Mobile Self Organisation through the SATIN Component Model

Stefanos Zachariadis

supervised by Dr. Cecilia Mascolo and

Dr. Wolfgang Emmerich

Software Systems Engineering Group

Department of Computer Science

University College London

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

Outline

- Background
- 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:

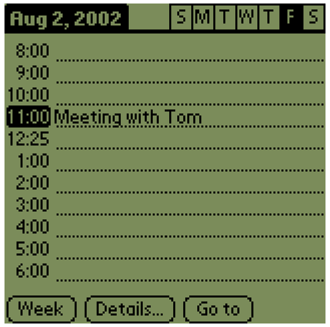
Palm Tungsten T3

64MB 400MHz
Serial/USB/Bluetooth/Infrared
320x480 24bit, Sound, Expansion

Trends in (Mobile) Computing (Example)

1997:

US Robotics Pilot 1000



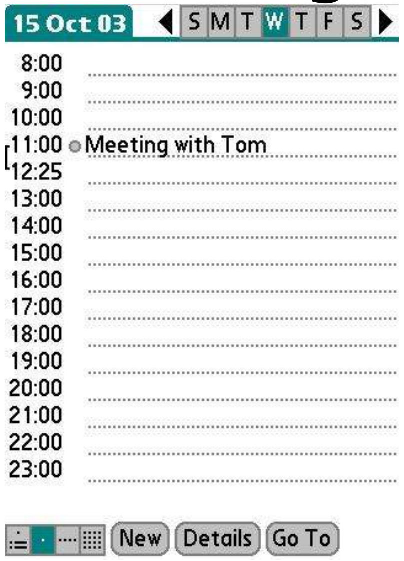
Aug 2, 2002 | S M T W T F S

8:00
9:00
10:00
11:00 Meeting with Tom
12:25
1:00
2:00
3:00
4:00
5:00
6:00

PalmOS 1.0 (DateBook)

2003:

Palm Tungsten T3



15 Oct 03 | S M T W T F S

8:00
9:00
10:00
11:00 Meeting with Tom
12:25
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00

PalmOS 5.2 (Calendar)

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
  - Reports show that users rarely install applications on mobile devices
    - Applications need to cater to users' needs throughout the device's lifetime

A Dynamic Environment

- Heterogeneity!
  - Device/Hardware (Physical)
  - Software (Logical)
  - Network
- Changes to the environment
  - => Changes to application requirements.

Self - Organisation

- System adaptation to accommodate changes to its requirements
- Suitability for mobility
- Allows systems to gain new functionality
  - Reacting to changes
- Approaches
  - Expert Systems
  - Genetic Algorithms

Logical Mobility

- Ability to sent parts of an application (or migrate/clone a process) to another host
- Popularised by Java
- Classification into paradigms
  - Client/Server (CS)
  - Remote Evaluation (REV)
  - Code on Demand (COD)
  - Mobile Agents (MA)
- Various middleware (mobile & stationary) systems exploit this

Components

- Component = functionality
- Coarse-grained guide
- Monolithism vs Componentisation
- Collocation vs Distribution
  - Complexity
  - Size
  - Networking
  - Autonomy

SATIN

- System Adaptation Targeting Integrated Networks
- Component Model & Middleware
- Minimal Footprint
- Interaction & Autonomy

Component Model Outline

- Local Component Model
- Distribution Built into the Model
  - But not components
  - Using Logical Mobility
- Applications and the system itself are components

Components

- Encapsulation of functionality
- Facets
- Properties & Attributes
  - Extensible
  - Heterogeneity (Debian)
  - Identifier, Versioning, Dependencies
    - <ID, "identifier">
    - <VER, version number>
    - <DEP, 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

- Logical Mobility Entity (LME)
  - Generalisation of class, object, data, component
- Logical Mobility Unit (LMU)
  - Composition of LMEs
  - Attributes & Properties
  - Handler
  - Fine grained mobility

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
  - Advertised
- Abstracting sending/receiving/requesting LMUs
- Uses attributes for matching
- Synchronous and Asynchronous primitives
  - Operation Formalised using Mobile Unity

Middleware

- Fully componentised
- Advertising & Discovery
  - Advertisable Components
    - Advertising message
  - Advertiser Components
    - Register Advertisable Components
  - Discovery Components
    - Listeners / Notification

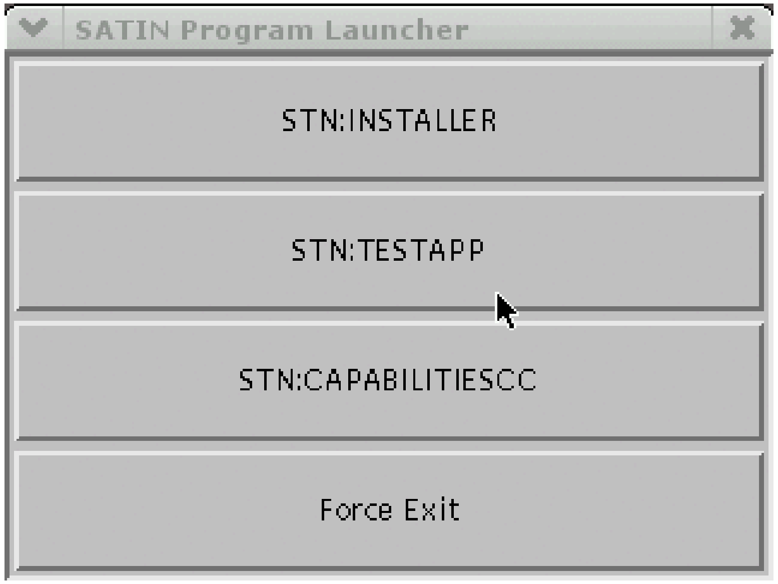
Logical Mobility

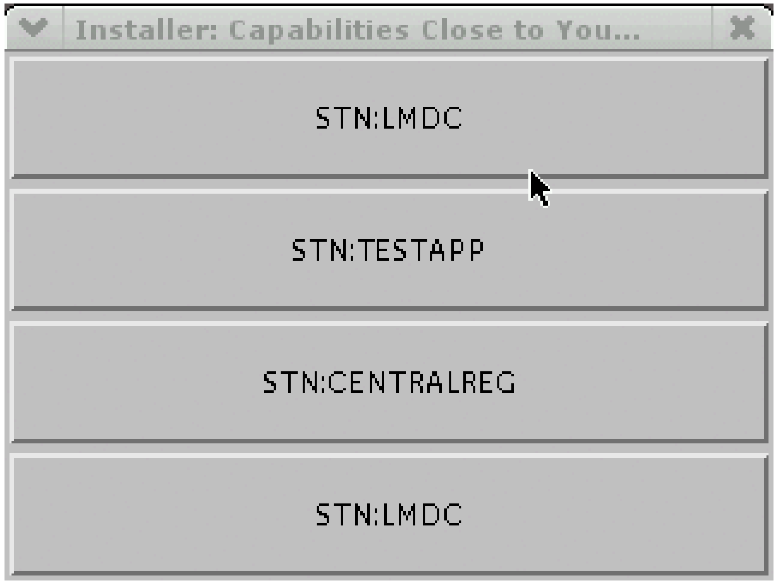
- Finer Grained
- Not only Components, but Classes/Objects
  - Patching
- Logical Mobility as a computational paradigm

Example Application: Dynamic Launcher

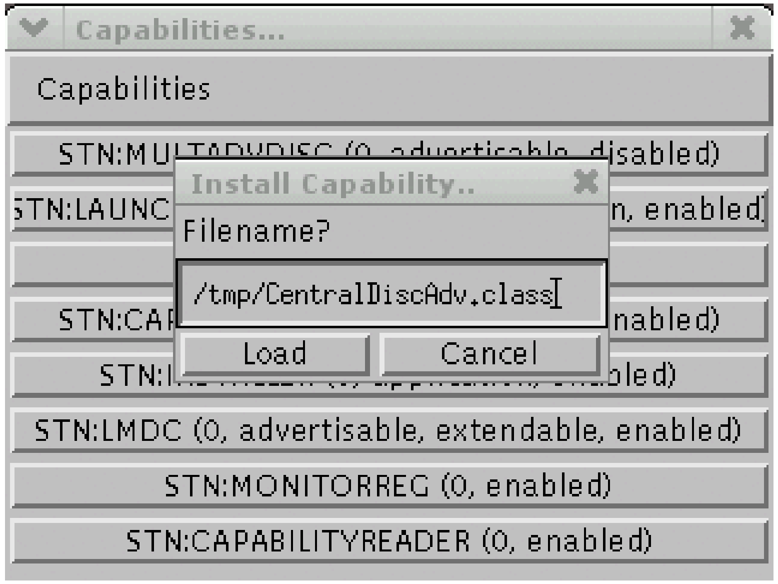
- Similar in Functionality to PDA Launchers
- Installs Components from multiple sources
  - Centralised Source, p2p...
  - Uses any discovery components installed to find components available
  - Uses Deployer to request and receive components
- Transparent update
  - Using any Discovery components installed and Deployer to find and install updates

Dynamic Launcher [2]





Dynamic Launcher [3]



Capabilities...

Capabilities

STN:MU ... disabled)

STN:LAUNC ... n, enabled)

STN:CAP ... nabled)

STN:I ... bled)

STN:LMDC (0, advertisable, extendable, enabled)

STN:MONITORREG (0, enabled)

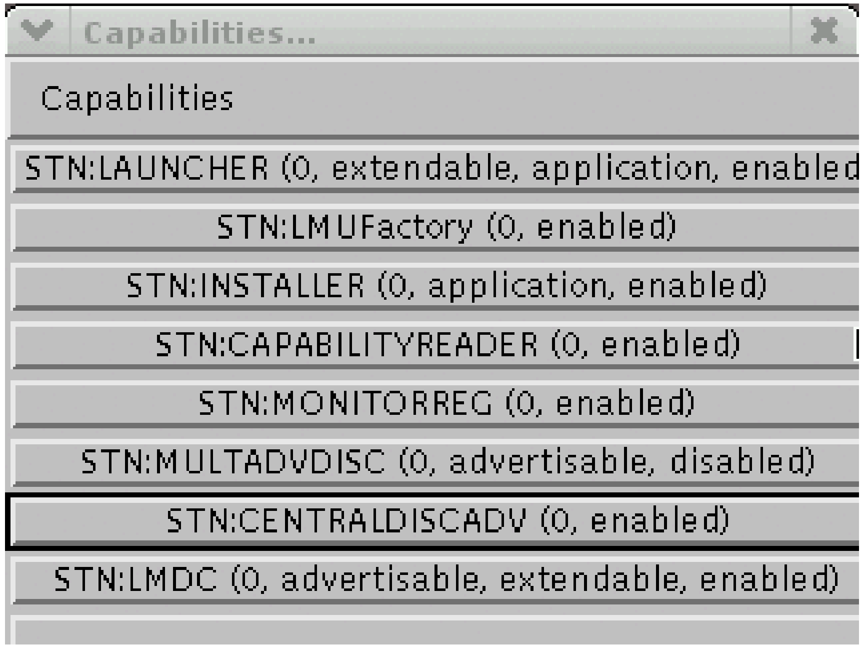
STN:CAPABILITYREADER (0, enabled)

Install Capability..

Filename?

```
/tmp/CentralDiscAdv.class
```

Load | Cancel



Capabilities...

Capabilities

STN:LAUNCHER (0, extendable, application, enabled

STN:LMUFactory (0, enabled)

STN:INSTALLER (0, application, enabled)

STN:CAPABILITYREADER (0, enabled)

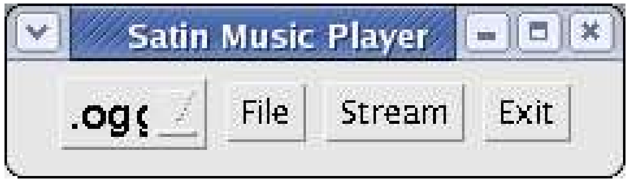
STN:MONITORREG (0, enabled)

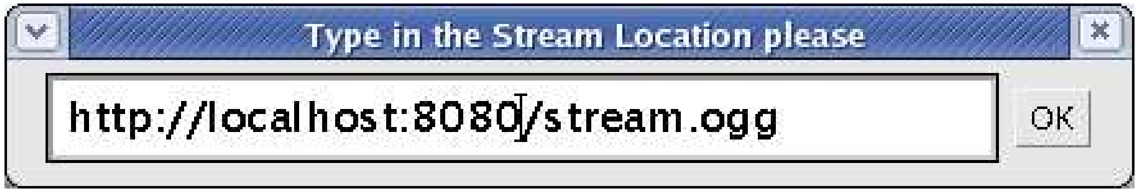
STN:MULTADVDISC (0, advertisable, disabled)

STN:CENTRALDISCADV (0, enabled)

STN:LMDC (0, advertisable, extendable, enabled)

Example Application: Music Player





http://localhost:8080/stream.ogg

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
    - System Reconfiguration (UIC, ReMMoC)
    - 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

- Applications
- Integration with CARISMA
- PhD Thesis ?

Conclusion

- The SATIN Component model
  - Distribution as a service
  - Attributes for heterogeneity
  - Applications & System: interconnected local components
  - Reconfiguration of Local Components
- The SATIN Middleware System
  - Componentised Middleware (Advertising and Discovery)
  - Logical Mobility as a Computational Primitive
- Security?