Instrumented Environments
Andreas Butz, butz@ifi.lmu.de, www.mimuc.de
SW Toolkits
Thematic map of SW infrastructures

AR = Augmented Reality
IE = Instrumented Environments
DMS = Distributed Multimedia Systems
World map of SW infrastructures
Timeline of SW infrastructures
Xerox ParcTab

http://sandbox.parc.xerox.com/parctab/

- Infrared network
  - Base stations in the ceiling
- Each base station was controlled by a IR gateway
- Each tab represented by a SW agent (tab agent)
- Applications written in
  - modula-3
  - Tcl/TK
  - Using MacTabit (~VNC)
BEACH (FhG IPSI Ambiente)

http://www.ipsi.fraunhofer.de/ambiente/
Timeline of SW infrastructures
Context Toolkit Framework

- Supports real-world model/methodology and provides library (distributed: XML/HTTP, input-focused)
- Component model: facilitates building of applications in Java
Timeline of SW infrastructures
AR Toolkit

- Library for Marker recognition
- Can be used for camera-based tracking
  - With head-mounted displays
  - With other screens
- C library
- Java wrapper available
- Works on
  - Windows
  - Linux
  - PDAs (WinCE)
Timeline of SW infrastructures
MIT aire + MetaGlue

http://aire.csail.mit.edu/ see video
Stanford Interactive Workspaces

http://iwork.stanford.edu/
Fluidum SW infrastructure

Event heap

Device manager
connect/disconnect/lookup service, resource allocation, data transfer

Users

Information units

Service

Device manager

Laptop
PC
PDA
Plasma display
Audio

Desk camera

Room camera

RFID

IrDA discovery

User modelling service

Object locator service

Plan recognition service

Event heap

Service

Service

Service

Desk camera

Device description

Video input

Picture input
Device Modeling
Universal Plug and Play (UPnP)

http://www.upnp.org/

Just send data over the network
  - (No executables)
  - Minimize version issues
  - Minimize security issues

Keep implementation private
  - Be agnostic re: programming language, OS
  - Update implementation w/o affecting interop
    - Improve performance
    - Reduce footprint
    - Improve capabilities

Agree on meaning / format of data
  - Choose substrate of proven protocols
  - Define device (service) specific protocols in a Forum
UPnP Tactics

- Start simple
  - Build in only universal things that everybody needs (and can live with)
  - Add as needed

- Minimize requirements
  - Basic IP network connectivity
  - Common HTTP protocol stack

- Leverage existing standards
  - HTTP, XML
Goals

- Describe the protocols for communication between
  - Control points
    - Controller, usually client
  - Device
    - Controlled, usually server
  - An actual device might contain both functions
Steps to UPnP Networking

0 Control point and device get addresses
1 Control point finds interesting device
2 Control point learns about device capabilities
3 Control point invokes actions on device
4 Control point listens to state changes of device
5 Control point controls device and/or views device status using HTML UI
UPnP protocol stack

<table>
<thead>
<tr>
<th>Protocol stack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UPnP vendor [purple-italic]</strong></td>
</tr>
<tr>
<td><strong>UPnP Forum [red-italic]</strong></td>
</tr>
<tr>
<td><strong>UPnP Device Architecture [green-bold]</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol Stack</th>
<th>Description</th>
<th>Protocol Stack</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>HTTP [black]</td>
<td>HTTP [black]</td>
</tr>
<tr>
<td>UDP [black]</td>
<td></td>
<td>TCP [black]</td>
<td></td>
</tr>
<tr>
<td>IP [black]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Jini

http://www.jini.org/
Jini, Java, RMI

- Jini is 100% Pure Java
- Write once, run anywhere
- Designed for building robust network apps
- built on the Java standard RMI

- Jini uses RMI for object-object communication
- Full object module support
- Pass any Java object and its code
- Works in any compliant JVM
- Easy to implement (i.e. automatic serialization)
- Provides foundation for addition of
  - multicast, replication and basic security

“Jini.. a networked federation of Java virtual machines”
## Jini Architecture

<table>
<thead>
<tr>
<th>Jini</th>
<th>Infrastructure</th>
<th>Programming Model</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Discovery</td>
<td>Lease</td>
<td>JavaSpaces</td>
</tr>
<tr>
<td></td>
<td>Lookup</td>
<td>Event</td>
<td>TX Manager</td>
</tr>
<tr>
<td></td>
<td>Extended Security</td>
<td>Transaction</td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>Java VM</td>
<td>Java API’s</td>
<td>Enterprise Beans</td>
</tr>
<tr>
<td></td>
<td>RMI</td>
<td>Beans</td>
<td>JNDI</td>
</tr>
<tr>
<td></td>
<td>Security</td>
<td>...</td>
<td>JTS</td>
</tr>
</tbody>
</table>

“Jini can be seen as an extension of the infrastructure, programming model, and services of Java”
Discovery

- Allows Jini services (both HW and SW) to:
  - Find and join a group of Jini devices
  - Advertise its capabilities
  - Provide any required SW and attributes
  - Works with JVM-enabled or non-JVM devices
    - Send out a multicast packet with reference to yourself
    - Receive a RMI reference to the Lookup service

“Discovery solves the problem of finding the place to start in an unknown network”
Discovery in Brief

1. The device advertises and looks for a service

2. Lookup Servers run instances of the discovery service which listen for multicast requests from discovering entities

3. The discovering entity performs a multicast that requests references to the lookup service.

4. The lookup server calls a remote method on the discovering entity’s exported object instance passing a remote reference to its lookup service as the parameter
Lookup

- Repository of available services
- Stores service as an extensible set of Java objects
  - ID, interface, GUI’s, attributes, drivers ...
- Service objects downloaded to user as required
- May be federated with other Lookup services
- Lookup service interface:
  - Registration, Access, Search, Removal

“The lookup service binds the federation together”
User Modeling
A very brief intro to user modeling

- A user model is:
  - Any kind of information about the user
  - Stored in one or several systems (distr. UM)
  - Used for adapting system output and/or behavior

- Example:
  - Recommendations by Amazon
A practical example for UM

Hallo, Dr. Andreas Butz! Hier sind Ihre persönlichen Empfehlungen.
(Wenn Sie nicht Dr. Andreas Butz sind, klicken Sie bitte hier.)

Dr. Andreas Butz, verdienen Sie EUR 1080,10. Jetzt verkaufen und Platz schaffen!
Acquisition of data for a UM

- Explicit
  - Type in your name, age, address, credit card
  - Adjust your preferences, skills, interests

- Implicit
  - Items purchased in the past
  - Money spent
  - Pages visited / items looked at?
  - Navigation speed ??
    - Automatic detection of web bots ;-)
Construction of a UM from data

- According to data collected, systems can
  - Store an individual profile of the user
  - Assign the user to a predefined stereotype
  - Find new stereotypes by clustering users
  - Make default assumptions for missing info
    - From global defaults
    - From stereotype
Adapting system behavior from UM

- Greeting customers by name
- Offer customers to sell their used stuff
- Filling in the correct credit card number
  - security issues, cookies,
  - Try signing on to amazon.co.uk with your account (email address & PW) from amazon.de
  - What parts of the UM do they have?

- Recommendations from domain models
  - Buyers of a DVD Player need DVDs

- Recommendations from „collaborative filtering“
  - Customers who bought X, also bought Y in the past
Ideas for UM in IEs

- Individual UM
  - Identity: detected for example from BT phone
  - Person’s name for communication
  - Person’s preferences for the room setup
  - Personal working environment
  - Messages for this person
  - User’s current plan/task/goal
  - Privacy settings

- Stereotype UM
  - Age group
  - Novice/expert
  - Technical/untechnical
  - Slow/fast typer
  - Tall/short person
  - ...???
Context Modeling
Context Toolkit
(with slides courtesy of Anind Dey)

- Anind K. Dey (prev. Intel, UCB, now CMU)
- [http://www.cs.cmu.edu/~anind/](http://www.cs.cmu.edu/~anind/)
- Toolkit to support Context-Aware applications
- Strong formalization of “context”
- Implementation in Java.
- Can be distributed on several machines in the environment
Context and Context-Awareness

- Focused on input

- Context: *any information relevant to an interaction that can be used to characterize the situation of an entity*

- Context-Awareness
  - General model of interactive computing
  - Addresses subset of ubicomp problems: input
Value of Context

- Potential for improved usability
  - Very important for mobile users with poor input devices

- “Smarter” applications

- Increased communications bandwidth
Design Space for Context-Aware Applications

- Toolkit allows exploration of design space
- Basic types of context:
  - Location, identity, time, activity
  - Simple/singular → complex/multiple
  - Combinations
- Uses of context:
  - Present to user
  - Automatically perform set of services
  - Tag captured information to ease retrieval
Example

- Tour guides, travel assistants, personalization software

- Reminder to buy milk
  - When to deliver: not time/location specific
  - How to deliver: appropriate modality
Building Applications

- M. Weiser: The whole point of ubiquitous computing, of course, is the applications.

- But ... what if the applications are hard to build? And, what if this inhibits our ability to build compelling applications?
Why Context is Hard to Use

- Acquired from sensors
  - Not just keyboards and mice – lots of heterogeneous devices

- Need to abstract data

- Distributed

- Dynamic
Results of Difficulties

- *Ad hoc* application building
  - Difficult to build, reuse and evolve
- Small variety of sensors
- Small variety of context: mostly *location*
- Few applications, mostly simple: mostly *presenting context*

- Practical: difficult to prototype, test and evaluate
Context Toolkit: Research Contributions

- Conceptual framework requirements
  - Provide framework for designing apps more easily
  - Lower threshold to enable more designers
- Context Toolkit itself
  - Implementation and exploration of design space
- Support investigation of complex problems and more realistic apps
  - Raise ceiling
  - Privacy, uncertainty, security, end-user programming
Toolkit Requirements

- Context specification
- Discovery
- Separation of concerns
- Storage
- Constant availability
- Transparent communications
- Interpretation
Look to input handling

- Graphical User Interface (GUI) widgets
  - separation of concerns
  - callbacks and attributes
  - query/subscribe
  - common interface

- e.g. button
Context Widgets

- Responsible for acquiring and abstracting data from particular sensor, separation of concerns, storage
Context Interpreters

- Convert or interpret context to higher level information
- Context not available at appropriate level
Context Aggregators

- Collect context relevant to particular entities (recall definition)
- Further separation, simplifies design

Diagram:

- In/Out Board
- Building Aggregator
  - Location Widget
  - Location Widget
  - ID to Name Interpreter
  - Face Recognition
  - Smart Card Reader
Context Toolkit Framework

- Supports real-world model/methodology and provides library (distributed: XML/HTTP, input-focused)
- Component model: facilitates building of applications
Experiences: Benefits

- Provides separation of concerns

- Lightweight integration and re-use of components

- Easy to create and evolve apps, allowing exploration of the design space
  - Add context to context-less apps
  - Add more context to context-aware apps
Aware Home (MANSE ’99)

- Great testbed for context-aware computing
- 3 goals: elderly, infants, everyone
- Context Toolkit is the s/w infrastructure in the Aware Home
Applications Built

- Simple use of location:
  - Turn lights on and off (perform service)

- Location and id (perform service)
  - Information Guide: present info about user’s group (CHI ’99)
  - Context-Aware Mailing List
In/Out Board – 3 versions *(CHI ’99)*

- Context used: location, identity, time
- How used: present context
In/Out Board Architecture

• Simple app demonstrates support for **reusability** (don’t have to re-build infrastructure on per-application basis) and **evolving** applications
Serendipitous Meetings

• Context used: location, id, time, activity
• How used: present, perform service, tag
• record and tag drawings and audio for later retrieval
Ink written *before* current time is in original color

Ink written *after* current time is in lighter color

Current time within session

Selected session

Selected day

Day containing whiteboard activity
Meeting Architecture

For each possible location of the mobile board:

- Location Widget
  - iButton Dock
- Location Widget
  - iButton Dock
- ID to Name Interpreter

Context Architecture
Conference Assistant (ISWC '99)

- Context used: location, multiple levels of identity, activity, time
- How used: present, service, tag

Schedule

Retrieved slide

Slide text

Query Interface

User notes

Identity, Location, Activity of People, Places, Things

context widgets
Conference Assistant Arch.

For each presentation space:

- Conference Assistant
- User Aggregator
- Recommend Interpreter
- Record Widget
- Presentation Aggregator
- Content Widget
- Location Widget
- Question Widget
- Software
- iButton Dock
- Memo Widget
- Software
- For each user/colleague
- Registration Widget
- GUI
- Context Architecture