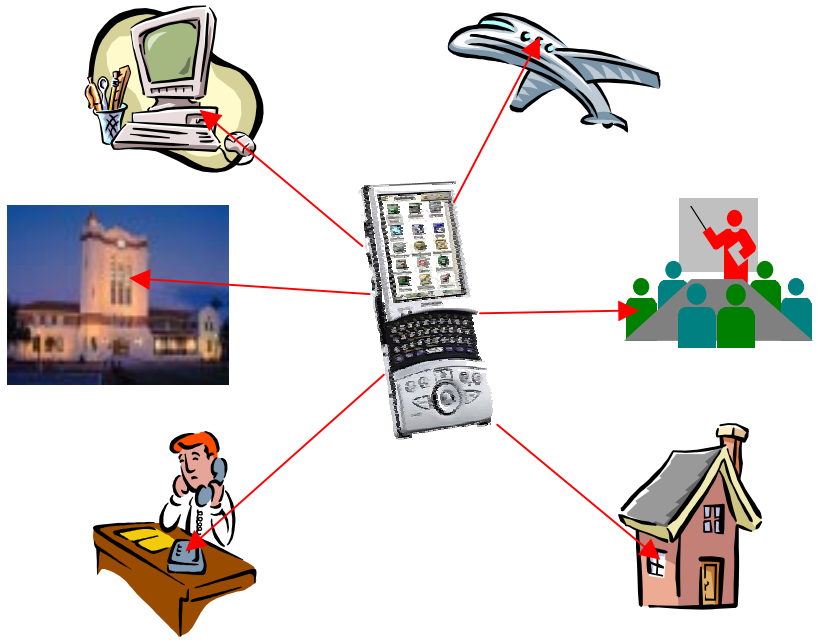


Sun Microsystems Laboratories

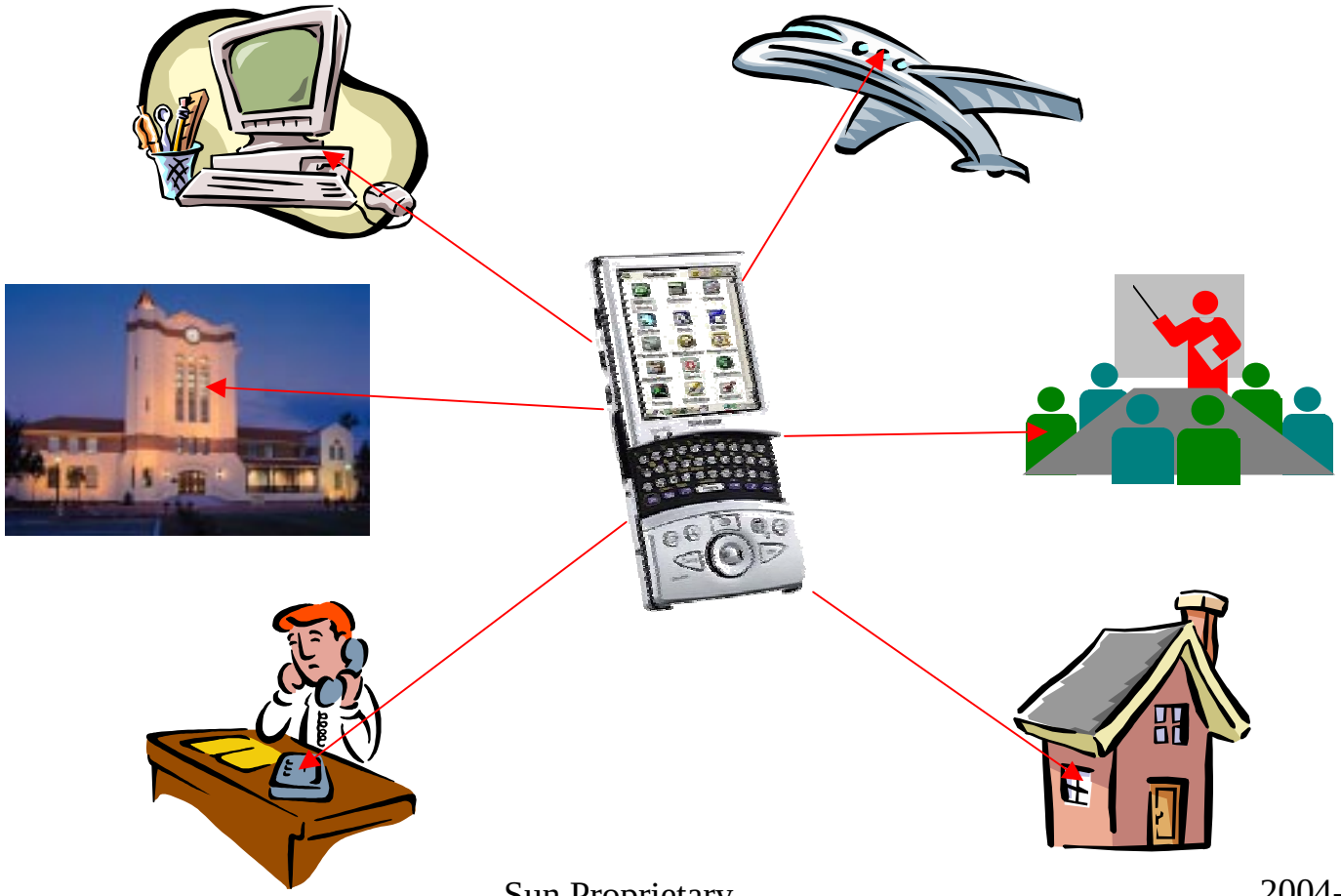
Stephen Uhler

Sun Labs

March 31, 2004

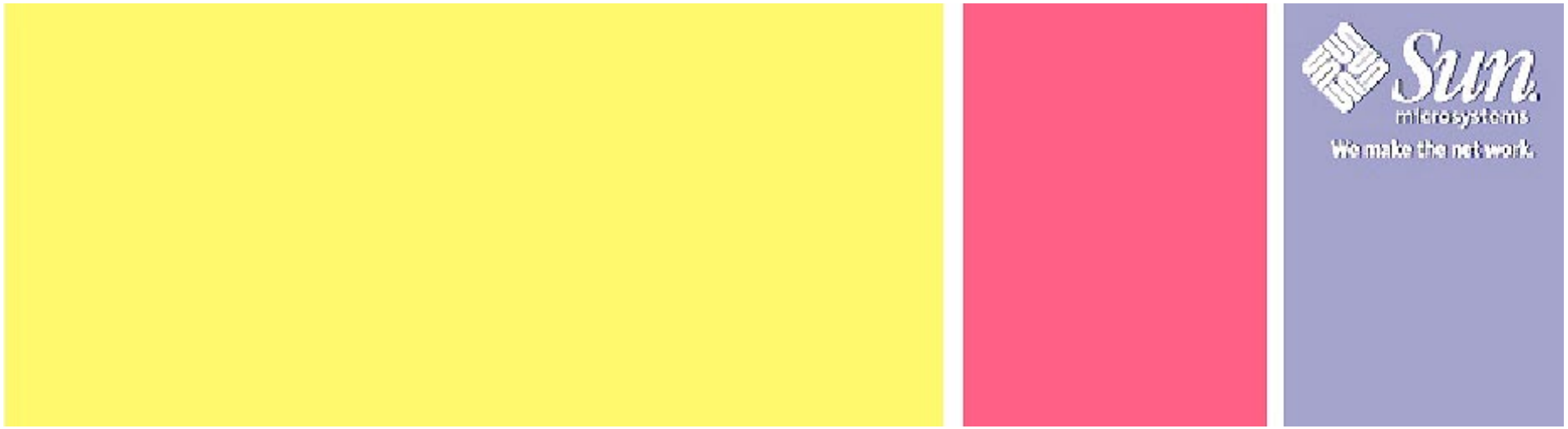
Sun microsystems
We make the net work.

Enterprise Access wherever you are



Enterprise PDA

Introduction and Background



EPDA Project at a Glance

- Vision

  Integrate next generation PDA's into the fabric of the Enterprise computing environment

- Goals
  - If it works in your office, it works on your PDA
  - Java is the development platform of choice
  - Light weight "thin client" architecture

What is an Enterprise PDA ?

- Network enabled PDA
- Enterprise integration of PIM applications
- Access to “Secret Sauce”*

* Key company specific applications

PDA Telephone Convergence

Phone:

- Ubiquitous Network connectivity

PDA:

- Open Development Platform
- PIM Integration
- Text Input, pointing device

Hardware Platform

- Requirements
  - Almost always wireless networking
  - Pocket sized
  - Color display
  - Integrated audio
  - Portable development platform
- Prototype (~~Yopy~~) Sharp Zaurus 5600 ~~5000~~
  - ~~206~~400mhz strong-arm, 320x240 full color display
  - ~~32m~~ 128m+ flash, ~~16m~~ 64m ram, 802.11b + Sprint PCS + GSM
  - Open Source Linux Java (CDC Personal Profile)
    - C, C++, Java (cvm), Tcl/TK, X11

Technical Approach

- ✓ Infrastructure
  - Linux, C, C++ TK, X11
  - Java, audio, networking
- ✓ Sample Applications
  - Understand the ePDA environment
  - Explore the application space
  - Prototype specific customer solutions
- ❑ Universal Client
  - Combine the ease-of-deployment of the web, with the richness of a traditional desktop GUI
  - Survey existing efforts
  - Build prototype(s)

Sample Application Explorations

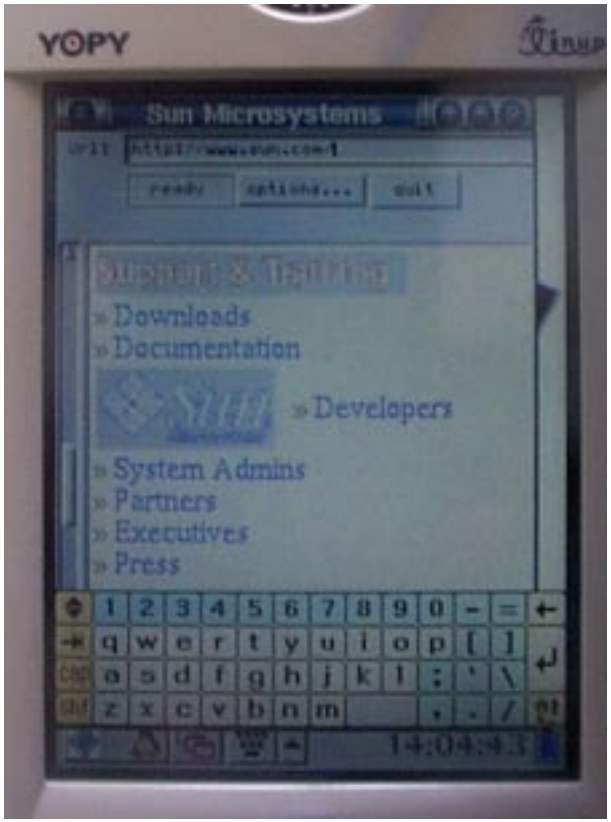
- How do we build new and leverage legacy applications on an ePDA?
  - Dillo – a pan & scan PDA browser
  - Mangler – a page reformatting browser
  - Server side HTML transcoding engine
  - Stock Tracker – Application specific browsing
  - PDA audio conferencing (cb radio)
  - SLIM (instant messaging) client
  - Universal Client

Sample Application Explorations

- Pan & Scan Browser
  - Normal browser with a small viewport
- Lessons
  - Can view almost any web page
  - Only usable for the most desperate

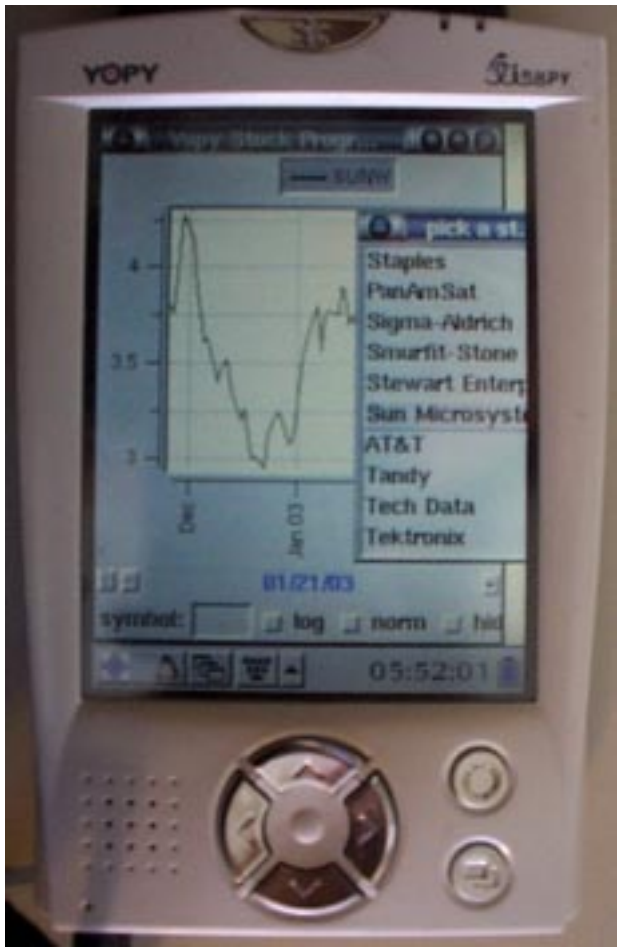
Sample Application Explorations

- Mangling browser
  - Automatically re-layout the content for a smaller display
- Lessons
  - Helps with certain types of web sites
  - Large effort required for comprehensive solution
  - Adding server-side transcoding improves performance but limits applicability



Sample Application Explorations

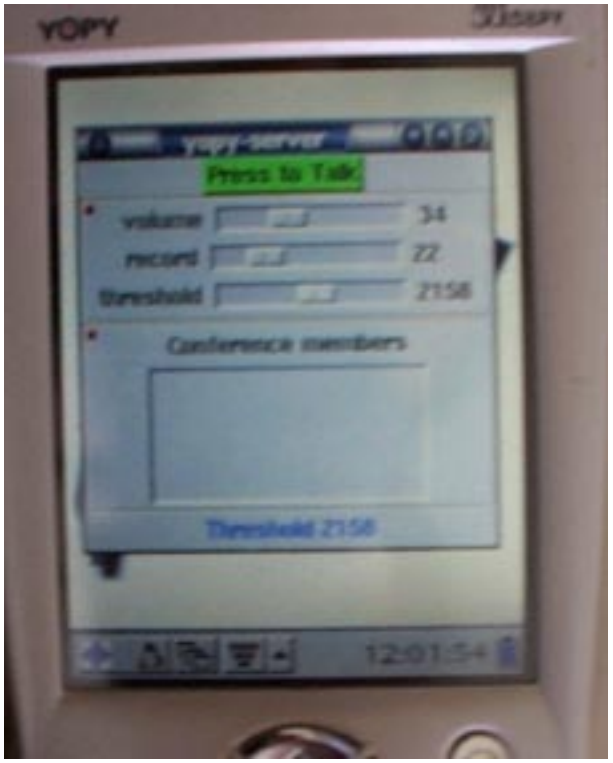
- Stock Tracker
  - Custom “web” client interacts with legacy web server
  - Client “pretends” to be a normal web browser to the server
- Lessons
  - Better user experience than with “stock” browser
  - Writing custom clients is a drag

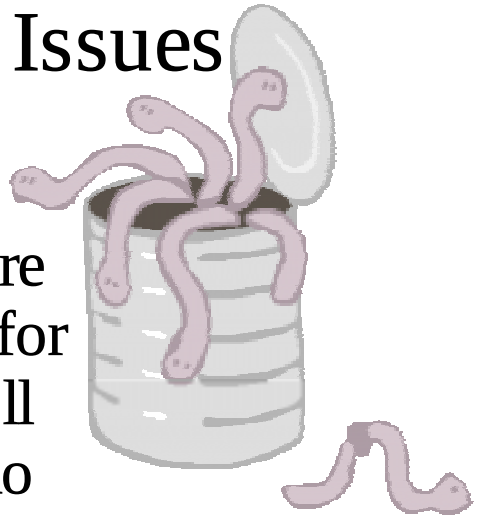


Enterprise PDA

Sample Application Explorations

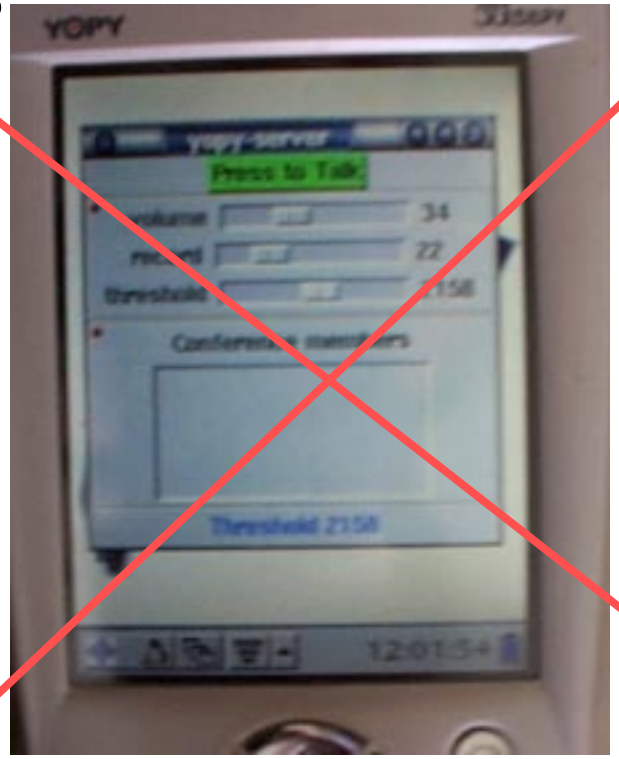
- Audio Conferencing
  - "CB radio" like interface
  - Interoperate with desktop clients
  - Provide "who's talking" feed back
- Lessons & Issues
  - Existing wireless infrastructure inadequate for real-time full duplex audio
  - PDA (and desktop) audio hardware is unpredictable and often inadequate



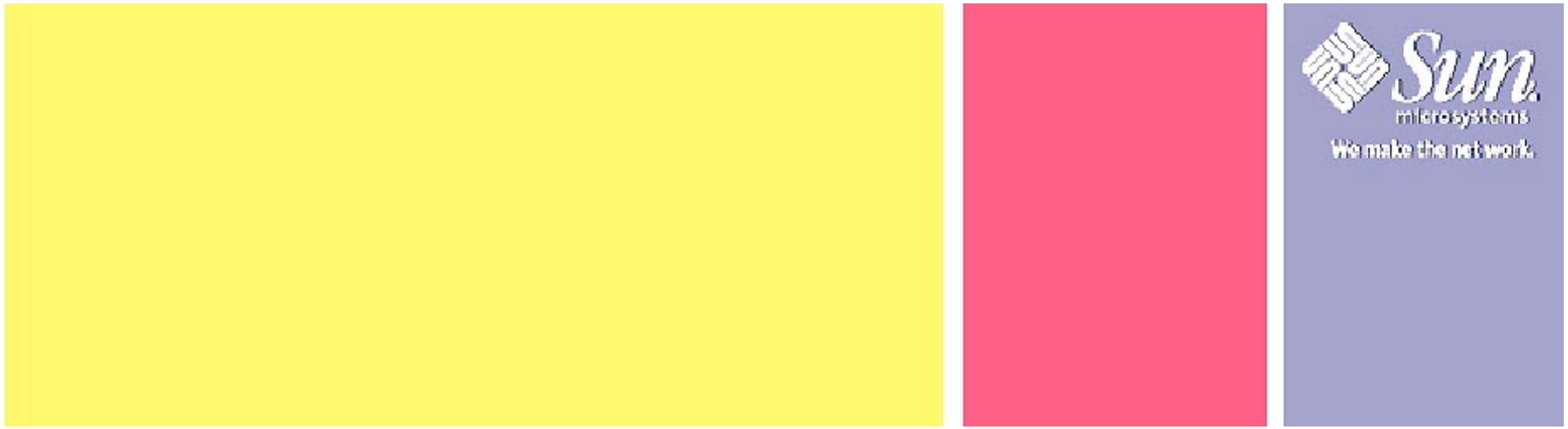


Sample Application Explorations

- Audio Instant Messaging (SLIM)
  - Integrate audio into an Instant Messaging framework
  - Half duplex “almost” real-time
- Lessons:
  - By choosing a user model carefully, audio latency problems are mitigated



Univeral Client



Universal Client Overview

- Thesis
  - The web has revolutionized the way we use computer networks.
    - Changes to an application are done on the "server“: there is no need to update the software on each client.
    - A single "application" may access many different servers: no one-to-one mapping between clients and servers
  - The "universal" nature of the client has come at a price
    - 15+ years of GUI research has been tossed out
    - The client ↔ server event model has been stripped to the barest minimum (remember IBM3270 terminals?).

Universal Client Big Idea

- How can we preserve the benefits of the web services computing model, yet gain back some of the user interface richness we lost in the process?
- The Universal Client: a "next generation" browser
  - Replace **HTML** with **UCML**, a component-centric markup language instead of a presentation-centric one
  - Replace the page-at-a-time event model with a richer component based event model
  - Retain compatibility with the existing web services server infrastructure (**HTTP/XML**)

Benefits

- Compelling Java Client story
- Leverage existing Web Services infrastructure
- Lower network bandwidth requirements
- Better UI potential
  - Application applicability over a wider range of devices
  - Better local feedback on cranky networks
  - Richer user experience

Universal Client Research

- Investigate existing bits and pieces
- Learn (and borrow) from the best

XWT Droplets
XFORMS XUL Pocket Linux
XAML Remote-AWT
AUIML UIML I3ML

Client-server & Markup

JFresco
Biss/awt Koala
Sawt Gwt SubArctic
Kawt JAPI Mica
FreeBongo Epios
Zaval Dog.gui

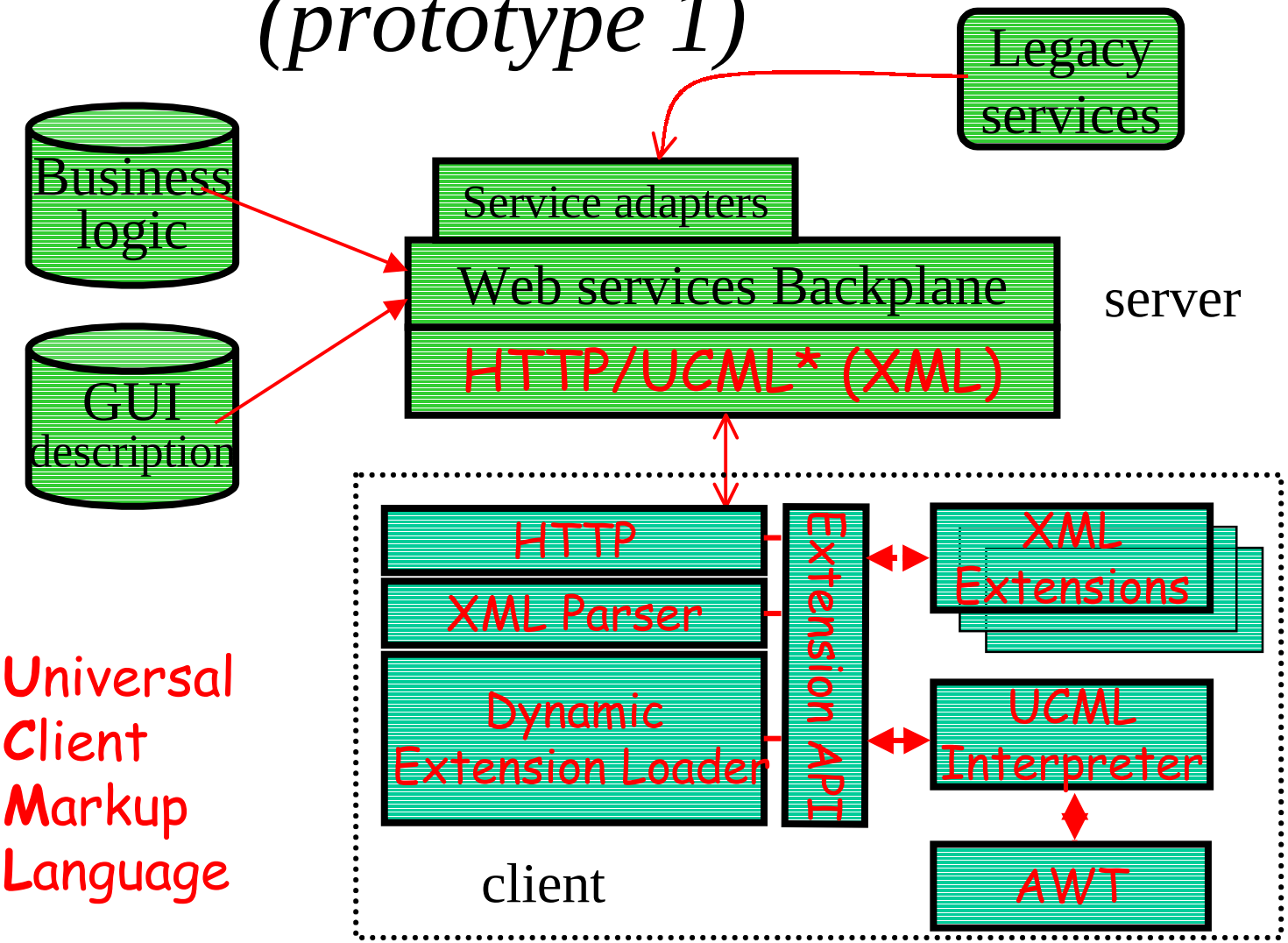
SDL/JSDL GTK+
DirectFB
Microwindows
TinyX

Java bindings

Graphics engines

Universal Client Architecture

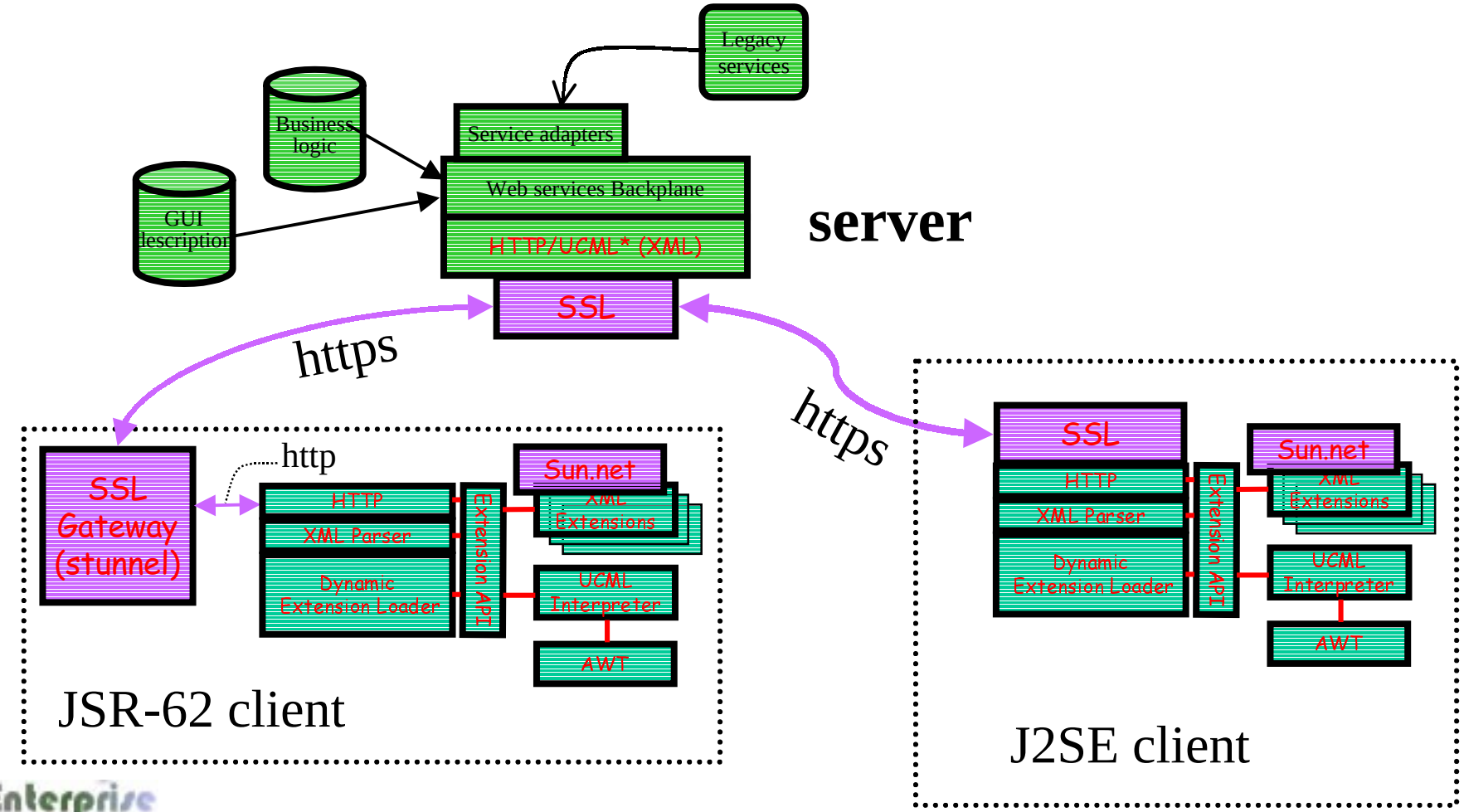
(prototype 1)



* Universal Client Markup Language

Universal Client Architecture

Security model



UCML – What is it?

- XML markup language that supports:
  - GUI component descriptions and layouts

```
<button text="quit" border="2" />
```

  - Event model definitions

```
<event component="quitButton" type="action">
  <configure name="status" text="bye now" />
  <send src="quit.ucml" />
</event>
```

  - A namespace mechanism

```
<namespace name="mine"> … </namespace>
```

  - A plugin extension architecture

```
<load class="uc.plugin.PointCloud"
      src="point.jar|cloud.jar" />
```

UCML: component descriptions and layouts

- Layouts look familiar to HTML developers
  `<table> … </table>`
- Components are aligned with the underlying implementation
  - Simplifies prototype development
  - Subject to change with experience
- Layouts and components are dynamic
  - Groups of components may be similarly configured with a single command
  - Component positioning may be changed at will

UCML Event model goals

- Provide local processing to increase responsiveness and minimize network bandwidth
- Keep markup simple, to empower *web site* developers
- Get the right balance

Namespace Management

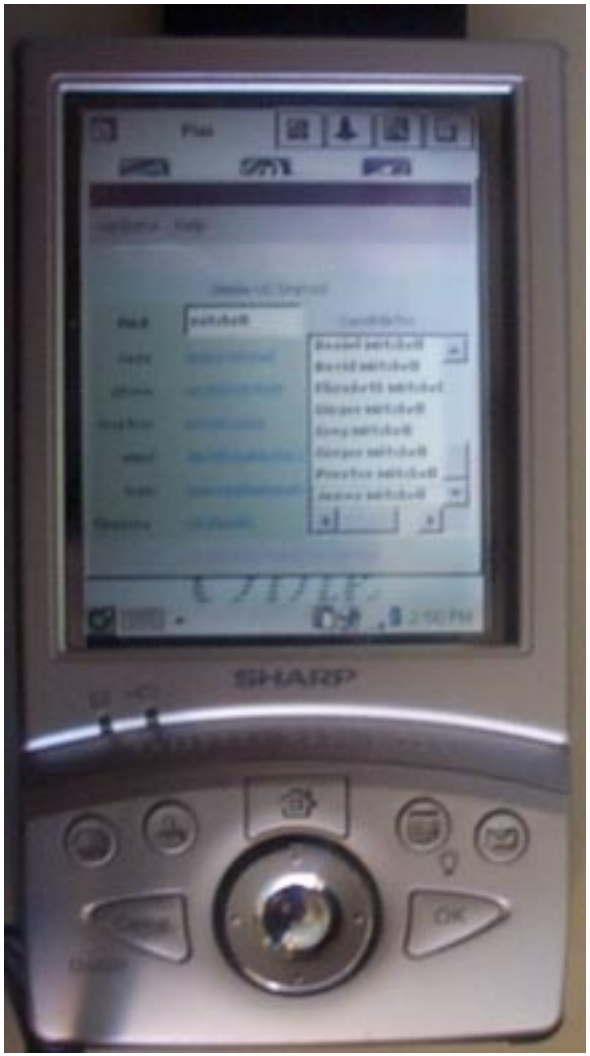
- Multiple applications need to run simultaneously
- Applications are composed with separately developed parts
- Parts need to interoperate without interfering

Dynamic extensions

- New functionality is integrated into the client
  - New components
  - Dynamic base class extensions
  - Additional flow control structures
- Extensions are java bytecodes that conform to the client extension API's
- Leverage Java security model
- Developers use new markup tags and attributes to access the extensions

UC1 Status

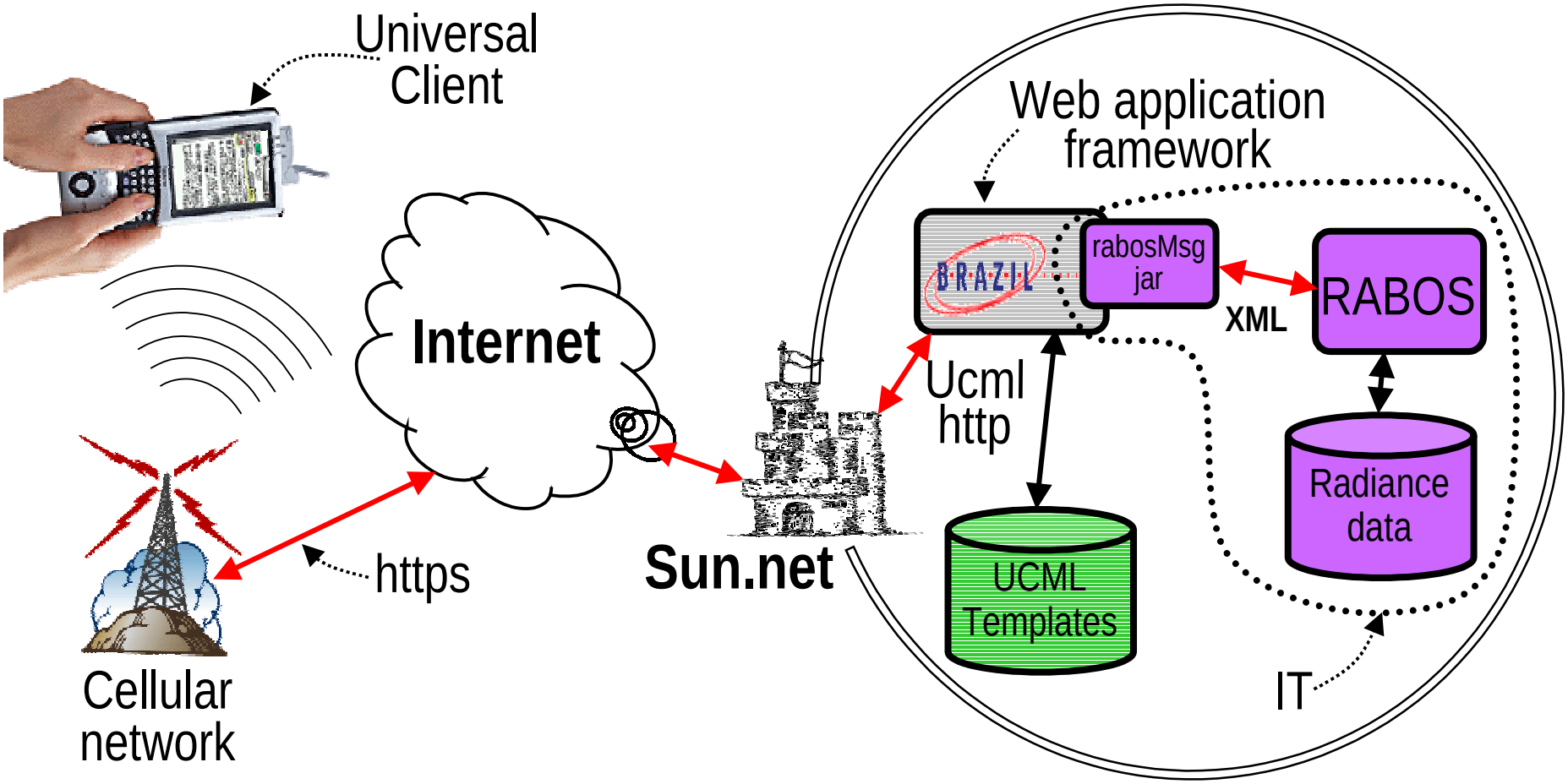
- Universal Client
  - 1st prototype running
  - End-to-end integration with web services infrastructure
  - Based on CDC Personal Profile
- Lessons
  - Promising initial results
  - JSR-62 provides an acceptable development base
  - New prototype implementation required to meet remaining goals



2004-0345

Enterprise PDA

Field Service Pilot Deployment Scenario



Field Services Pilot: key features

- End-to-end security (using ssl)
- XML (ucml) over HTTP
- Client conforms to JSR-62 (J2ME personal Profile)
- Swan access using existing mechanisms
  - Sun.net
  - ASN (access.sun.net)
  - VPN*

* If vpn client is available on the

Summary and Conclusions

- Valuable experience gained with custom apps
- Promising initial results (UC1)
- Positive reaction from customers
- Fertile and important area of research
- Lots of work left to do
- Make sure Java remains compelling in the vortex of phone/PDA convergence

Bibliography

- http://slack.sfbay:7777/
  - Mobile Computing “Proof-of-Concept”
- http://archivist.eng/docs/2003/450-0554/ucml.txt
  - UCML markup summary
- http://slack.sfbay:2004/org/index.ucml
  - Sample UC “orgtool” application
  - → View in browser to see UCML markup
  - → Run:

```
java –jar /net/slack.sfbay/export/home/demo/uc.jar \
 http://slack.sfbay:2004/org.ucml*
```

*Should mostly work using any modern Java vm