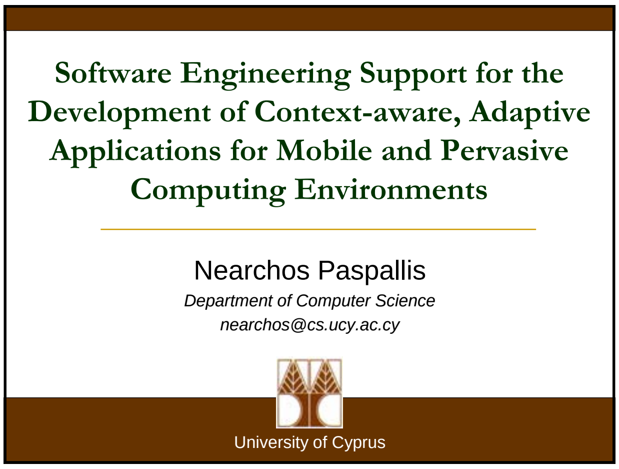


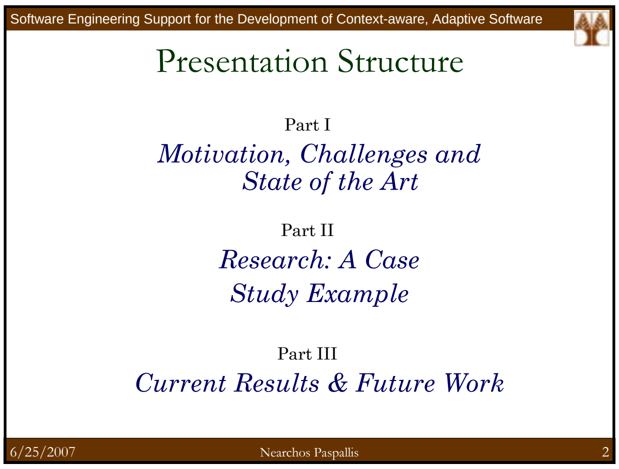
Software Engineering Support for the Development of Context-aware, Adaptive Applications for Mobile and Pervasive Computing Environments

Nearchos Paspallis

Department of Computer Science

nearchos@cs.ucy.ac.cy

University of Cyprus



Software Engineering Support for the Development of Context-aware, Adaptive Software

Presentation Structure

Part I

Motivation, Challenges and State of the Art

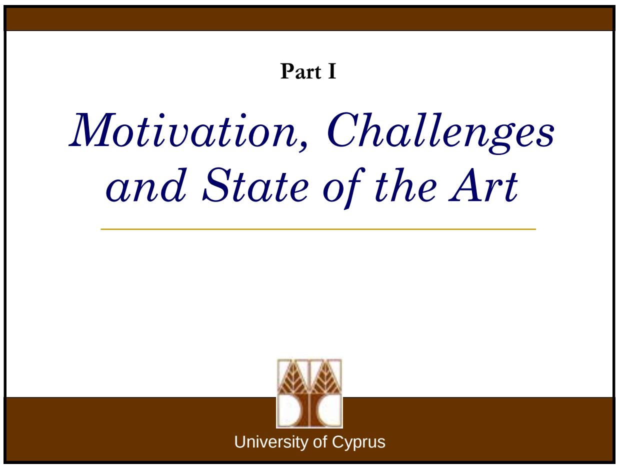
Part II

Research: A Case Study Example

Part III

Current Results & Future Work

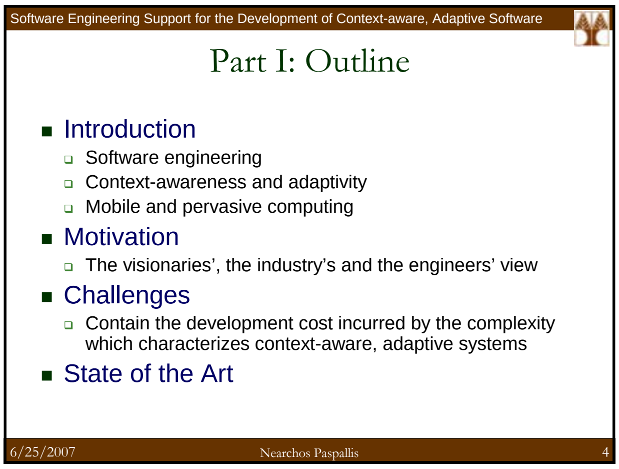
6/25/2007 Nearchos Paspallis 2



Part I

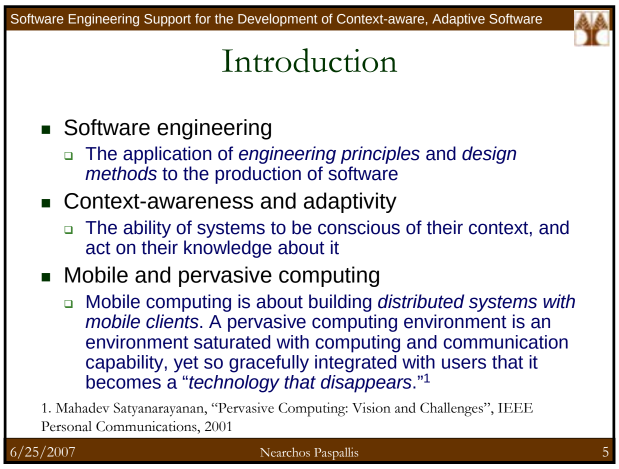
Motivation, Challenges and State of the Art

University of Cyprus



Part I: Outline

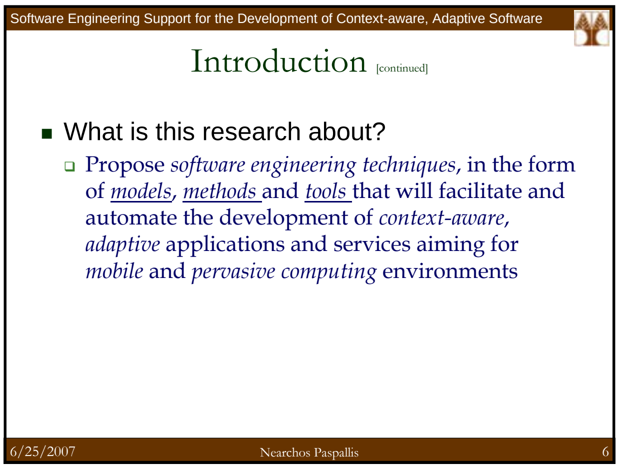
- Introduction
  - Software engineering
  - Context-awareness and adaptivity
  - Mobile and pervasive computing
- Motivation
  - The visionaries', the industry's and the engineers' view
- Challenges
  - Contain the development cost incurred by the complexity which characterizes context-aware, adaptive systems
- State of the Art



Introduction

- Software engineering
  - The application of *engineering principles* and *design methods* to the production of software
- Context-awareness and adaptivity
  - The ability of systems to be conscious of their context, and act on their knowledge about it
- Mobile and pervasive computing
  - Mobile computing is about building *distributed systems with mobile clients*. A pervasive computing environment is an environment saturated with computing and communication capability, yet so gracefully integrated with users that it becomes a "*technology that disappears*."[1]

1. Mahadev Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, 2001



Introduction [continued]

- What is this research about?
  - Propose *software engineering techniques*, in the form of *models*, *methods* and *tools* that will facilitate and automate the development of *context-aware, adaptive* applications and services aiming for *mobile* and *pervasive computing* environments

Motivation

- The visionaries' view

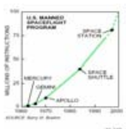
  "… a vision of processing power so distributed throughout the environment that computers per se effectively disappear".[2]
  - An immediate issue and a hundred-year problem
  - Missing: standards for interoperability, ubiquitous network infrastructure, *appropriate design documents and conventions*, a compelling and clearly stated value proposition
  - Already have: necessary processor speed and storage capacity, addressing scheme, display technologies, wireless networking protocols, bridges between atoms and bits, standards for representation and communication of structured data

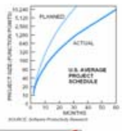
2. Adam Greenfield, "Everyware: The dawning age of ubiquitous computing", New Riders publishing, ISBN: 0-321-38401-6

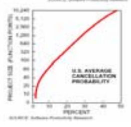
Motivation [continued]

- The industry's view
  - The software problem

"In 2000, total sales of software reached approximately $180 billion, supported by a large workforce encompassing 697,000 software engineers and 585,000 computer programmers. [in US]"[4]







Users as bugs

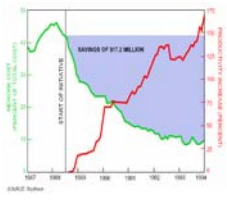
Users as evolution

3. Gibbs, W. Wayt, "Software's Chronic Crisis," Scientific American, September 1994, pp. 86–95.

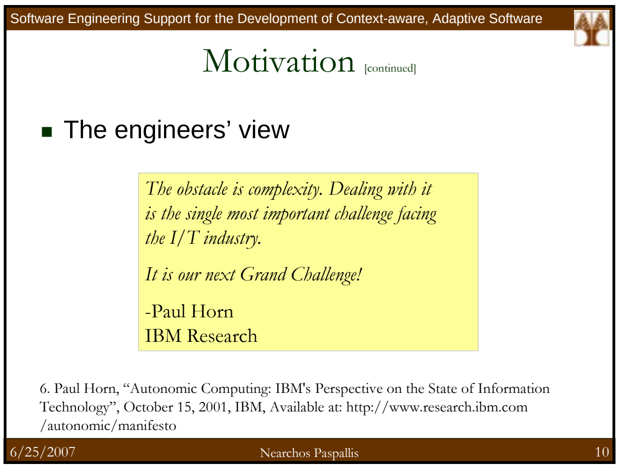
4. National Institute of Standards and Technology, "Software Errors Cost U.S. Economy $59.5 Billion Annually, NIST Assesses Technical Needs of Industry to Improve Software-Testing", Available at: http://www.nist.gov/public_affairs/releases/n02-10.htm

Motivation [continued]

- The industry's view
  - Software engineering matters
    - Raytheon saved $17.2M since 1988 [until 1994] by applying more rigorous software development processes
  - The changing face of software
    - Web 2.0, Developing world, *Ubiquitous computing*
    - Open source and *agile methodologies*
    - Services, AJAX, *AOP*, *Model-driven software development*



5. Bill Griswold, "CSE 218: Advanced Topics in Software Engineering", Available at: http://www.cs.ucsd.edu/~wgg/CSE218/index.html



Software Engineering Support for the Development of Context-aware, Adaptive Software

Motivation [continued]

- The engineers' view

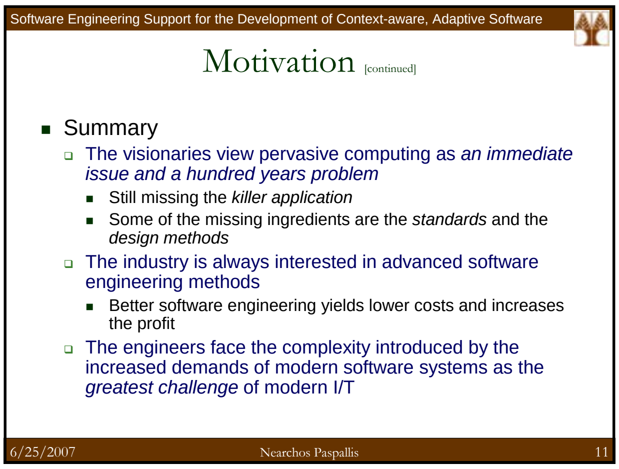
The obstacle is complexity. Dealing with it is the single most important challenge facing the I/T industry.

It is our next Grand Challenge!

-Paul Horn
IBM Research

6. Paul Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology", October 15, 2001, IBM, Available at: http://www.research.ibm.com /autonomic/manifesto

6/25/2007 Nearchos Paspallis 10

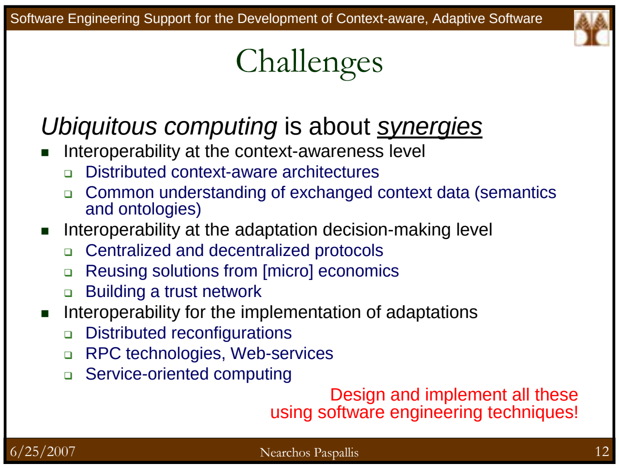


Software Engineering Support for the Development of Context-aware, Adaptive Software

Motivation [continued]

- Summary
  - The visionaries view pervasive computing as *an immediate issue and a hundred years problem*
    - Still missing the *killer application*
    - Some of the missing ingredients are the *standards* and the *design methods*
  - The industry is always interested in advanced software engineering methods
    - Better software engineering yields lower costs and increases the profit
  - The engineers face the complexity introduced by the increased demands of modern software systems as the *greatest challenge* of modern I/T

6/25/2007 Nearchos Paspallis 11



Software Engineering Support for the Development of Context-aware, Adaptive Software

Challenges

Ubiquitous computing is about *synergies*

- Interoperability at the context-awareness level
  - Distributed context-aware architectures
  - Common understanding of exchanged context data (semantics and ontologies)
- Interoperability at the adaptation decision-making level
  - Centralized and decentralized protocols
  - Reusing solutions from [micro] economics
  - Building a trust network
- Interoperability for the implementation of adaptations
  - Distributed reconfigurations
  - RPC technologies, Web-services
  - Service-oriented computing

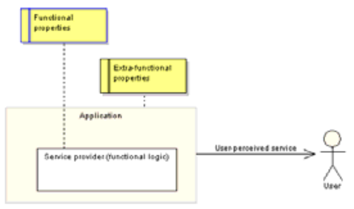
Design and implement all these using software engineering techniques!

6/25/2007 Nearchos Paspallis 12

Challenges [continued]

Ubiquitous computing is a crosscutting concern

- Ubiquitous computing is about seamlessly adapting the interaction between a user and a system with the purpose of optimizing her or his *perceived* Quality of Service


Functional properties
Extra-functional properties
Application
Service provider (functional logic)
User perceived service
User

State of the Art

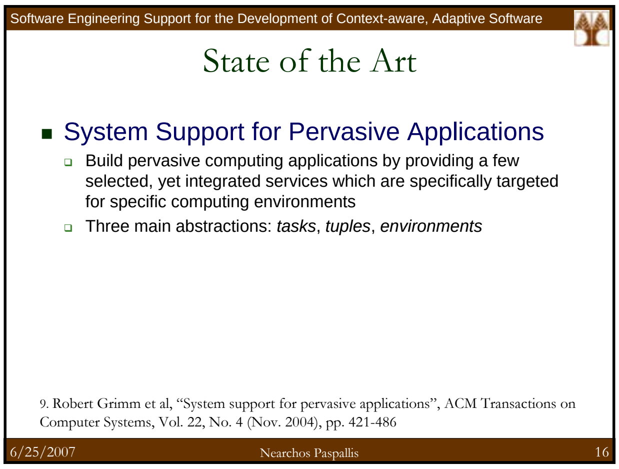
- Using Architecture Models for Runtime Adaptability
  - Components and composition plans are defined offline
  - The adaptation strategy is hard-coded in the composition plans
  - Distributed planning and reasoning are limited to a centralized model
  - All possible application variants are constructed and evaluated dynamically at runtime; the optimal one is selected each time using utility functions

7. Jacqueline Floch, Svein Hallsteinsen, Frank Eliassen, Ketil Lund, and Eli Gjørven, "Using Architecture Models for Runtime Adaptability", IEEE Software, Mar./Apr. 2006 Vol. 23, No. 2, pp. 62-70.

State of the Art

- ISAMadapt: Abstractions and Tools for Designing General-purpose Pervasive Applications: Experiences with Auto-adaptive and Reconfigurable Systems
  - Context-awareness is expressed at the programming language level with a basis of four main abstractions: *context*, *adapters*, *adaptation commands* and *behavior management policies*
  - ISAMadapt: an explicit *pervasive programming language*
  - EXEHDA : an underlying middleware supporting ISAMadapt

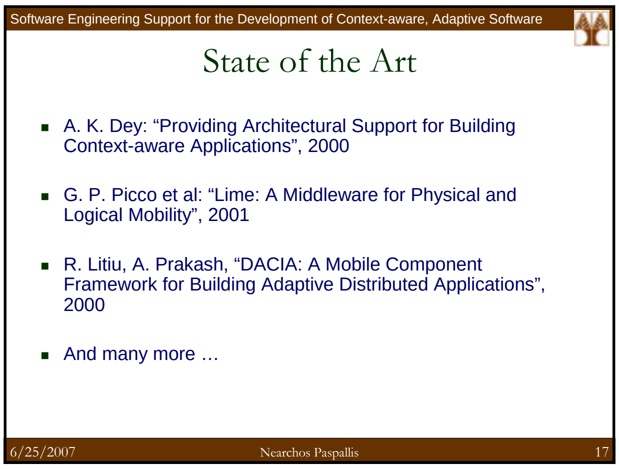
8. I. Augustin, A. C. Yamin, L. C. da Silva, R. A. Real, G. Frainer, C. F. Geyer, "ISAMadapt: Abstractions and Tools for Designing General-purpose Pervasive Applications: Experiences with Auto-adaptive and Reconfigurable Systems", Software Practice and Experience, Vol. 36, No. 11-12 (Sep. 2006), pp. 1231-1256.



State of the Art

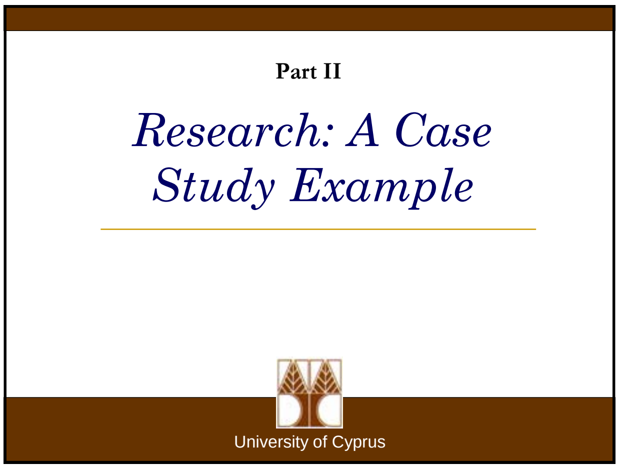
- System Support for Pervasive Applications
  - Build pervasive computing applications by providing a few selected, yet integrated services which are specifically targeted for specific computing environments
  - Three main abstractions: *tasks*, *tuples*, *environments*

9. Robert Grimm et al, "System support for pervasive applications", ACM Transactions on Computer Systems, Vol. 22, No. 4 (Nov. 2004), pp. 421-486



State of the Art

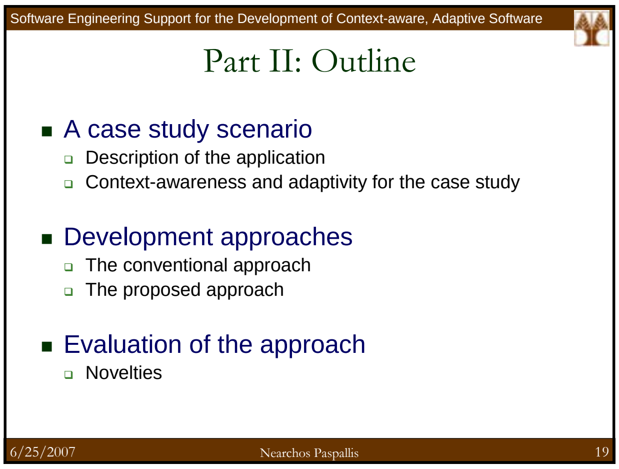
- A. K. Dey: "Providing Architectural Support for Building Context-aware Applications", 2000
- G. P. Picco et al: "Lime: A Middleware for Physical and Logical Mobility", 2001
- R. Litiu, A. Prakash, "DACIA: A Mobile Component Framework for Building Adaptive Distributed Applications", 2000
- And many more …



Part II

Research: A Case Study Example

University of Cyprus

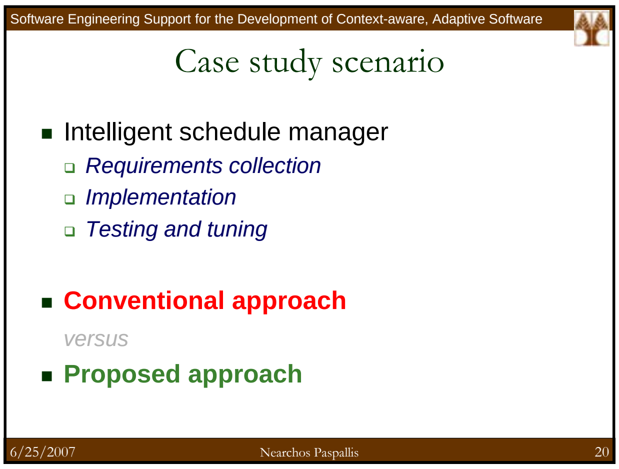


Software Engineering Support for the Development of Context-aware, Adaptive Software

Part II: Outline

- A case study scenario
  - Description of the application
  - Context-awareness and adaptivity for the case study
- Development approaches
  - The conventional approach
  - The proposed approach
- Evaluation of the approach
  - Novelties

6/25/2007 Nearchos Paspallis 19



Software Engineering Support for the Development of Context-aware, Adaptive Software

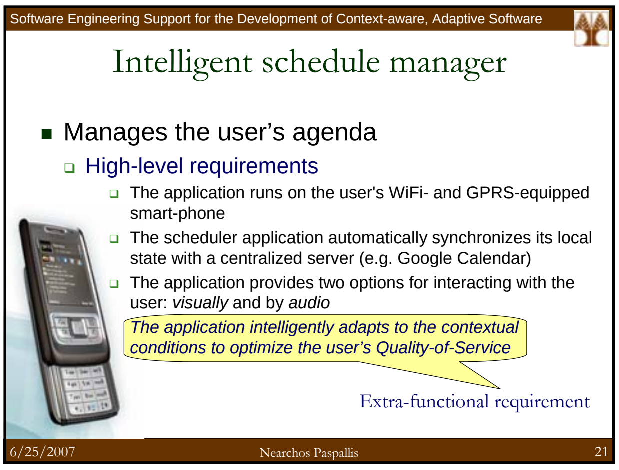
Case study scenario

- Intelligent schedule manager
  - *Requirements collection*
  - *Implementation*
  - *Testing and tuning*
- **Conventional approach**

versus

- **Proposed approach**

6/25/2007 Nearchos Paspallis 20



Software Engineering Support for the Development of Context-aware, Adaptive Software

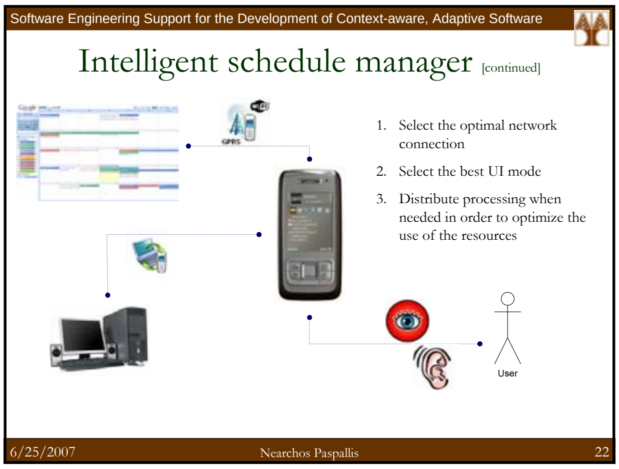
Intelligent schedule manager

- Manages the user's agenda
  - High-level requirements
    - The application runs on the user's WiFi- and GPRS-equipped smart-phone
    - The scheduler application automatically synchronizes its local state with a centralized server (e.g. Google Calendar)
    - The application provides two options for interacting with the user: *visually* and by *audio*

The application intelligently adapts to the contextual conditions to optimize the user's Quality-of-Service

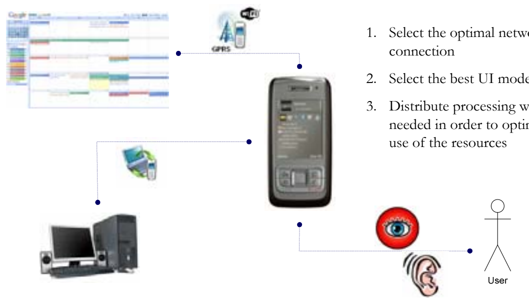
Extra-functional requirement

6/25/2007 Nearchos Paspallis 21



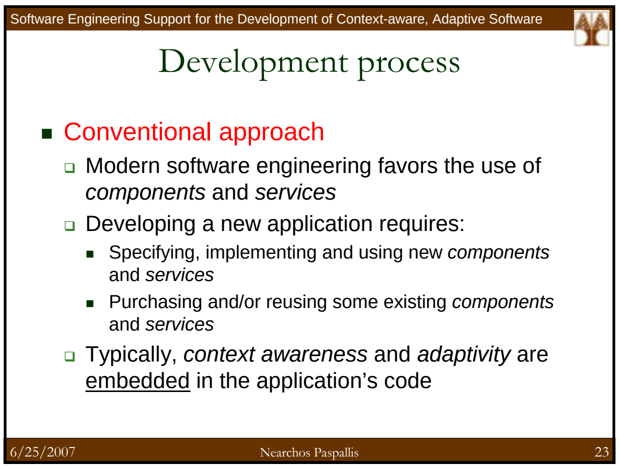
Software Engineering Support for the Development of Context-aware, Adaptive Software

Intelligent schedule manager [continued]



1. Select the optimal network connection
2. Select the best UI mode
3. Distribute processing when needed in order to optimize the use of the resources

6/25/2007 Nearchos Paspallis 22

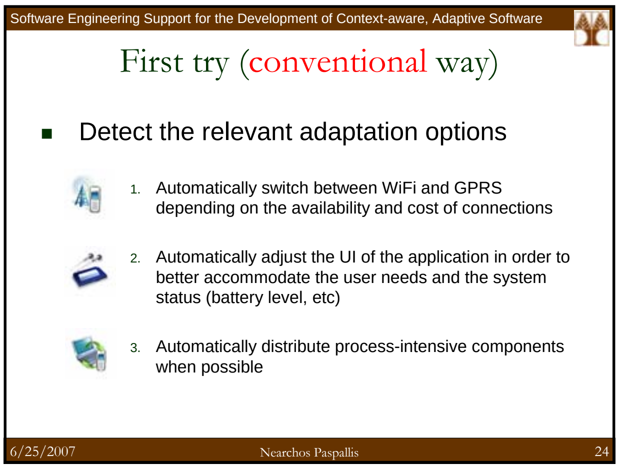


Software Engineering Support for the Development of Context-aware, Adaptive Software

Development process

- Conventional approach
  - Modern software engineering favors the use of *components* and *services*
  - Developing a new application requires:
    - Specifying, implementing and using new *components* and *services*
    - Purchasing and/or reusing some existing *components* and *services*
  - Typically, *context awareness* and *adaptivity* are embedded in the application's code

6/25/2007 Nearchos Paspallis 23



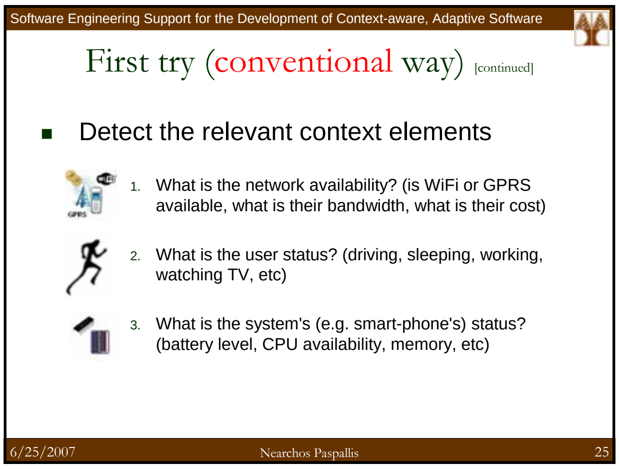
Software Engineering Support for the Development of Context-aware, Adaptive Software

First try (conventional way)

- Detect the relevant adaptation options

1. Automatically switch between WiFi and GPRS depending on the availability and cost of connections
2. Automatically adjust the UI of the application in order to better accommodate the user needs and the system status (battery level, etc)
3. Automatically distribute process-intensive components when possible

6/25/2007 Nearchos Paspallis 24

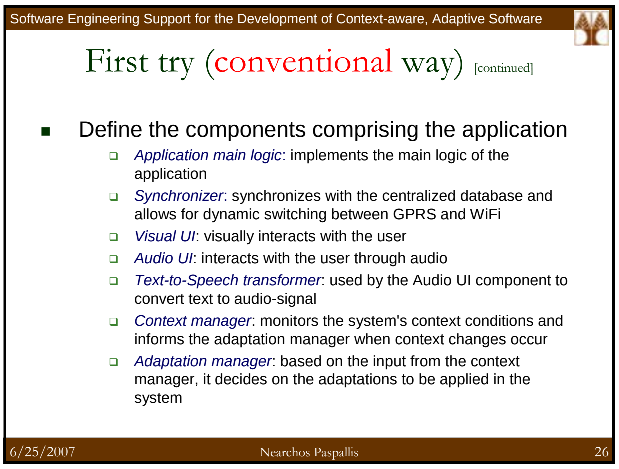


Software Engineering Support for the Development of Context-aware, Adaptive Software

First try (conventional way) [continued]

- Detect the relevant context elements
  1. What is the network availability? (is WiFi or GPRS available, what is their bandwidth, what is their cost)
  2. What is the user status? (driving, sleeping, working, watching TV, etc)
  3. What is the system's (e.g. smart-phone's) status? (battery level, CPU availability, memory, etc)

6/25/2007 Nearchos Paspallis 25

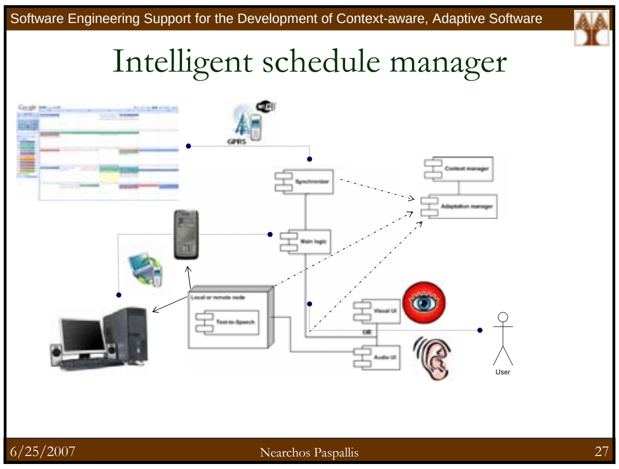


Software Engineering Support for the Development of Context-aware, Adaptive Software

First try (conventional way) [continued]

- Define the components comprising the application
  - *Application main logic*: implements the main logic of the application
  - *Synchronizer*: synchronizes with the centralized database and allows for dynamic switching between GPRS and WiFi
  - *Visual UI*: visually interacts with the user
  - *Audio UI*: interacts with the user through audio
  - *Text-to-Speech transformer*: used by the Audio UI component to convert text to audio-signal
  - *Context manager*: monitors the system's context conditions and informs the adaptation manager when context changes occur
  - *Adaptation manager*: based on the input from the context manager, it decides on the adaptations to be applied in the system

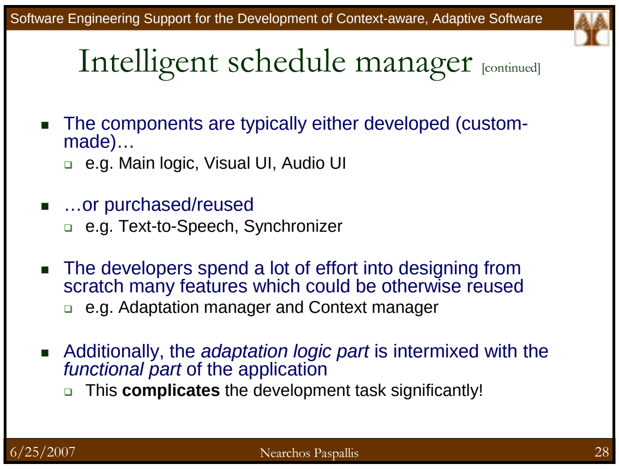
6/25/2007 Nearchos Paspallis 26



Software Engineering Support for the Development of Context-aware, Adaptive Software

Intelligent schedule manager

6/25/2007 Nearchos Paspallis 27

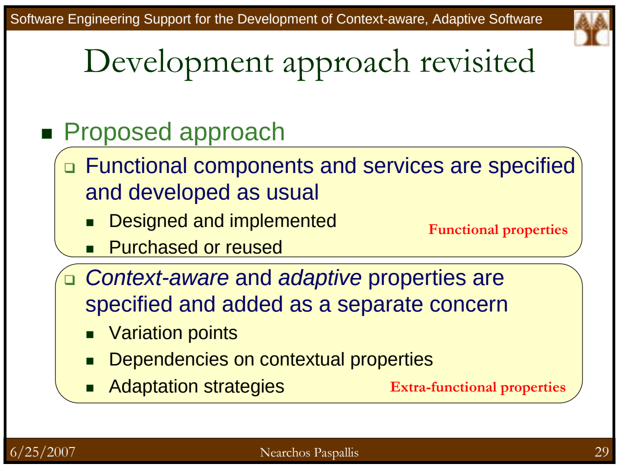


Software Engineering Support for the Development of Context-aware, Adaptive Software

Intelligent schedule manager [continued]

- The components are typically either developed (custom-made)…
  - e.g. Main logic, Visual UI, Audio UI
- …or purchased/reused
  - e.g. Text-to-Speech, Synchronizer
- The developers spend a lot of effort into designing from scratch many features which could be otherwise reused
  - e.g. Adaptation manager and Context manager
- Additionally, the *adaptation logic part* is intermixed with the *functional part* of the application
  - This **complicates** the development task significantly!

6/25/2007 Nearchos Paspallis 28



Software Engineering Support for the Development of Context-aware, Adaptive Software

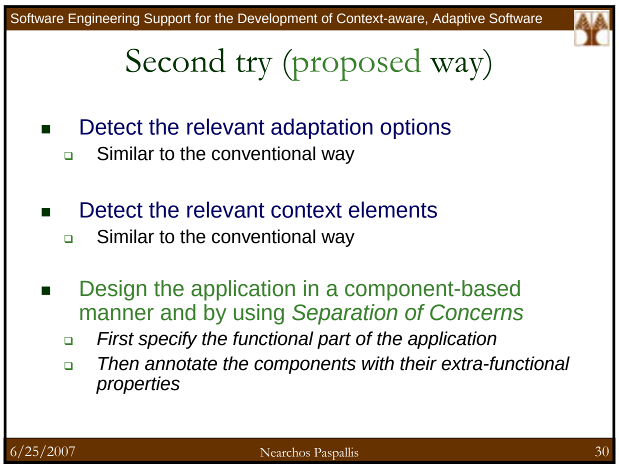
Development approach revisited

- Proposed approach
  - Functional components and services are specified and developed as usual
    - Designed and implemented
    - Purchased or reused

    Functional properties
  - *Context-aware* and *adaptive* properties are specified and added as a separate concern
    - Variation points
    - Dependencies on contextual properties
    - Adaptation strategies

    Extra-functional properties

6/25/2007 Nearchos Paspallis 29

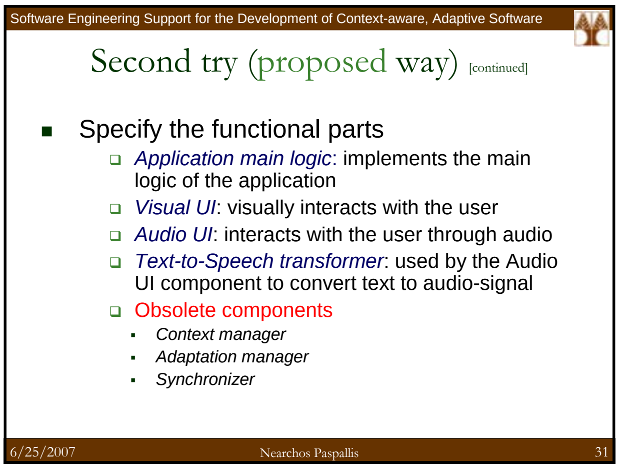


Software Engineering Support for the Development of Context-aware, Adaptive Software

Second try (proposed way)

- Detect the relevant adaptation options
  - Similar to the conventional way
- Detect the relevant context elements
  - Similar to the conventional way
- Design the application in a component-based manner and by using *Separation of Concerns*
  - *First specify the functional part of the application*
  - *Then annotate the components with their extra-functional properties*

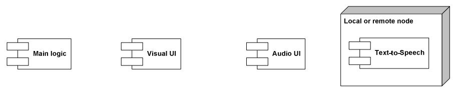
6/25/2007 Nearchos Paspallis 30



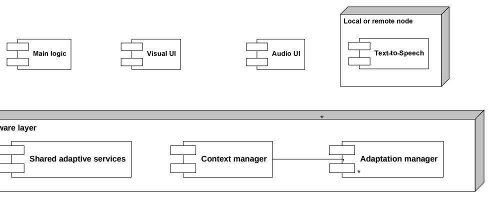
Second try (proposed way) [continued]

- Specify the functional parts
  - *Application main logic*: implements the main logic of the application
  - *Visual UI*: visually interacts with the user
  - *Audio UI*: interacts with the user through audio
  - *Text-to-Speech transformer*: used by the Audio UI component to convert text to audio-signal
  - Obsolete components
    - *Context manager*
    - *Adaptation manager*
    - *Synchronizer*

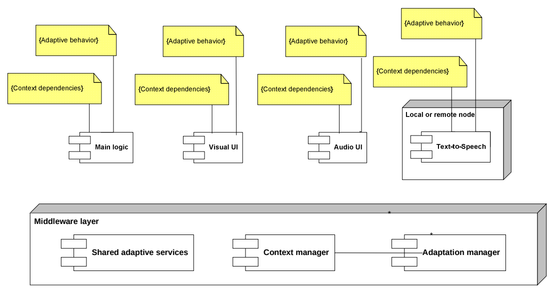
Second try (proposed way) [continued]



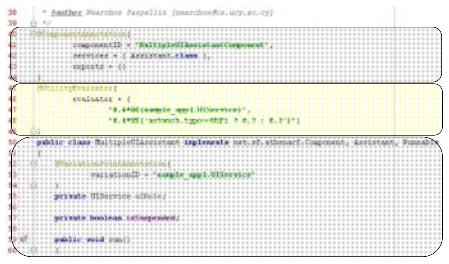
Second try (proposed way) [continued]



Second try (proposed way) [continued]



Second try (proposed way) [continued]

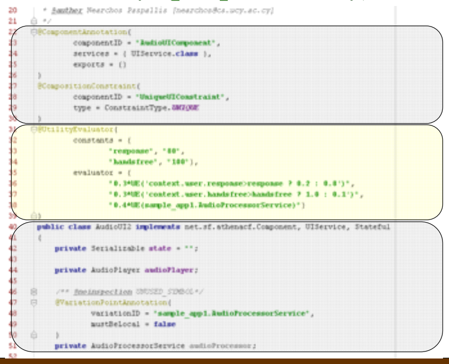


```
 * Author Nearchos Paspallis [nearchos@cs.ucy.ac.cy]
 */
@ComponentAnnotation(
        componentID = "MultipleUIAssistantComponent",
        services = { Assistant.class },
        exports = {}
)
@UtilityEvaluator(
        evaluator = {
                "0.6*UE(sample_app1.UIService)",
                "0.4*UE('network.type==WiFi ? 0.7 : 0.3')"}
)
public class MultipleUIAssistant implements net.sf.athenacf.Component, Assistant, Runnable
{
    @VariationPointAnnotation(
            variationID = "sample_app1.UIService"
    )
    private UIService uiRole;

    private boolean isSuspended;

    public void run()
    {
```

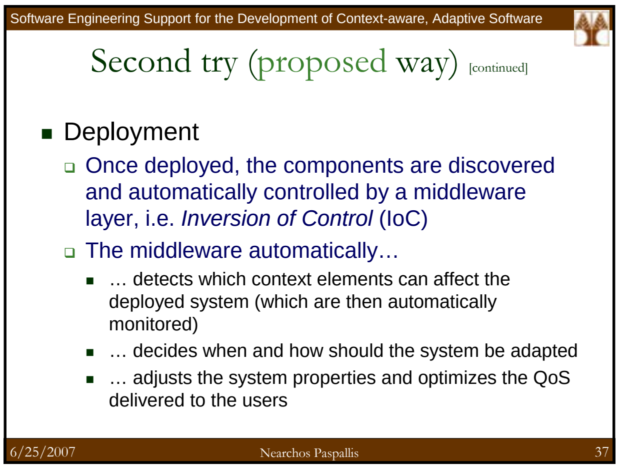
Second try (proposed way) [continued]



```
 * Author Nearchos Paspallis [nearchos@cs.ucy.ac.cy]
 */
@ComponentAnnotation(
        componentID = "AudioUIComponent",
        services = { UIService.class },
        exports = {}
)
@CompositionConstraint(
        componentID = "UniqueUIConstraint",
        type = ConstraintType.UNIQUE
)
@UtilityEvaluator(
        constants = {
                "response", "80",
                "handsfree", "100"},
        evaluator = {
                "0.3*UE('context.user.response>response ? 0.2 : 0.8')",
                "0.3*UE('context.user.handsfree>handsfree ? 1.0 : 0.1')",
                "0.4*UE(sample_app1.AudioProcessorService)"}
)
public class AudioUI2 implements net.sf.athenacf.Component, UIService, Stateful
{
    private Serializable state = "";

    private AudioPlayer audioPlayer;

    /** @noinspection UNUSED_SYMBOL*/
    @VariationPointAnnotation(
            variationID = "sample_app1.AudioProcessorService",
            mustBeLocal = false
    )
    private AudioProcessorService audioProcessor;
```

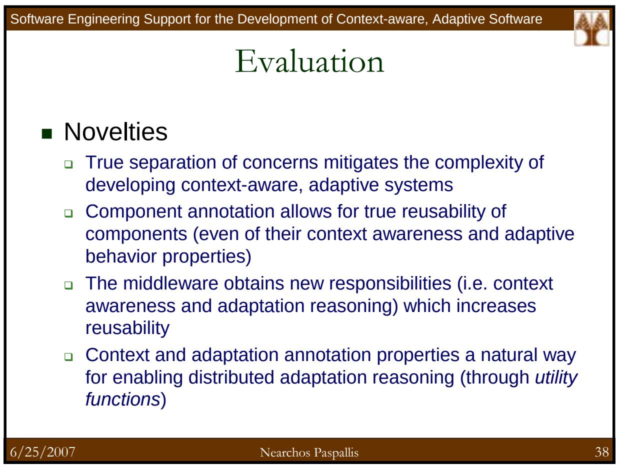


Software Engineering Support for the Development of Context-aware, Adaptive Software

Second try (proposed way) [continued]

- Deployment
  - Once deployed, the components are discovered and automatically controlled by a middleware layer, i.e. *Inversion of Control* (IoC)
  - The middleware automatically…
    - … detects which context elements can affect the deployed system (which are then automatically monitored)
    - … decides when and how should the system be adapted
    - … adjusts the system properties and optimizes the QoS delivered to the users

6/25/2007 Nearchos Paspallis 37

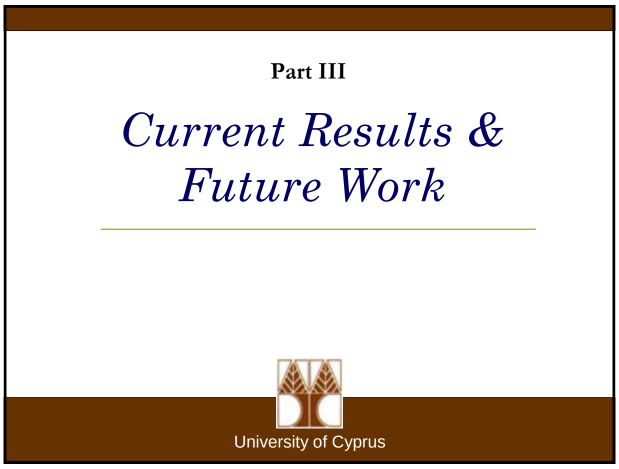


Software Engineering Support for the Development of Context-aware, Adaptive Software

Evaluation

- Novelties
  - True separation of concerns mitigates the complexity of developing context-aware, adaptive systems
  - Component annotation allows for true reusability of components (even of their context awareness and adaptive behavior properties)
  - The middleware obtains new responsibilities (i.e. context awareness and adaptation reasoning) which increases reusability
  - Context and adaptation annotation properties a natural way for enabling distributed adaptation reasoning (through *utility functions*)

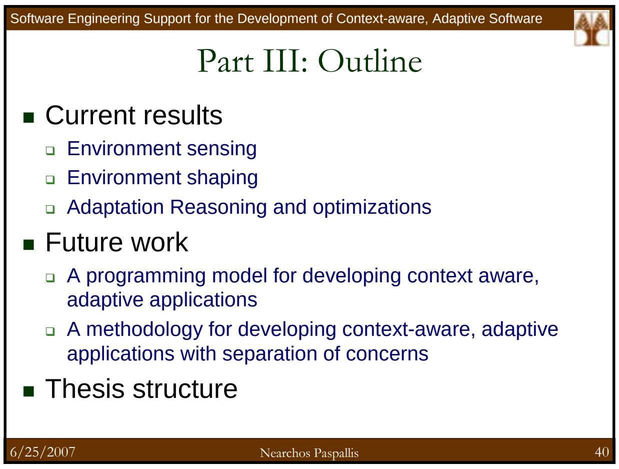
6/25/2007 Nearchos Paspallis 38



Part III

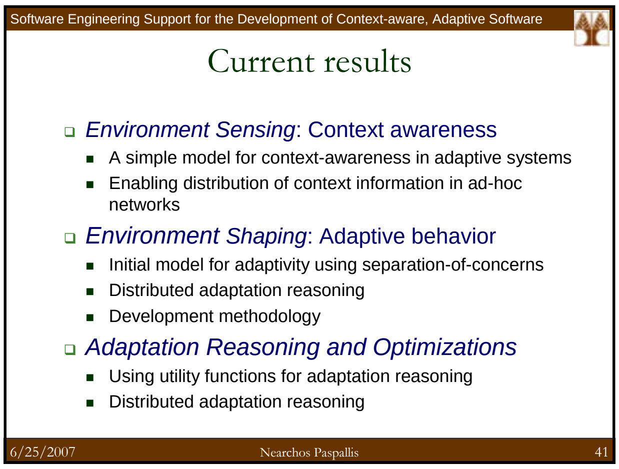
Current Results & Future Work

University of Cyprus



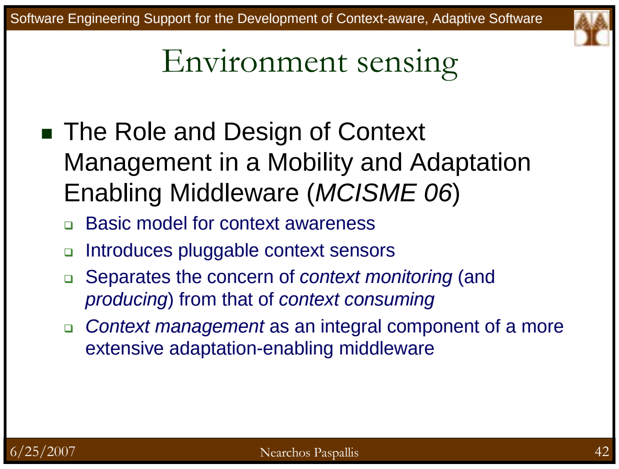
Part III: Outline

- Current results
  - Environment sensing
  - Environment shaping
  - Adaptation Reasoning and optimizations
- Future work
  - A programming model for developing context aware, adaptive applications
  - A methodology for developing context-aware, adaptive applications with separation of concerns
- Thesis structure



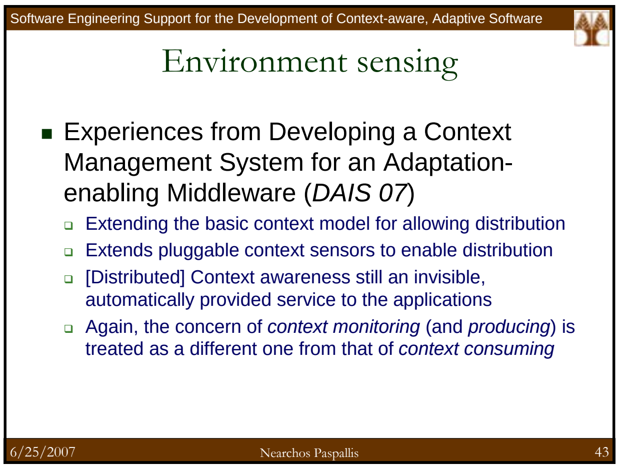
Current results

- *Environment Sensing*: Context awareness
  - A simple model for context-awareness in adaptive systems
  - Enabling distribution of context information in ad-hoc networks
- *Environment Shaping*: Adaptive behavior
  - Initial model for adaptivity using separation-of-concerns
  - Distributed adaptation reasoning
  - Development methodology
- *Adaptation Reasoning and Optimizations*
  - Using utility functions for adaptation reasoning
  - Distributed adaptation reasoning



Environment sensing

- The Role and Design of Context Management in a Mobility and Adaptation Enabling Middleware (*MCISME 06*)
  - Basic model for context awareness
  - Introduces pluggable context sensors
  - Separates the concern of *context monitoring* (and *producing*) from that of *context consuming*
  - *Context management* as an integral component of a more extensive adaptation-enabling middleware

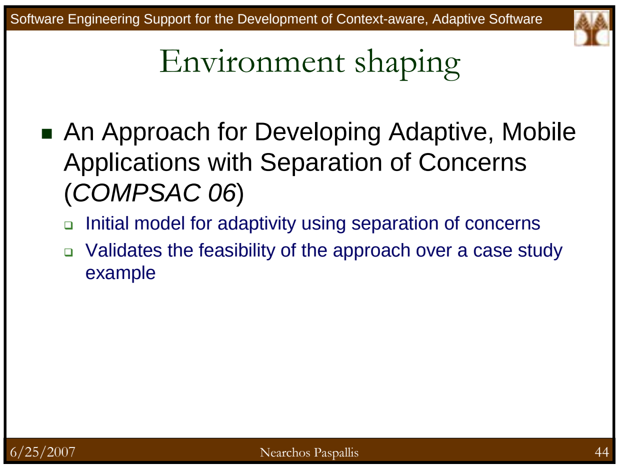


Software Engineering Support for the Development of Context-aware, Adaptive Software

Environment sensing

- Experiences from Developing a Context Management System for an Adaptation-enabling Middleware (*DAIS 07*)
  - Extending the basic context model for allowing distribution
  - Extends pluggable context sensors to enable distribution
  - [Distributed] Context awareness still an invisible, automatically provided service to the applications
  - Again, the concern of *context monitoring* (and *producing*) is treated as a different one from that of *context consuming*

6/25/2007 Nearchos Paspallis 43

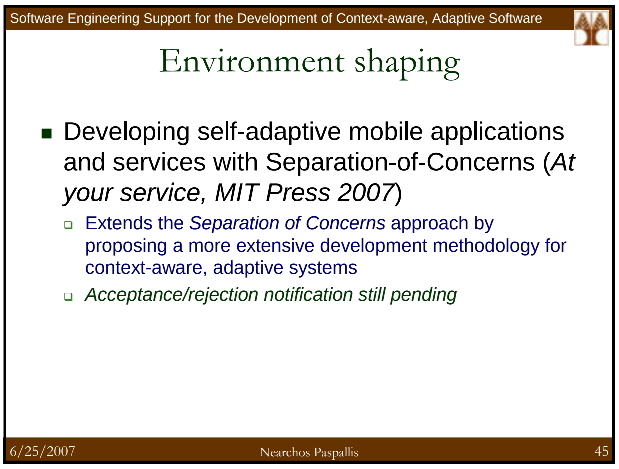


Software Engineering Support for the Development of Context-aware, Adaptive Software

Environment shaping

- An Approach for Developing Adaptive, Mobile Applications with Separation of Concerns (*COMPSAC 06*)
  - Initial model for adaptivity using separation of concerns
  - Validates the feasibility of the approach over a case study example

6/25/2007 Nearchos Paspallis 44

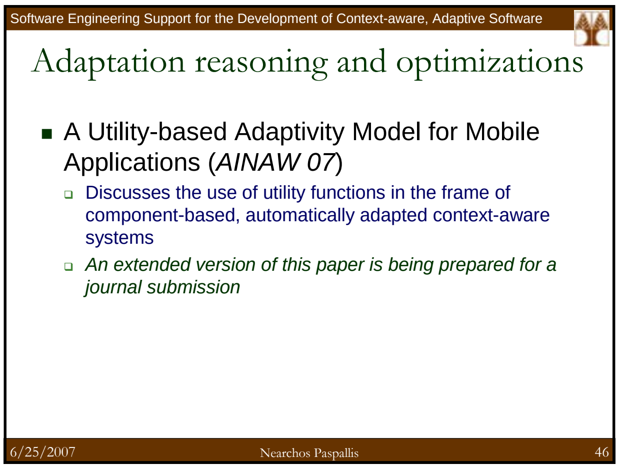


Software Engineering Support for the Development of Context-aware, Adaptive Software

Environment shaping

- Developing self-adaptive mobile applications and services with Separation-of-Concerns (*At your service, MIT Press 2007*)
  - Extends the *Separation of Concerns* approach by proposing a more extensive development methodology for context-aware, adaptive systems
  - *Acceptance/rejection notification still pending*

6/25/2007 Nearchos Paspallis 45

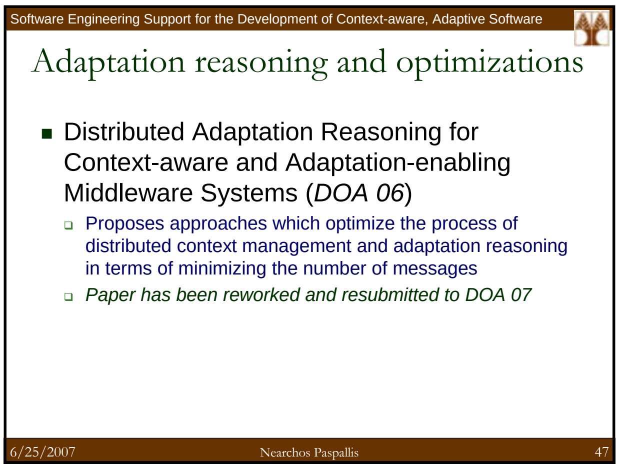


Software Engineering Support for the Development of Context-aware, Adaptive Software

Adaptation reasoning and optimizations

- A Utility-based Adaptivity Model for Mobile Applications (*AINAW 07*)
  - Discusses the use of utility functions in the frame of component-based, automatically adapted context-aware systems
  - *An extended version of this paper is being prepared for a journal submission*

6/25/2007 Nearchos Paspallis 46

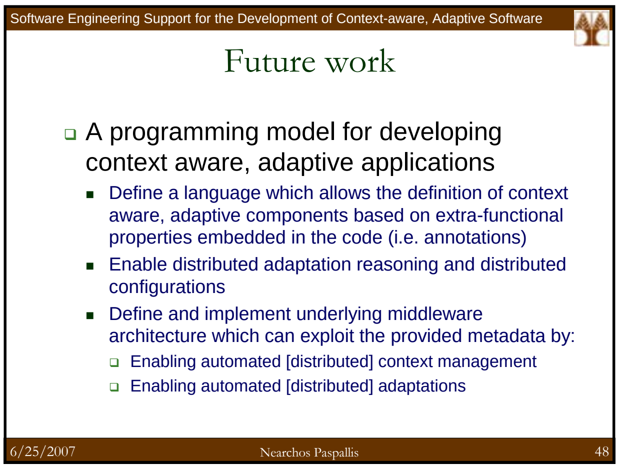


Software Engineering Support for the Development of Context-aware, Adaptive Software

Adaptation reasoning and optimizations

- Distributed Adaptation Reasoning for Context-aware and Adaptation-enabling Middleware Systems (*DOA 06*)
  - Proposes approaches which optimize the process of distributed context management and adaptation reasoning in terms of minimizing the number of messages
  - *Paper has been reworked and resubmitted to DOA 07*

6/25/2007 Nearchos Paspallis 47

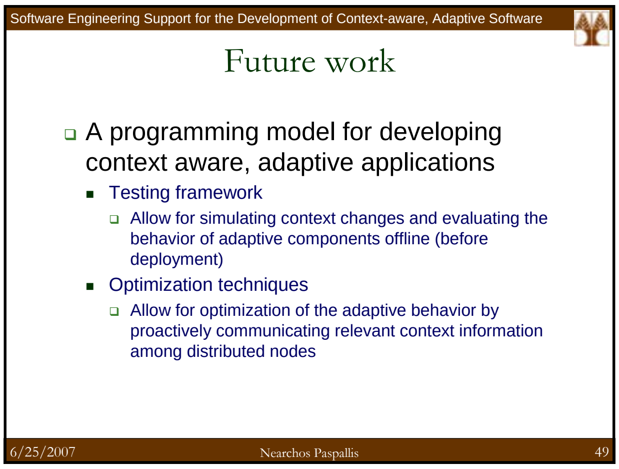


Software Engineering Support for the Development of Context-aware, Adaptive Software

Future work

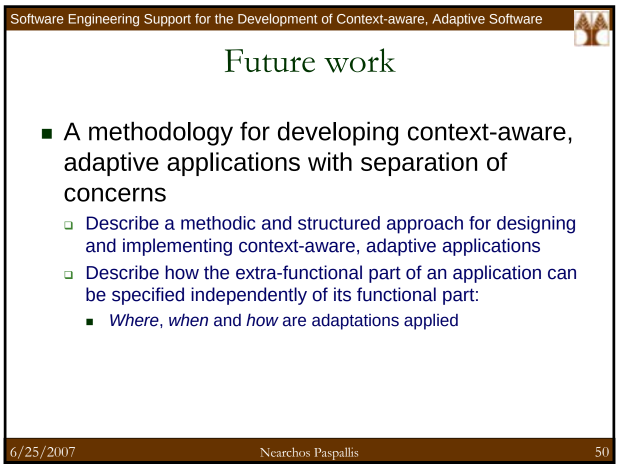
- A programming model for developing context aware, adaptive applications
  - Define a language which allows the definition of context aware, adaptive components based on extra-functional properties embedded in the code (i.e. annotations)
  - Enable distributed adaptation reasoning and distributed configurations
  - Define and implement underlying middleware architecture which can exploit the provided metadata by:
    - Enabling automated [distributed] context management
    - Enabling automated [distributed] adaptations

6/25/2007 Nearchos Paspallis 48



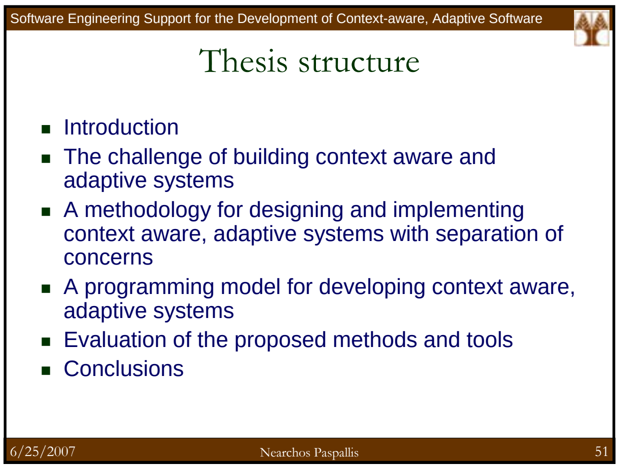
Future work

- A programming model for developing context aware, adaptive applications
  - Testing framework
    - Allow for simulating context changes and evaluating the behavior of adaptive components offline (before deployment)
  - Optimization techniques
    - Allow for optimization of the adaptive behavior by proactively communicating relevant context information among distributed nodes



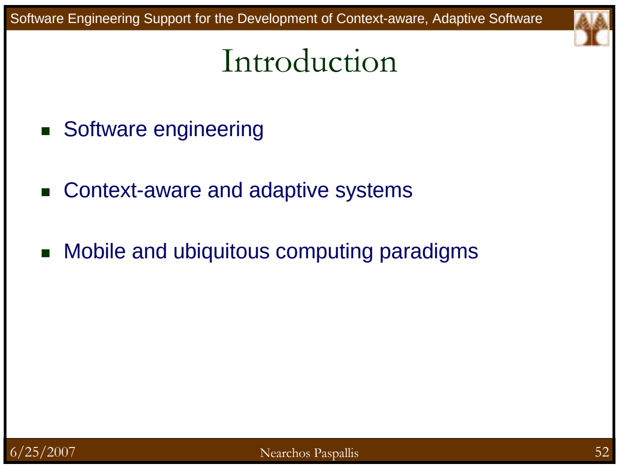
Future work

- A methodology for developing context-aware, adaptive applications with separation of concerns
  - Describe a methodic and structured approach for designing and implementing context-aware, adaptive applications
  - Describe how the extra-functional part of an application can be specified independently of its functional part:
    - *Where*, *when* and *how* are adaptations applied



Thesis structure

- Introduction
- The challenge of building context aware and adaptive systems
- A methodology for designing and implementing context aware, adaptive systems with separation of concerns
- A programming model for developing context aware, adaptive systems
- Evaluation of the proposed methods and tools
- Conclusions

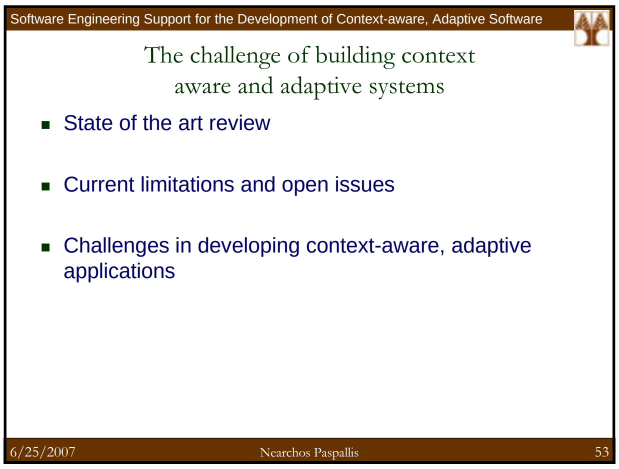


Software Engineering Support for the Development of Context-aware, Adaptive Software

Introduction

- Software engineering
- Context-aware and adaptive systems
- Mobile and ubiquitous computing paradigms

6/25/2007 Nearchos Paspallis 52

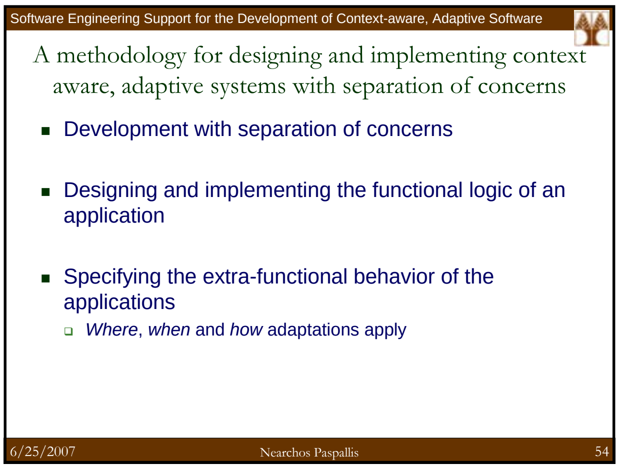


Software Engineering Support for the Development of Context-aware, Adaptive Software

The challenge of building context aware and adaptive systems

- State of the art review
- Current limitations and open issues
- Challenges in developing context-aware, adaptive applications

6/25/2007 Nearchos Paspallis 53

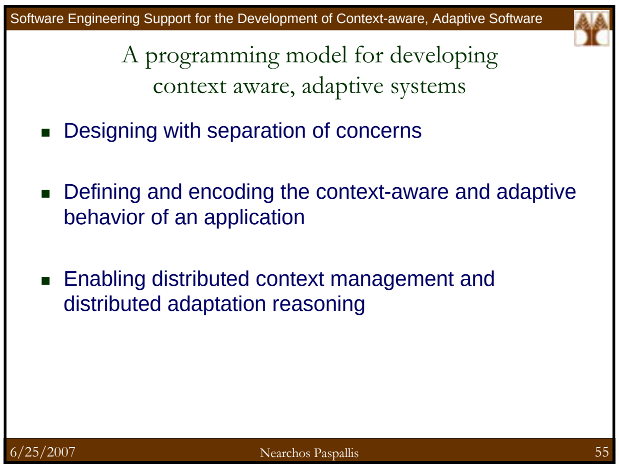


Software Engineering Support for the Development of Context-aware, Adaptive Software

A methodology for designing and implementing context aware, adaptive systems with separation of concerns

- Development with separation of concerns
- Designing and implementing the functional logic of an application
- Specifying the extra-functional behavior of the applications
  - *Where*, *when* and *how* adaptations apply

6/25/2007 Nearchos Paspallis 54

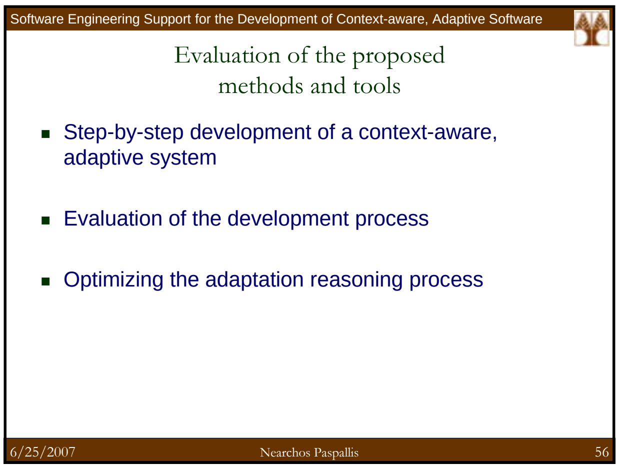


Software Engineering Support for the Development of Context-aware, Adaptive Software

A programming model for developing context aware, adaptive systems

- Designing with separation of concerns
- Defining and encoding the context-aware and adaptive behavior of an application
- Enabling distributed context management and distributed adaptation reasoning

6/25/2007 Nearchos Paspallis 55

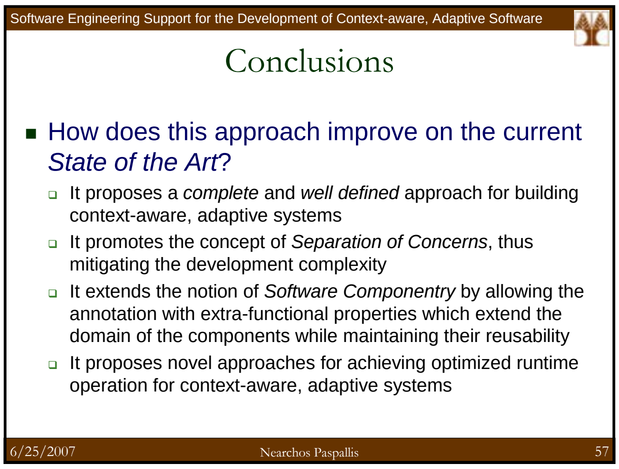


Software Engineering Support for the Development of Context-aware, Adaptive Software

Evaluation of the proposed methods and tools

- Step-by-step development of a context-aware, adaptive system
- Evaluation of the development process
- Optimizing the adaptation reasoning process

6/25/2007 Nearchos Paspallis 56

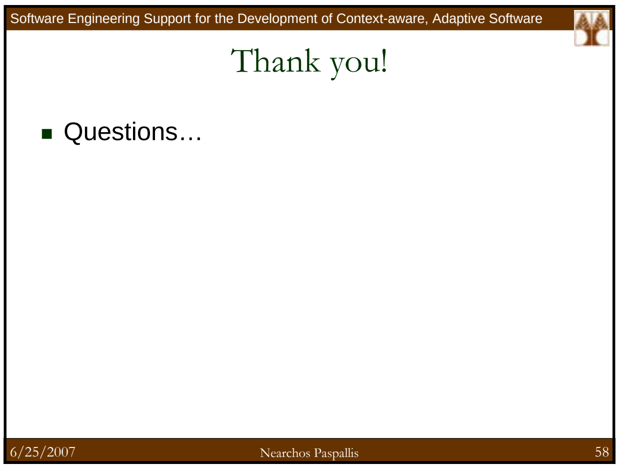


Software Engineering Support for the Development of Context-aware, Adaptive Software

Conclusions

- How does this approach improve on the current *State of the Art*?
  - It proposes a *complete* and *well defined* approach for building context-aware, adaptive systems
  - It promotes the concept of *Separation of Concerns*, thus mitigating the development complexity
  - It extends the notion of *Software Componentry* by allowing the annotation with extra-functional properties which extend the domain of the components while maintaining their reusability
  - It proposes novel approaches for achieving optimized runtime operation for context-aware, adaptive systems

6/25/2007 Nearchos Paspallis 57



Thank you!

- Questions…