On the Design Framework of Context Aware Embedded System

Xian-He Sun
With Abhay Daftari, Nehal Mehta, Shubhanan Bakre
Illinois Institute of Technology
Request

• Position, View Point
• Software Engineering for Embedded System
• Requirement and Implementation

Outline

• From Single device to Coordinated Smart Space
• Context Awareness: A New Challenge
• Aspect Oriented Software Design
• Scarlet – The IIT Context Aware Infrastructure
• Conclusion and Open Questions
Embedded Systems: What is the new

- Devices become smaller and more powerful
- What is the new challenge?
- From “autonomous computing” to coordinated “human-center computing”
Coordinated Embedded System – Smart Space

• Modern Warships
• What is the relation with pervasive computing?
Pervasive Computing

- Computers have become an embed intrinsic part of a sophisticated, networked, pervasive and ubiquitous computing environments around humans.
- **Pervasive Computing**: create a ubiquitous environment that combines processors and sensors with network technologies (wireless and otherwise) and intelligent software to create an immerse environment to improve life.
- Is that ubiquitous environment a ubiquitous, coordinated embedded system?
Pervasive Computing

MIT’s view of pervasive computing
Evolution of Pervasive Computing

Remote communication
- protocol layering, RPC, end-to-end args...

Fault tolerance
- ACID, two-phase commit, nested transactions...

High Availability
- replication, rollback recovery...

Remote information access
- dist. file systems, dist. databases, caching...

Distributed security
- encryption, mutual authentication...

Distributed Systems

Mobile networking
- Mobile IP, ad hoc networks, wireless TCP fixes...

Mobile information access
- disconnected operation, weak consistency...

Adaptive applications
- proxies, transcoding, agility...

Energy-aware systems
- goal-directed adaptation, disk spin-down...

Location sensitivity
- GPS, WaveLan triangulation, context-awareness...

Smart spaces
- Invisibility
- Localized scalability
- Uneven conditioning

By Satyanarayanan
Context Awareness is a Challenge

Effective Use of Smart Spaces

Localized Scalability

Human Centered Services

Invisibility

Masking Uneven Conditioning

Coordinated Embedded Systems
Context Aware Embedded System

• **Context**
  – Useful information other than user input

• **Context Awareness**
  – Ability to *capture, understand and adapt* to surrounding context information

• **Context aware embedded system**
  – Capture context information via ‘*embedded*’ devices
    Takes action without explicit user input
  – Improves user experience by achieving collaboration and integration of embedded systems
Role of Context

Traditional Class Environment

- Professor T informs students about the updated course website for lecture slides
- They need to bring the slides in the class for better understanding
- Some of the students either did not read the notification
- Some of them forgot about it before the class

Smart Class Environment

- If
  - Professor T is moving towards the projector and
  - lights in the room are off
- Then the environment pervasively transfers the presentation slides from the professor’s handheld device to students’ handheld device
- The projector starts the presentation
Mobile Computing

Remote communication
- protocol layering, RPC, end-to-end args...

Fault tolerance
- ACID, two-phase commit, nested transactions...

High Availability
- replication, rollback recovery, ...

Remote information access
- distributed file systems, distributed databases, caching, ...

Distributed security
- encryption, mutual authentication, ...

Mobile networking
- Mobile IP, ad hoc networks, wireless TCP fixes, ...

Mobile information access
- disconnected operation, weak consistency, ...

Adaptive applications
- proxies, transcoding, agility, ...

Energy-aware systems
- goal-directed adaptation, disk spin-down, ...

Location sensitivity
- GPS, WaveLan triangulation, context-awareness, ...

Distributed Systems → Mobile Computing → Pervasive Computing

Smart spaces
- Invisibility
- Localized scalability
- Uneven conditioning

By Satyanarayanan.
Continued Service is a Challenge

- Any time, any where service
- Device, network mobility
- Adaptation, context aware
- Application software versus infrastructure system
Distributed System

Remote communication
- protocol layering, RPC, end-to-end args . . .

Fault tolerance
- ACID, two-phase commit, nested transactions . . .

High Availability
- replication, rollback recovery, . . .

Remote information access
- dist. file systems, dist. databases, caching, . . .

Distributed security
- encryption, mutual authentication, . . .

Mobile networking
- Mobile IP, ad hoc networks, wireless TCP fixes, . . .

Mobile information access
- disconnected operation, weak consistency, . . .

Adaptive applications
- proxies, transcoding, agility, . . .

Energy-aware systems
- goal-directed adaptation, disk spin-down, . . .

Location sensitivity
- GPS, WaveLan triangulation, context-awareness, . . .

Distributed Systems

Mobile Computing

Pervasive Computing

Smart spaces
- Invisibility
- Localized scalability
- Uneven conditioning

By Satyanarayanan.
Grid Computing

- Global computing infrastructure
- Mimic electrical power grid
- Resources sharing
- Research Focus
  - Abstraction of Common Services into an Infrastructure
  - Supporting application development
Smart Cyberspace as a Challenge

- Grid as the computing infrastructure
- Embedded systems form the end `smart space’ for `human centered’ service
- Embedded devices as the entry to the cyberspace
- Global smartness
Context Aware System Design

- Existing context aware systems
  - Tied to a specific platform
  - Requires too much knowledge to expand
  - Too difficult to expand
  - Not modular and lacked reuse of common functionality

- Seeking to develop a framework for something better
  - Infrastructure – common functionality
  - Applications – adaptation-specific functionality
Separation of Application and Infrastructure

Infrastructure

(2) Decide and find the devices, which can provide above requested information
(3) Collect and store context information on timely basis
(4) Compose raw context information to meaningful situation

Application

(1) Decide the desired context information
(5) If a meaningful situation occurs then take some action.
Software Engineering (finally!)

- Increasing complexities due to growth in technologies
- Integration and Extensibility are critical issues
- Current design approaches cannot decouple the complexities arising out of integration and evolution intent

Need Better Design Methodology !!!
Can Aspect Orientation Help?

Monterey workshop 03
Aspect Oriented Software Development

- Relatively new design methodology based on the principle of ‘Separation of Concerns’
- ‘Concern’
  - Well defined entity in a software (e.g. Security, Synchronization, Logging, Functional properties, etc)
- ‘Crosscutting Concern’
  - A concern whose implementation crosscut the implementation of other concerns
  - Results in ‘Tangled Code’
- ‘Aspect’
  - A modularized realization of crosscutting concern
Example

- **Good Modularity**
  - XML Parsing concern modularized in one component

- **Bad Modularity**
  - Code for Logging concern is scattered across multiple components

Figures from AspectJ tutorial – source code for org.apache.tomcat
Adaptation is a Concern of CA Applications

- Concern A
- Concern B: CA Adaptor
Context Aware Infrastructure Requirements

**Functional Requirements**

- Context collection
- Context Storage/Management
- Context Subscription/Delivery
- Context Analysis/Composition Ability

**Non Functional Requirements**

- Scalability
- Modularity
- Cross platform
- Security
- Extensibility
- Ability to Evolve
- Quality of Service
- Fault Tolerance
- Mobility
- User Friendly Interface

Context Aware System is a good candidate to apply Aspect Orientation!!!
Implementation

- **Scarlet**: A IIT context aware infrastructure prototype
- Follows design principles of Grid’s OGSA (Open Grid Service Architecture)
  - Grid computing has faced similar challenges
  - Caused much of the problems with the change from Globus 2.x to Globus 3 (OGSA)
Cross-Platform

- OS & language independent
- Specifying a comm. model, rather than object parameters
- SOAP over HTTP for most communication
- Context providers are described using WSDL documents

Modular

- Every part of Scarlet is a module
  - base, provider, consumer, registry, etc
- Allows easy replacement with customized components
- Only needed components are loaded
Extensible

- In the future we may need additional functionality
  - Context caching, preemptive fetching, etc
- Different size systems have different needs

Scalable

- Supporting all devices from embedded sensors to super computers
- Acceptable to deliver limited functionality for some devices
  - A handheld doesn't need a domain registry
Some Implementation Details

- Uses following tools
  - Python 2.2
  - SOAPpy 0.10
  - PyXML 0.8.2
- Runs on variety of systems (not complete list)
  - Windows 98/2000/XP
  - Linux x86, MIPS, SPARC, and Arm (embedded)
  - OpenBSD x86 and SPARC
  - Solaris x86 and SPARC
Scarlet Network
Sample Applications

- Service Browser
- Wireless Strength Monitor
- Television Assistant
- Tour Guide
Conclusion

- Position: Technical advance in embedded system lead to new challenges in software engineering/development
- **Context aware, Continued Service, Global Smartness**
- **Aspect Orientated**: a base of software design
- **Scarlet**: a context aware computing infrastructure
- **Questions** in Software Engineering
  - New language for context representation?
  - New modeling for context awareness?
  - Is the client/server model a good model for Grid computing?
  - Can peer-to-peer extended beyond file sharing?
  - …