Software Engineering Support for the Development of Adaptive, Mobile and Pervasive Computing Applications

Nearchos Paspallis
Department of Computer Science
nearchos@cs.ucy.ac.cy

University of Cyprus

Presentation Structure

Part I

Definitions, Challenges and State of the Art

Part II

Motivation, Current Results and Future Work

Definitions, Challenges and State of the Art

University of Cyprus

Outline

- Software Engineering
  - Evolution of the Programming Model
- Mobile and Pervasive Computing Systems
  - Challenges
  - Autonomic and Proactive Computing
- Context Awareness and Adaptive Systems
  - Separation of Concerns
- State of the Art
  - Research & Commercial Products

Software Engineering

- The application of *engineering principles* and *design methods* to the production of software
- Covers the complete lifetime of software
  - *Design*, *creation* and *maintenance*
- Requirements
  - Minimize *development cost* and *duration*
  - Detect and contain *development risks*
  - Minimize *maintenance cost*

Component-based Software Engineering

- Software Components
  - C. Szyperski: *"…units of composition with contractually specified interfaces and explicit context dependencies only; [they] can be deployed independently and are subject to composition by third parties"*
  - *Black-box* versus *White-box* abstractions
  - Also *Glass-box*, *Gray-box* abstractions
- Establish a marketplace of software parts that can be used in multiple projects
  - Shock-absorbers example

Distributed Programming

- Evolution of the Distributed Programming Model
  - 1970's: Messages
    - *Enabling the delivery and receipt of messages allows distributed computers to coordinate and collaborate*
  - 1980's: Procedure Calls
    - *Procedure calls abstract some of the difficulties of message communication, i.e. by enabling synchronous communication*
  - 1990's: Objects
    - *The use of objects abstracts even more details away from the developers, allowing them to act on remote objects in a way similar to local objects*
  - 2000's: *Components, Middleware, Services*
    - *The use of services is the latest approach to distributed computing; it introduces many new technologies such as transparent fault tolerance, transaction support, interoperability, etc*

Local Versus Distributed Computing

- Latency
  - Remote calls are 4-5 orders of magnitude worse than local calls (most likely worse in the future)
  - The most apparent difference but not the most fundamental
- Memory Access
  - Memory access (i.e. using pointers) is fundamentally different in local versus distributed systems
  - Complete transparency or complete user control
- Partial Failure and Concurrency
  - These appear to be the *fundamental* differences of distributed computing
  - Partial failure is a significant part of distributed computing
  - Concurrency is inherent in distributed computing as opposed to local, multi-threaded computing

Mobile and Pervasive Computing

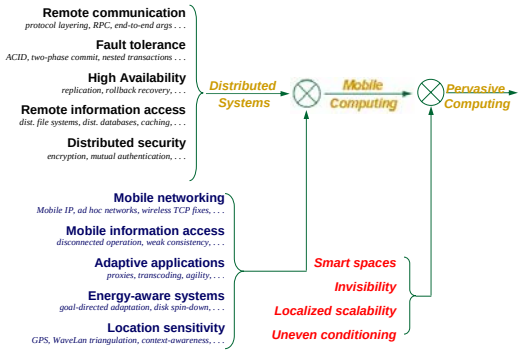
- Mobile Computing
  - Distributed systems with a network to communicate between different, mobile machines
  - Wireless communication enables mobility
  - Constraints of mobile computing environments
    - Poor resource availability
    - Less secure and less reliable compared to static counterparts
    - Mobile connectivity can be highly variable in terms of *bandwidth*, *latency*, and *reliability*

Mobile and Pervasive Computing

- Pervasive Computing
  - First introduced by *Mark Weiser* of Xerox PARC (as *Ubiquitous Computing*)
  - Three waves of computing
    - Past: Few mainframes shared by lots of people
    - Present: Personal computing era (one-on-one)
    - Future: Many computers for each person (mobiles, PDAs, etc)
  - The age of *calm technology*
    - *… the technology recedes to the background of our lives, seamlessly offering its services in a transparent way*

Evolution of Distributed Computing

[Satyanarayanan , *Pervasive Computing: Vision and Challenges*, IEEE 2001]



Pervasive Computing

- Requirements for evolving to the Pervasive Computing era
  - Smart spaces
    - Embedding computing devices in the our environment
    - Convergence of the *physical* and *computing* worlds
  - Invisibility
    - Technology recedes to the background of our lives (literally and metaphorically)
  - Localized scalability
    - Too many computing outlets interacting together (inverse square law)
  - Uneven conditioning
    - Non-uniform penetration of pervasive computing (transition period)

Core Challenges in Pervasive Computing

- Resource poverty
  - Smaller computing devices
- Communication uncertainty
  - Wireless communications
- Finite energy sources
  - Portable computing devices
- Multi-model interaction
  - Visual, audio, etc
- Scarce user attention
  - Ubiquitous computing vision
- Less security and robustness
  - Inherent in wireless communications

Autonomic Computing

- Definition
  - "*designing and building systems capable of running themselves, adjusting to varying circumstances and managing their resources to handle their workloads most efficiently*"
    --Paul Horn
  - Modeled over a self-regulating biological system
    - The human body example: heart beating, lungs breathing, eyes adjusting to light, sweating to cool, etc

Proactive Computing

- Definition
  - "*[it] is about getting out in front and anticipating your needs rather than just reacting to them*"
    --David Tennenhouse
  - *Getting physical*
    - *Bridge the gap between virtual and physical worlds*
  - *Getting real*
    - *Respond to external stimuli at faster-than-human speeds*
  - *Getting out*
    - *From human-centered to human-supervised*

Comparing Pervasive, Proactive and Autonomic Computing

- Both proactive and autonomic computing
  - Embrace the vision for pervasive computing
  - Aim to provide solutions to overcome the growth of today's computing systems
  - Target to minimize the degree of human involvement
- Autonomic computing
  - Modeled over biological systems
  - Aims at equipping systems with self-regulating mechanisms
- Proactive computing
  - Extends autonomic and pervasive computing
  - Targets *monitoring* and *shaping* of the physical world

Context Awareness

- Definition
  - "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.*"

    -- Anind K. Dey
- Classification of context
  - Computing context
  - User context
  - Physical context

Context Awareness

- Classification of applications
  - Active context-aware applications
  - Passive context-aware applications
- Centralized versus Distributed Architectures
  - Financial motive for distribution (less hardware)
  - Distribution challenges
    - Standardized context modeling
    - Interoperability
    - Semantics
    - Privacy, trust and security

Adaptive Computing

- The ability of software or hardware components to dynamically change their behavior at runtime
  - Adaptations aim at limiting the resource consumption or improving the user experience
  - Fundamental technology for mobile and pervasive computing
  - Examples
    - Switching WiFi on and off, or switching from WiFi to GPRS and back
    - Intel Pentium Speed-Step technology
    - Hard disk spin-down
    - Automated hoarding
    - Switching from visual to audio interaction and back

Adaptive Computing

- Classification
  - Parameter adaptation
    - TCP control window parameter adaptation
    - Switch between a set of (pre-defined) strategies
  - Compositional adaptation
    - Supports the use of algorithms which were not available during original development
    - Switch between a set of (possibly new) strategies

State of the Art in Research Projects

- Medium and large scale research projects
  - NSF funded
    - Aura – Captures high level intends (auras) / Values user attention
    - Coda – Distributed FS with transparent synchronization (hoarding)
    - Odyssey – Data access functionality API (QoS Vs Resource Consumption) / Targets existing applications
    - Rainbow – Builds on software architectures (graphs of interacting elements) / Tries to separate the application from the adaptation logic
  - IST funded
    - Madam – Aims to ease the development of adaptive, mobile applications / Similar to Rainbow, it also tries to separate concerns
    - Music – Targets to provide a middleware for pervasive computing
    - Runes – Middleware-based/ Components with well defined *interfaces* and *receptacles* / Unlike Madam, not a complete development toolkit
  - Industrial projects
    - Active badge/map, Call forwarding – Logical mobility applications
    - Jini, UPnP, Bonjour – Allow interoperability among commercial products

Lessons Learned

- The areas of mobile and pervasive computing have a significant momentum
- Dealing with the software development complexity is one of the greatest challenges faced by the IT industry
- *Software engineering* support can facilitate the software development of mobile and pervasive systems
  - *Monitoring* and *shaping* the environment
  - Supporting *more intelligent* interoperability

Motivation, Current Results and Future Work

University of Cyprus

Outline

- Motivation Scenario
  - Pervasive computing
    - Ambient umbrella
    - Ambient picture frame
  - Challenges
- Open Research Questions
  - Development complexity
  - User, context and application modeling
  - Middleware support
- Research Plan
  - Current results
  - Research approach
  - Potential impact
  - Research plan

Motivation Scenarios

- Illustrate which technologies are currently possible and which technologies are not
- Demonstrate the need for distributed adaptations as an enabling technology
- Highlight the research opportunities for automated context management & adaptation reasoning

Ambient Umbrella

- Users buy the umbrella and simply leave it next to their door
- The handle changes color according to the weather forecast to let users know whether they will need it during the course of the day or not

- Data is (uni-directionally) communicated over ubiquitous networks (e.g. GSM/GPRS networks)

Ambient Picture Frame

- Users buy the digital frame
- Then they upload their pictures and place the frame at a favorite spot to continuously display a slideshow of the pictures

- By today's standards, only a minimum of configurations and adaptations are possible
  - E.g. setting the time when the display should automatically turn on and off

Ambient Picture Frame

- Many possibilities exist to make this device truly adaptive and improve its utilization
  - Automatically realize when there are potential viewers in the area so that the display is automatically turned on and off
  - Automatically adjust the display brightness and speaker volume to match the ambient light and noise conditions
  - Delegate the display to a nearby TV screen when selected
  - Delegate the control to other devices when needed (e.g. to a universal remote control or an ambient voice recognition system)

Ambient Picture Frame

- Centralized Vs Decentralized
  - Centralized implies that all the functionality is embedded in the digital frame (light and noise sensors, RFID readers, etc)
    - **Significantly higher manufacturing cost!**
  - Decentralized solutions introduce *more development complexity* but they provide better utilization of the equipment (i.e. reuse)
    - **Important savings in manufacturing cost**
  - Some adaptations are not even possible with the centralized approach (e.g. the display delegation)

Challenges

- Developing software for pervasive computing
  - **Conceptualization**
    - Dynamic and unpredictable deployment environment
    - Support synergies even among dynamically available components
  - **Implementation**
    - Interoperability (evident need for standards)
    - Testing and certification
    - Reusability of code and middleware support
  - **Maintenance**
    - Allow user interventions to assist with uneven conditioning
    - Privacy and security issues

Open Research Questions

- How is the development complexity tamed?
  - Pervasive computing requires interdisciplinary solutions
  - Handle both hardware and software adaptations
  - Transition period (uneven conditioning)
  - Software complexity increases development effort!
    - Separation of concerns
    - Reusable software components
    - Middleware support
    - Reuse and adapt techniques from service & component engineering

Open Research Questions (continued)

- How can users, components, services and resources be modeled in pervasive computing?
  - The users, components, services and resources are all elements of the pervasive computing environment
  - Developing software for pervasive computing applications requires that these elements are appropriately modeled
  - Support automated reasoning for self-adaptation!
    - Rule-based Vs Utility function-based decisions
    - Distributed decisions (negotiation)
    - Localized scalability, Interoperability, Uneven conditioning
    - Methods for encoding the actual user needs to utility functions

Open Research Questions (continued)

- How can Middleware architectures benefit the development of software for pervasive and mobile computing?
  - Environment *Monitoring*
    - Context management
    - Sharing of context information
  - Environment *Shaping*
    - Reasoning on and implementing adaptations
    - To which extend can the context monitoring and software adaptivity be hidden from the developers?
    - Separation of concerns

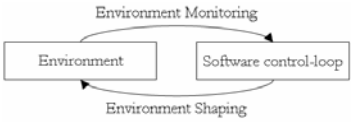
Research Approach

- What is the overall goal
  - Propose and demonstrate improvements in the software engineering process for developing adaptive, mobile and pervasive computing applications
    - Propose basic models which can be used to enable both *monitoring* and *shaping* of the environment
    - Provide prototype implementation of these models to justify the validity and the appropriateness of these models
    - Design and implement a pluggable middleware architecture which enables the implementation and deployment of distributed, adaptive applications
    - Propose and verify (both theoretically and experimentally) different architecture plug-ins (mainly wrt adaptation logic implementation)

Current Results

Software engineering support in two dimensions

- Environment *monitoring*
- Environment *shaping*



Current Results

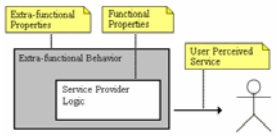
- Environment *monitoring*
  - Implemented a context management architecture
    - Pluggable context providers (sensors and reasoners)
      - Reusable components
      - Offers a simulated mode
    - Integration with the middleware (e.g. only context elements required by running applications are monitored)
    - Currently extending the approach to support distribution of context

Current Results

- Environment *shaping*
  - Implemented a prototype component framework
    - Allows developers to specify *adaptive* components
    - Automatically manages the components lifecycle
  - Provided support for dynamic re-compositions
    - Adaptation characteristics described with *Java annotations*
    - Adaptation strategies defined with utility functions
  - Main contribution
    - Using *separation of concerns* to mitigate the development cost

Current Results

- Separation of concerns



  - Developers create code by independently defining the functional and the extra-functional properties of software
    - Functional properties: provided & required interfaces
    - Extra-functional properties: Component properties (through Java annotations) & Adaptation strategies (currently with utility functions)

Potential Impact

- Today the design of adaptive, mobile and pervasive computing systems is *cutting edge* activity
  - Evaluate the merits of alternative designs and implementation strategies
  - Create a body of knowledge which the designers of adaptive, mobile and pervasive systems can consult
  - Directly or indirectly contribute to the formation and the specification of suitable protocols and standards
  - Controlled and reproducible experiments (simulations)
  - Demonstrate cost-savings in the process of modeling, prototyping and developing adaptive applications

Research Plan

- Expand and validate the knowledge

6 months

- Extend the component framework to implement the intended pluggable architecture

12 months

- Validate and compare alternative adaptation reasoning approaches
- Examine the possibility of applying control-theory results to the development of adaptive software for pervasive computing

18 months+

- Gather experimental results to validate different adaptation reasoning techniques
- Run real experiments to evaluate the applicability of the proposed adaptation reasoning techniques

Questions…

University of Cyprus

Current Publications

- *Monitoring* the environment
  - Marius Mikalsen, Nearchos Paspallis, Jacqueline Floch, Erlend Stav, Akis Chimaris, George A. Papadopoulos, **Distributed Context Management in a Mobility and Adaptation Enabling Middleware (MADAM)**. *21st Annual ACM Symposium of Applied Computing, Track of Dependable and Adaptive Systems (SAC)*, Dijon, France, April 23 -27, 2006, pp. 733-734
  - Marius Mikalsen, Nearchos Paspallis, Jacqueline Floch, George A. Papadopoulos, Pedro Antonio Ruiz, **Putting Context in Context: The Role and Design of Context Management in a Mobility and Adaptation Enabling Middleware**, *7th International Conference on Mobile Data Management (MDM)*, Nara, Japan, May 9-12, 2006, IEEE Computer, pp.76-83
- *Shaping* the environment
  - Nearchos Paspallis, George A. Papadopoulos, **An Approach for Developing Adaptive, Mobile Applications with Separation of Concerns**, *30th Annual International Computer Software and Applications Conference (COMPSAC)*, Chicago, IL, USA, September 17-21, 2006, IEEE (to appear)
  - Nearchos Paspallis, George A. Papadopoulos, **Distributed Adaptation Reasoning for a Mobility and Adaptation Enabling Middleware**, *8th International Symposium on Distributed Objects and Applications (DOA)*, Montpellier, France, Oct 30 - Nov 1, 2006, Springer, pp. 17-18