Adapting Mobile Systems Using Logical Mobility Primitives

Stefanos Zachariadis
Joint Work With Cecilia Mascolo and Wolfgang Emmerich
Software Systems Engineering & Mobile Systems Interest Groups
Department of Computer Science
University College London

http://www.cs.ucl.ac.uk/staff/s.zachariadis
Outline

• Background
• Logical Mobility
• Component Model
• Middleware System
• Implementation
• Related Work
• Future Work
• Conclusion
Trends in (Mobile) Computing (Hardware)

• They are getting faster
• They are getting connected
• They are getting smaller
• They are getting everywhere
Trends in (Mobile) Computing (Software)

• Not much innovation
• Monolithic apps
• Lack of middleware
• Static apps
Trends in (Mobile) Computing (Example)

1997: US Robotics Pilot 1000
- 128KB 16MHz Serial
- 160x160BW

2003: Palm Tungsten T3
- 64MB 400MHz
- Serial/USB/Bluetooth/Infrared
- 320x480 24bit, Sound, Expansion
Trends in (Mobile) Computing
(Example)

1997:
US Robotics Pilot 1000
PalmOS 1.0 (DateBook)

2003:
Palm Tungsten T3
PalmOS 5.2 (Calendar)

Black Box -> Market Saturation
The Mobile Environment

• Limitations (compared to traditional computing)
  – Memory, battery power, CPU power, erratic (expensive) connectivity
  – Improving but lagging still

• Different usage paradigms
  – Input/output
  – Speed, ease of use, frequent but brief usage
    • E.g. Check schedule
  – People don't install 3rd party applications
  – Applications need to cater to users’ needs throughout the device’s lifetime

• Ubiquitous Computing -> Dynamic Environment

• The need for dynamic change
Adaptation

- Change to accommodate changes to its requirements
  - Informal: Adaptation is the process by which a system can dynamically acquire or drop functionality.
- Suitability for mobility
- Architecture & Means for Adaptation
  - Not Decision
- How to adapt?
- How to engineer an adaptable system?
Logical Mobility

• Ability to send parts of an application (or migrate/clone a process) to another host
• Popularised by Java
• Classification into paradigms
• Encapsulate Functionality
• Numerous examples
  – Active networking, resource exploitation...
  – Need for systematic and flexible use of all paradigms
  – Send & receive
Platform for Logical Mobility

Classifier -> Instance -> LogicalMobilityEntity

Class

DataType

Handler

LogicalMobilityUnit

getAttribute(k: DataType) : Attribute

properties

0..*

Attribute

key : DataType
value : DataType
getValue() : DataType

MutableAttribute

setValue(value: DataType) : void
Platform for Logical Mobility (2)

• Modeled as Concurrent Processes (FSP)

<table>
<thead>
<tr>
<th>Application</th>
<th>Communications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Controller</td>
</tr>
<tr>
<td>Trust &amp; Security</td>
<td>Sendor/Receiver</td>
</tr>
<tr>
<td>Serialisation/Deserialisation Engine</td>
<td></td>
</tr>
</tbody>
</table>

• Can be used to implement any paradigm
Components

- Component = functionality
- Coarse-grained adaptation guide
- Monolithism vs Componentisation
SATIN

• System Adaptation Targeting Integrated Networks
• Component Meta Model & Middleware
• Low Footprint
• Interaction & Autonomy
Component Model Outline

• Local Component Model
• Late - Binding
• Logical Mobility as a first class citizen
  – by encapsulating and offering the platform
• Everything is a component
Component Model Outline (2)
Components

• Encapsulation of functionality
• Facets
• Properties & Attributes
  – Extensible
  – Heterogeneity (Debian)
  – Request template
  – Identifier, Versioning, Dependencies
Container

• Component Specialisation
• Registry/host of components
  – References to all components
• One on each instance
• Dynamic Registration/Removal (delegated)
  – Registrars can have different policies
• Listeners/Custom Notification
Distribution

• Use LM platform defined before

• Logical Mobility Entity (LME)
  – Generalisation of class, object, data and component

• Application is a Reflective Component
Reflective Components

• Component Specialisation
• Components that can be changed
  – LMU Recipients
  – The Container is Reflective
  – Inspect LMUs
    • Acceptance
    • Rejection
    • Partial Acceptance
    • Handler Instantiation
Deployer

• Component Specialisation
• At least one in each instance
• Abstracting sending/receiving/requesting LMUs
• Uses attributes for matching
• Synchronous and Asynchronous primitives
• Can be used to implement all paradigms
Middleware

- Component Based
  - “Equal” Components

- Advertising & Discovery
  - Advertisable Components
    - Advertising message
  - Advertiser Components
    - Register Advertisable Components
  - Discovery Components
    - Listeners / Notification
Middleware (2)
Example Application: Dynamic Launcher

• Similar in Functionality to PDA Launchers

• Installs Components from multiple sources
  – Centralised Source, p2p...
  – Uses any discovery components installed to find components available
  – Uses Deployer to request and receive components

• Transparent update
  – Using any Discovery components installed and Deployer to find and install updates
Dynamic Launcher [2]
Dynamic Launcher [3]
Example Application: Music Player
Example Application: Scripting Framework

- Initialising the Container -
- Container (ID=STN:CONTAINER,FACETS=Discovery,VER=1)
  initialised -
- Creating Self -
- Registering Self (ID=STN:SHELL) -
- This is SATIN version 0.8 -
- Running on Linux 2.6.5-1.358 / i386 -
- Hostname: hamsalad.cs.ucl.ac.uk -
- Java 1.4.2_04 / Sun Microsystems Inc. -
- A reference to the container will be made available via the object reference container -
- Starting the beanshell... -

BeanShell 2.0b1.1 - by Pat Niemeyer (pat@pat.net)

bsh % Component c=container.getComponent(``STN:SHELL``);
Some Numbers

- J2ME cdc personal profile
- 84KB jar
- Dynamic Launcher
  - 22KB jar
  - Startup Time on PDA: 21 seconds
  - Memory Usage on PDA: 1155KB
  - Update to PDA from peer: 2063 ms
- Music Player
  - 3.6KB jar application
  - 105KB jar codec
- SATIN Scripting Framework
  - 280.6KB jar
Related Work

• Logical Mobility Middleware
  – Limited Use of LM
    • Too Specific (Lime, PeerWare, Jini, XMIDDLE)
  – Not geared for mobility
    • Disconnections pre-announced (Fargo-DA)
    • Fixed advertising and discovery (one.world)
Related Work (2)

• Component Model Systems
  – Distributed ones unsuitable
    • Large
    • No autonomy (P2PComp, PCOM)
  – Local Component Models
    • Heterogeneity
    • Some make a distinction between Component providers and consumers (Beanome/OSGi)
Future Work

  - EU Project for pervasive computing security
  - Demo @ IST 2004

  - EU Project for middleware for ubiquitous computing

- **Q-CAD**
  - QoS-aware resource discovery framework
  - Joint work with Licia Capra

- Open source!
Conclusion

• Platform for Logical Mobility
• The SATIN Component model
  – Distribution as a service
  – Attributes for description
  – Applications & System: interconnected local components
  – Reconfiguration of Local Components
• The SATIN Middleware System
  – Componentised Middleware (Advertising and Discovery)
  – Logical Mobility as a Computational Primitive
Any Questions?

Publications and more information at
http://www.cs.ucl.ac.uk/staff/s.zachariadis

Thank you!