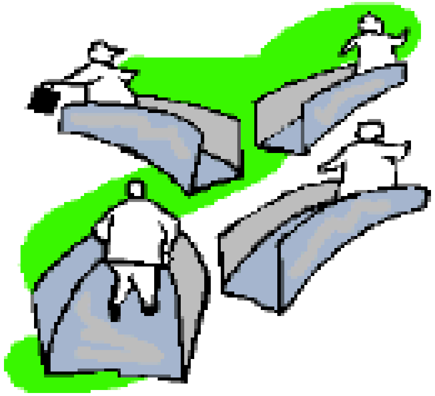
O_2S

Building distributed applications…

… a notoriously hard problem!

A few of the reasons:

- **Distributed state.**
  - ***System "state" may not be a well-founded notion!***
- **Failures of remote resources, communications…**
  - ***User turns off his iPAQ… or***
  - ***Gets into steel-shielded elevator***
  - ***Symptom: silence.***
- **Lack of process hierarchy**



Goal: provide a model that addresses these issues

- **Illusion: "circuit" of interconnected modules, assembled by application.**
- **Simulate localized state, serialized stream of application-related events.**

Levels? Who wants Levels?

Research issue: do we ***want*** a strong abstraction between planning & component levels?

- Alternative component models intermingle these functions, to good effect…
- Some O2 projects – e.g., INS – represent opposite extreme

Issues:

- Planning depends on low-level resources, capabilities
- Efficiency: constrains optimizaton

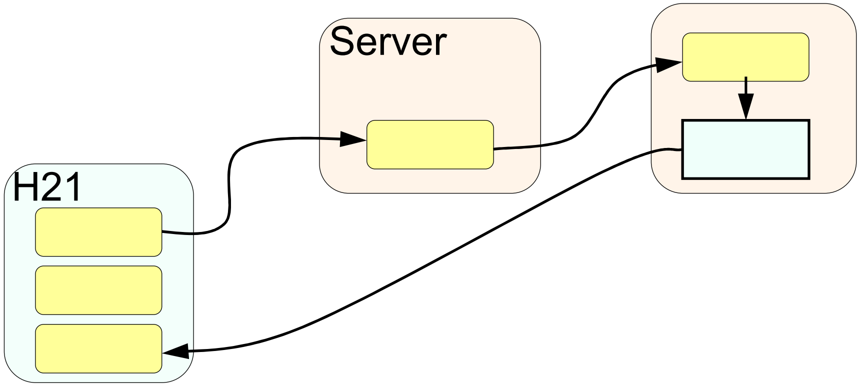
Research Questions:

- *What is the range of applications that fit well into this paradigm?*
- *What are the costs of this abstraction, in real applications?*

Pros:

- POLICY centralized, scriptable
- Ideal target for Goals layer
- Don't buy Goals? Can do planning in C/Java/… code

The O2S Application Model



Application code:

- Assembles a “circuit diagram” of pebbles, connections; then
- Monitors serialized stream of related events
- Interacts with centralized, coherent, synthesized “application state”

What happens if…

- **Some component crashes?**
- **Someone reboots their iPAQ?**
- **Loses network connectivity?**
- **Hits QUIT?**

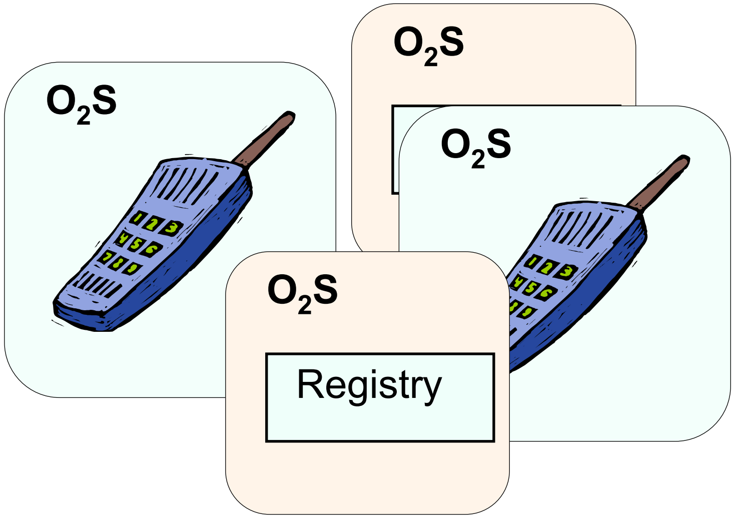
O2S System: presents coherent illusion…

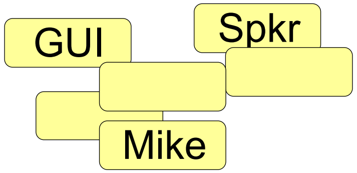
- **Common system code** in each host (device, server, …):
  - Hosts sandboxed ‘pebbles’
  - Reflects pebble state, errors, debugging spew to central app
  - Minimalist mechanism, not Policy
- **Application Framework:**
  - manages “circuit” model;
  - Hides administrative interactions

Our (Fetal) Code Base

Python-based prototype:

- XMLRPC interfaces; apps, planning in JAVA
- Portable host code:
  - O2S Listener (server)
  - Registration/keep-alive
  - Hosting of sandboxed pebbles, specified via URIs
  - Runs on iPAQ, LINUX Servers, Windows(**), …
- Several trivial apps





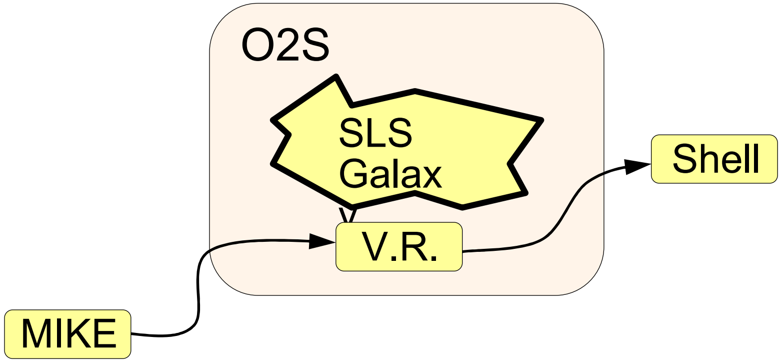
Start at Pebble library:

- Several primitive pebbles for iPAQ (audio in/out, tiny GUIs)
- Placeholder *server* pebbles (voice recognition, email)

Server-side pebbles

O2S System runs on handhelds, desktops, servers, …

- Common framework: Registrant, O2S Server, PebbleHost
- Shared by devices, apps, host/user proxies, services



Example: Voice Recognition

Easy, modular, programmatic access to functions of SLS Galaxy System

- **SLS Galaxy System**
- **Add O2S code, wrap it in a pebble interface: then**
- **App request (incl. grammar) instantiates `voice_recognizer` pebble**
- **Waveform input, text output connected to pebbles elsewhere**

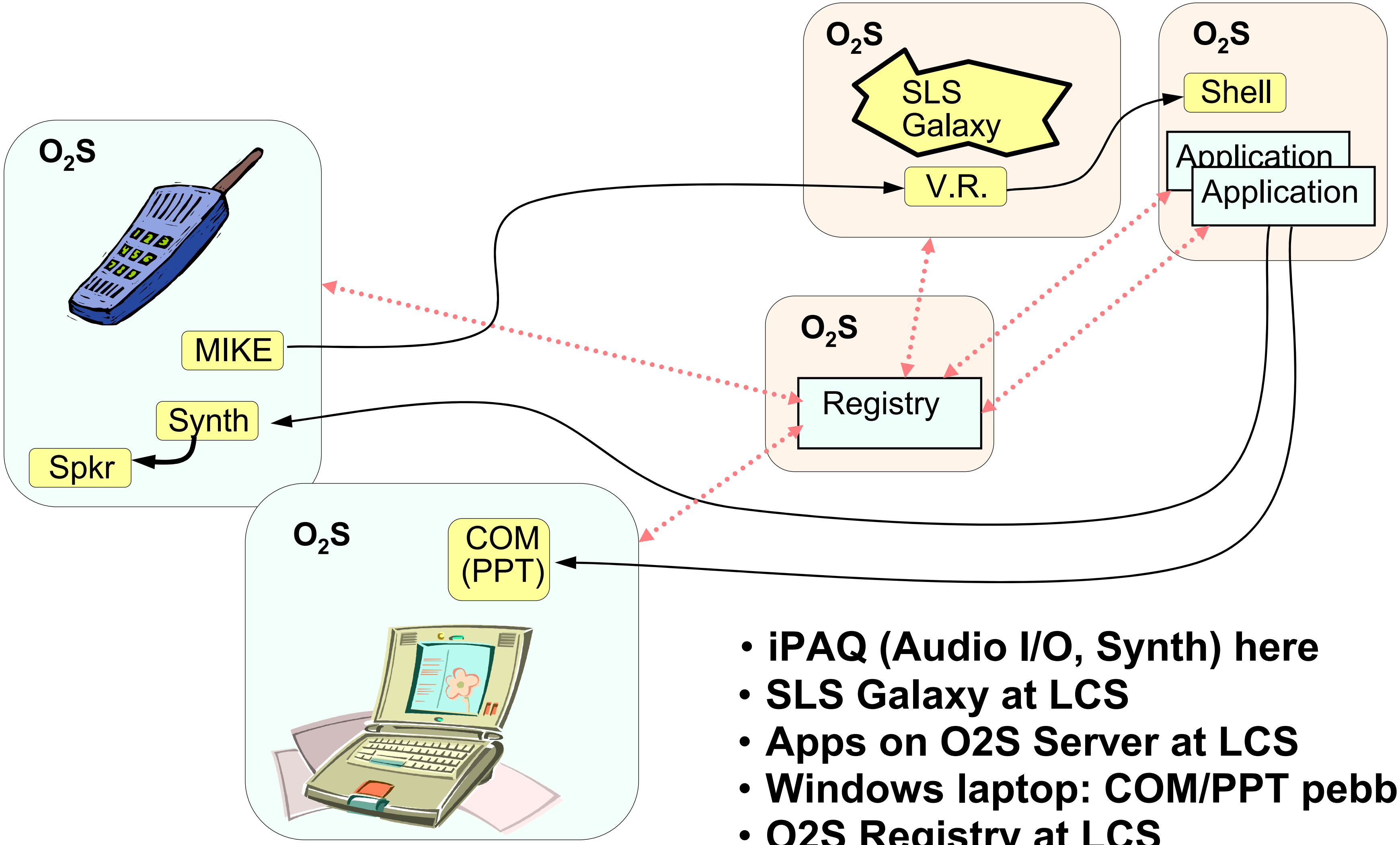
Primitive “Voice Shell”

```
af = AppFramework()

# Instantiate our required pebbles.  By default, any failure
#   shuts down (cleanly) the application:
shell = af.request(af.localhost, 'shell‘)
grammar = shell.request(‘grammar’)
recognizer = af.request(SPEECH_SERVER, 'voice_recognizer', grammar)
voice_in = af.req                                               e‘)

# Make the apropr
af.connect(recognizer.outpu
af.connect(voice_in.output,

# Instantiate a simple GUI
gui = af.request(O2S_CLIENT,

# Then, simply monitor even
while af.status == 'running
    event = af.next_event()
    if event.name == 'button_clicked': break
```

```
<command>
      = {vsh} {quotes} show me my stocks
      | {vsh} {quotes} how is my portfolio doing
      ;
```

```
<command>
      = {vsh} {connect-me-to} connect me to <person>
      ;

<person>
      = Cornelia {colyer}
      = Chris Terman {cjt}
      = Umar Saif {umar}
      = Steve Ward {steve}
      = David Saff {saff}
      = Eric Brittain {ericb}
      ;
```

Tinkertoy-set modularity



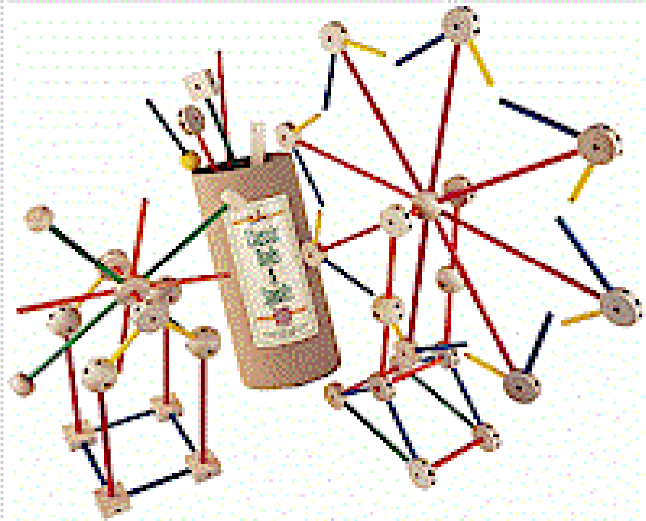
- iPAQ (Audio I/O, Synth) here
- SLS Galaxy at LCS
- Apps on O2S Server at LCS
- Windows laptop: COM/PPT pebble here
- O2S Registry at LCS

Should HP be interested?

The dawning age of pervasive computing…

… invading corporations, hospitals, universities, homes, …

POTENTIAL: Revolution, ala digital HW during 60s-80s



Hardware building blocks:

- Handhelds
- Desktop machines
- Printers & Peripherals
- Instruments

… things HP makes!

Software building blocks:

????

COMING: a "glue" technology…

The *TTL Data Book* for pervasive computing!

What will HP's role be?

O$_2$S