

one.world

Programming for Pervasive Computing Environments

Robert Grimm, Janet Davis, Eric Lemar,
Adam MacBeth, Steven Swanson,
Tom Anderson, Brian Bershad, Gaetano
Borriello, Steven Gribble, David Wetherall

<http://one.cs.washington.edu>

Pervasive Computing

- **Ubiquitous smart devices**
 - Deployed in working and living spaces
 - Some mobile, some stationary
- **Powerful network services**
 - Shared within the infrastructure
- **Devices and services are context aware and coordinate with each other**
 - Provide immediate and universal access to information
 - Support users in completing their tasks

Problem

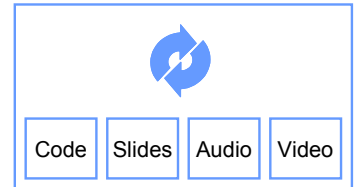
- **Giant, ad-hoc distributed system**
 - Tens of thousands of devices and services come and go
- **Yet, applications need to work continuously**
 - Adapt to a changing environment
 - Run even if
 - Devices are roaming
 - Users switch devices
 - Network provides only limited services, or none at all

Approach

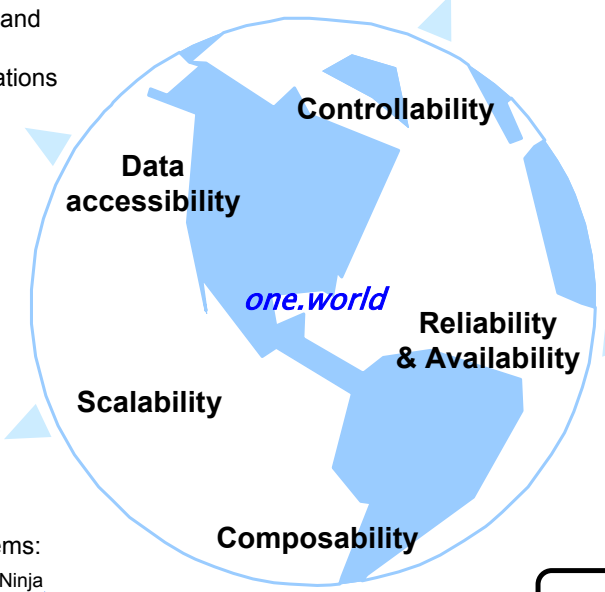
- **Alter way programmers think about applications**
 - Programming for change in
 - Remote resources
 - Local resources
 - Application data
- **Provide system support**
 - Integrated architecture
 - Designed to support pervasive applications
 - Provides powerful primitives to help programmers

Features

- **Structured I/O**
 - *Preserve structure of application data*
 - Data represented as tuples
 - Strongly typed records
 - Common interface to storage and communications
 - Atomic read and write operations
 - Storage only
 - Transactions
 - Searches
- **Asynchronous events**
 - *Scale better than threads*
 - Make execution state explicit
 - Provide control over scheduling
 - Have been used successfully across a wide range of other systems:
 - ← Tiny OS Palm OS Chinook Windows Ninja →
- **Encapsulation**
 - *Control storage and computations*
 - Hierarchical environments
 - Containers for stored tuples, active computations, and other environments
 - Provide isolation and resource controls
- **Integration of mobile code with storage**
 - *Mobile code needs local storage*
 - Code, like data, stored in environments
 - Computations can be check-pointed
 - Environments can be moved between nodes



An example environment hierarchy: A presentation application running in the outer environment; code, slides, audio, & video stored in nested environments.



```
public interface EventHandler {
    void handle(Event e);
}
```

The event handler interface

- **Dynamic linking and late binding**
 - *Easier to exchange data than to compose traditional interfaces*
 - Uniform event handling interface
 - Components
 - Export and import event handlers
 - Dynamically linked and unlinked
 - Discovery
 - Send event to matching resource

Status

- **Core implemented in Java**
 - Uses Berkeley DB for tuple storage
 - Missing: transactions, class loading from environments, and resource control
- **Exploring applications**
 - digime – PIM on steroids
- **Conducting an experiment: 490dp**
 - Projects compare Java & *one.world*
- **Made source release (v0.4)**