AmbientTalk
A scripting language for mobile phones

Tom Van Cutsem

Software Languages Lab
Brussels, Belgium
Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication
Mobile Ad Hoc Networks

Networks of mobile devices that use wireless p2p communication
Mobile Ad Hoc Networks

Networks of mobile devices that use wireless p2p communication

Zero Infrastructure

Volatile Connections
Mobile Ad Hoc Networks

Networks of mobile devices that use wireless p2p communication

Zero Infrastructure

Volatile Connections

Local interactions
AmbientTalk: fact sheet

- Object-oriented, functional patterns, dynamically typed
- Actor-based concurrency/distribution
- Mirror-based reflection
- JVM as platform
- Runs on Android and J2ME/CDC phones
Four Decades of Language Research

Scheme 1975

Smalltalk ‘70s-‘80s

Self 1986

Scheme 1975

Hewitt’s actors (PLASMA) ‘73-‘76

ABCL 1986

1986

1997

AmbientTalk 2006
Four Decades of Language Research

Smalltalk ‘70s–’80s

Hewitt’s actors (PLASMA) ‘73–’76

Scheme 1975

ABCL 1986

Self 1986

1997

AmbientTalk 2006
Four Decades of Language Research

- Smalltalk '70s-'80s
- Scheme 1975
- Hewitt’s actors (PLASMA) ‘73-’76
- ABCL 1986
- Self 1986
- 1997
- AmbientTalk 2006
Four Decades of Language Research

Smalltalk ‘70s-‘80s

1986

Scheme 1975

1997

Hewitt’s actors (PLASMA) ‘73-‘76

ABCL 1986

AmbientTalk 2006
Four Decades of Language Research

- Smalltalk (‘70s-‘80s)
- Scheme (1975)
- Hewitt’s actors (PLASMA) (‘73-‘76)
- ABCL (1986)
- Self (1986)
- 1986
- 1997
- AmbientTalk (2006)
Four Decades of Language Research

1975

Hewitt’s actors (PLASMA) ‘73-’76

1986

ABCL

Smalltalk ‘70s-‘80s

1986

Self

1997

AmbientTalk 2006

1986

Scheme 1975
Four Decades of Language Research
Four Decades of Language Research

- Scheme 1975
- Smalltalk '70s-'80s
- Hewitt’s actors (PLASMA) ‘73-'76
- ABCL 1986
- Self 1986
- AmbientTalk 2006
- 1997
Four Decades of Language Research

Smalltalk ‘70s–‘80s

Scheme 1975

Hewitt’s actors (PLASMA) ‘73–‘76

ABCL 1986

Self 1986

AmbientTalk 2006
Four Decades of Language Research

- **Hewitt’s actors (PLASMA) ‘73-’76**
- **Scheme 1975**
- **Smalltalk ‘70s-‘80s**
- **ABCL 1986**
- **Self 1986**
- **AmbientTalk 2006**
Four Decades of Language Research

- Smalltalk ‘70s-'80s
- Scheme 1975
- Hewitt’s actors (PLASMA) ‘73-'76
- ABCL 1986
- Self 1986
- AmbientTalk 2006
Four Decades of Language Research

Smalltalk ‘70s-‘80s

Scheme 1975

Hewitt’s actors (PLASMA) ‘73-’76

ABCL 1986

Self 1986

1997

AmbientTalk 2006
Four Decades of Language Research

Hewitt’s actors (PLASMA) ‘73-’76

Scheme 1975

Smalltalk ‘70s-‘80s

ABCL 1986

Self 1986

AmbientTalk 2006
Four Decades of Language Research

- Smalltalk ('70s-'80s)
- Scheme (1975)
- Hewitt’s actors (PLASMA) ('73-'76)
- ABCL (1986)
- Self (1986)
- ECMAScript Harmony (2006)

Connections:
- Smalltalk influences Self
- Scheme influences Self
- ABCL influences Self
- Self influences ECMAScript Harmony
- Hewitt’s actors (PLASMA) influences ABCL
How does AmbientTalk help?

**Volatile Connections**
- Asynchronous, buffered messaging
  - send messages, even when disconnected
- No blocking synchronization
  - receive events, even when disconnected
- Network failures ≠ exceptions
  - timeouts & leasing, whether connected or disconnected

**Zero Infrastructure**
- Peer-to-peer service discovery protocol
  - decentralized, location-based
Event Loop Concurrency

- AmbientTalk programs are event loops
- They react to events from the outside world
- Inter-event loop communication is asynchronous
Event Loop Concurrency

- AmbientTalk programs are event loops
- They react to events from the outside world
- Inter-event loop communication is asynchronous
def service := object: {
    def echo(text) {
        system.println("Received: "+text);
        text
    }
}
deftype EchoService;
def pub := export: service as: EchoService;

deftype EchoService;
def echoF := when: EchoService discovered: { |echoSvc|
    system.println("Discovered an echo service");
    echoSvc;
} within: 2.minutes

echoF<-echo("test1");
def resultF := echoF<-echo("test2")@TwoWay;
when: resultF becomes: { |value|
    system.println("Reply: "+ value);
}
echoF<-echo("test3");
AmbientTalk = OO + Events

Generate and receive application requests

\[ \text{obj} \leftarrow \text{msg} (\text{arg}) \]
\[ \text{def } \text{msg} (\text{param}) \{ \ldots \} \]

Follow-up on outstanding requests

\[ \text{when: } \text{future becomes: } \{ \mid \text{result} \mid \ldots \} \]

React to services appearing and disappearing

\[ \text{when: type discovered: } \{ \mid \text{ref} \mid \ldots \} \]

React to references disconnecting, reconnecting, expiring

\[ \text{when: ref disconnected: } \{ \ldots \} \]
\[ \text{when: ref reconnected: } \{ \ldots \} \]
\[ \text{when: ref expired: } \{ \ldots \} \]
Urbiflock

• P2P Geosocial networking framework

• Test deployment on Brussels public transport network
Summary

ad hoc   Zero Infrastructure   Volatile Connections
Volatile Connections

Zero Infrastructure

Android 1.6

J2SE 1.5

J2ME CDC

ad hoc

Zero Infrastructure

Volatile Connections