Pervasive Services Engineering: from Outside to Inside

Dr. Kun Yang
University of Essex, Colchester, UK

Thursday 26th March 2009

Agenda

- Pervasive Computing
  - Fundamentals on Perv. Serv. Engineering
  - Pervasive Service Creation & Deployment
  - Service Discovery in Wireless Environment
  - Service Offloading
  - Q&A
"Ubiquitous computing" names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives." – by Mark Weiser

“Motivation: a Remark from M. Weiser

“There is more information available at our fingertips during a walk in the woods than in any computer system, yet people find a walk among trees relaxing and computers frustrating. Machines that fit the human environment, instead of forcing humans to enter theirs, will make using a computer as refreshing as taking a walk in the woods.” [M. Weiser, 1991]"
Potential Impacts

- Impacts: all areas of information technology, including:
  - hardware components (e.g. chips for smart tags),
  - networks (e.g. wireless ad-hoc networks, sensor networks),
  - software components (e.g. intelligent human-computer interaction methods like voice input),
  - novel software engineering methodologies, services, etc.
  - social concerns, e.g., the privacy of context information.

Related Fields

- Pervasive computing represents a major evolutionary step in a line of work.
- Two distinct earlier steps in this evolution are distributed systems and mobile computing.
- There are new problems introduced by pervasive computing that have no obvious mapping to problems studied earlier.

- Distributed systems:
  - Back to mid-1970s
  - Arose at the intersection of PCs and LANs
  - The results span many areas that are foundational to pervasive computing
Mobile Computing

- While many basic principles of DS design continued to apply, some constraints of mobility forced the development of specialized techniques.
- The results achieved so far can be grouped into the following broad areas:
  - Mobile networking: Mobile IP, ad hoc protocol, improving TCP performance in wireless networks
  - Mobile information access:
    - Support for adaptive applications
    - System-level energy saving techniques
    - Location sensitivity
- Mobile computing is still a very active and evolving field of research.

Taxonomy of research areas in Pervasive Computing

## Pervasive Computing

- Effective Use of Smart Spaces
  - Sensor networks
  - Embedded systems
  - Intelligent inhabitant home
- Invisibility
  - Minimal user distraction
- Localized scalability
  - As smart spaces grow in sophistication, the intensity of interactions between a user’s personal computing space and his/her surroundings increases. So try to localize the communication of mobile users.
- Masking uneven conditioning of environments
- Context-awareness

## Context & Context-awareness

- “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 applications themselves.” - by A. Dey
- context-awareness can be simply regarded as the ability of using context.
- Context-awareness should support personalized access by delivering the right information to the right place at the right time.
- It might make sense to allow systems, to certain extent, to infer a user’s intention and thus to reduce user’s distraction.
- Context awareness should enable users to take advantage of unexpected opportunities. For example, a user could be informed that he will be passing a gas station with prices significantly less expensive than their usual station.
**Category of Context**

- A categorization of context types will help application designers decide the most likely pieces of context that will be useful in their applications.
- We can use *location*, *identity*, *time*, and *activity* as basic context types for characterizing the situation of a particular context entity.

![Diagram](image)

**Other Issues**

- **User Intent:**
  - tracking, current system is bad in doing this (e.g., MS paperclip).
  - how to represent, how rich, when and how to update, how to be accessed
  - How much burden to user, is the benefit worth the cost, how to quantify?
- **Cyber Foraging**
  - Dynamically augment mobile wireless computing by exploiting wired hardware infrastructure
  - Communication with a surrogate is via short-range wireless P2P.
  - How to discover the presence of surrogates? JINI, UPnP, Etc
  - Level of trust, load balancing at surrogates, scalability, system support.
- **Adaptation**
  - User side, network side, or the mixture
Other Issues – cont.

- High-level Energy Management
  - Complex capabilities increase the energy demand of sw on a mobile
  - E.g., Energy-aware memory management. – still keep invisibility
  - Can knowledge of user intent be exploited here?

- Balancing Pro-activity and transparency
  - Depending on user’s experience: how to specify user preferences and tolerances, are they static or changing dynamically?

- Privacy and Trust
  - More complex in pervasive computing
  - Mutual trust, what is the right balance

- Other fields: HCI (Human Computer Interface), expert system, AI (artificial intelligence) (esp. in decision making and planning)

---

Agenda

- Pervasive Computing
  - Fundamentals on Perv. Serv. Engineering
  - Pervasive Service Creation & Deployment
  - Service Discovery in Wireless Environment
  - Service Offloading
  - Q&A
Pervasive Services Features

- Proliferation of wireless mobile networks and devices
- The future success of these mobile systems lie in its ability to provide users with cost-effective services that have the potential to run anywhere, anytime and on any device without (or with little) user attention.
- Services of these features are usually termed as pervasive services (PS).
- Network-aware services
- Single vs. composite service

Lifecycle of Pervasive Service Eng.

- **Service creation** means the design and implementation of a service at compile time and not at the runtime, i.e. to generate the executable code for this service.
- After a service is created, it is deployed to the service user side (and probably to the supporting middleware if some special functions are needed) - and this procedure is usually called **service deployment**.
- **Service composition** refers to a dynamic procedure that composes a new service (or precisely a new part of a running service) from the components that are selected at run time.
- If the components needed by the new requirements of the service are not available at the design time then these components will have to be found at run time – and this is the function of **service discovery**.
- **Service adaptation**, while implemented by certain mechanism, usually describes the most important feature of pervasive services, i.e., the ability to adapt itself to the change of the environment (or context) in a dynamic manner.
- **Service management** usually means a collection of some or all of the above procedures.
PSE Requirements

- Service adaptability & Service mobility.
- Such adaptability requires a shift of service development from large, monolithic applications with fixed user interface mode and task logic (no matter how comprehensive it may be) to a new methodology where the service logic is constructed on the fly from collections of loosely coupled small components as driven by its context changes.
- This is because in a mobile environment it is hard for service developers to know at the service development stage which kind of environment the service user will be in and when.
- In other words, a pervasive service should grant its users as much flexibility as possible.

Current Approach to Provide Service Adaptability

- Most researches are concerned with a powerful middleware infrastructure sensing the interesting contextual changes and adapting accordingly, such as recent One.world and less recent Oxygen project and the Portolano Project.
- Usually, these middleware and their underlying context-gathering mechanisms are proprietary, making the services offered by different service providers hardly interoperate, and consequently leading to duplicated development of software component (and sometimes hardware such as sensors).
A New Angle: from outside to inside

While the necessity of the presence of a middleware infrastructure is appreciated, this lecture presents another way of tackling the adaptability requirement of pervasive services from another angle, i.e., the internal logic of pervasive service itself, which is largely neglected in the related research work.

The fundamental concept is to grant the adaptability to services themselves rather than their execution environment

- aiming to leave the latter as clean and simple as possible (ideally only standard JVM – Java Virtual Machine, plus some Java-based service discovery mechanism).
- It is believed that pervasiveness comes from simplicity. A simpler environment stands a much better chance of being more widely deployed and more accommodating.

The next question then is how to grant the high-level adaptability (or intelligence as it can be regarded as) to the service logic.

Technical Motivation

Technical core:
- to utilize policies (in the form of rule: IF <conditions> THEN <actions>) to describe the pervasive service logic at a higher level and
- to use models to enhance the information model (providing implementation) of the pervasive service.

Policy employed here is a concept derived from Policy-based Management (PBM), a successful technique widely researched for distributed systems and IP network management. A policy is employed to define a choice in the behaviour of a pervasive service.

Models utilized here specifically refer to a modern software engineering method led by OMG (Object Management Group) Model-Driven Architecture (MDA).
- platform-, language- and middleware-neutral features
Policy-based Management (PBM)

- The concept of PBM is not new. PBM reference architectures that cleanly isolate management logic from physical management operation have made their appearance in both the distributed system community and the network community (e.g., IETF Policy WG) a decade ago. However, the current PBM solutions are still not appropriate for service oriented computing though researchers have started to evolve PBM towards this direction.
- IETF’s policy information model, but no reference implementation so far.
- Applying MDA for policy information model can be an attempt towards a more flexible way of defining and implementing policy information model.

Model-driven Architecture (MDA)

- Its essence is the concept of separation between the specification of a system’s essential functions as Platform Independent Models (PIM) and the realisation of the system using more specific and detailed platform as Platform Specific Models (PSM).

- MDA supporting software or platform: open source (Eclipse), or commercial product.
**MDA – Behind the Scene**

**MOF**: Meta-Object Facility

---

**UML**: Unified Modelling Language; **DSL**: Domain-specific Language
**QVT**: (Query/View/Transformation); **EJB**: Enterprise Java Bean

---

**Essence #1: Pervasive Service as a PBM System**

- A pervasive service itself, as a PBM system, is a collection of the following components:
  - **policies** that define the service behaviours or guidance under different conditions and are defined by service developers;
  - a **service decision engine** that reasons over these policies (called the Policy Decision Point or PDP in IETF terminology); and
  - a **service execution engine** (called Policy Enforcement Point or PEP) that takes real action as a result of service decision making.

---

**MDA-enabled**
Essence #2: MDA Models as the Base of Pervasive Service Information Model (PSIM)

PS Information Model


MDA Components

Platform Independent Model (PIM)  Platform Specific Model (PSM)  Code

Proposed Techniques

XMF Toolkit + PoPSiDL  XMF toolkit  Java

Essence #3: Model-based Service Discovery for Wireless Mobile Networks

- Here “service discovery” actually means the discovery of a service component that is needed to by a running pervasive service to deal with certain service requirement as a result of context changes.
- We will investigate the service discovery mechanisms that are not dependent on a centralized server but rather are carried out in an ad hoc manner, e.g., to discover a service component available in its nearby mobile devices.
- The service discovery algorithm to be investigated reuses the models that are generated during static service creation/development stage.
An example PSE System Architecture

Service Adaptation Strategy

Service adaptation is to be carried out by two complementary mechanisms:

- **At the service creation stage**, policies are employed to describe the service behaviours (including user preferences about the service) in a user-friendly and non-technical manner, and models of different abstraction level are used to describe the implementation of the service in a platform- and technology-neutral way;

- **At the service execution stage**, dynamic service discovery and composition are in place to instantiate the service components missing at the service creation stage.
Agenda

- Pervasive Computing
- Fundamentals on Perv. Serv. Engineering
- Pervasive Service Creation & Deployment
- Service Discovery in Wireless Environment
- Service Offloading
- Q&A

Steps for Service Creation

1. Use MDA tools such as XMF to create these core and unique components that are needed by the pervasive service but are not currently available to the service provider. The components that the service developer believes can be found dynamically are left un-implemented. However these un-implemented components are each given an enabler to discover the components during service runtime. This enabler is the Service Discovery Client Side.

2. Use the above service specification tool to define the service behaviours using policies.

3. Then invoke the “Service Creation Module” to generate the pervasive service in Java language.

Pervasive Service Description Language

- Policy-based Pervasive Service Description Language: PoPSiDL
- to specify/update the possible behaviours (actions) of a pervasive service under different circumstances (conditions) by using policies.
- policies take the following rule-based format:
  - IF [condition(s)] THEN [action(s)].
  - It means action(s) is/are taken if the corresponding condition(s) is/are true.
- For instance, the following policy in a pervasive service forces the mobile to vibrate instead of beeping during meeting time:
  - IF (location == meetingRoom) and (time within meetingSchedule) THEN MobileVibratingOnly
- The PoPSiDL parser is created in a semi-automated manner using JavaCC.
Pervasive Service Analysis and Modelling

- In summary, the following six generic entities are usually involved in a typical pervasive service: user, user device, function, function provider, access network, and core network,
  - each having, apart from their own attributes, four extra attributes, namely, identity, time, location and activity.
- These artefacts constitute the meta-model of a specific domain: pervasive service, as depicted by XMF in next slide.
- This meta-model is called the domain specific language or DSL in MDA terminology.

---

Domain Specific Language (or Meta-model) for Pervasive Services
Pervasive Service Information Model

- After the modelling language and the description language of pervasive services have been defined in the previous two slides, a pervasive service’s high-level behaviours can be specified by a PoPSiDL script.
- The next step is how to execute the script, i.e., to implement the pervasive service. This is how the information model for this pervasive service (i.e., PSIM) comes into place.
- Since PIM is application specific, here we utilize the SNAMU service as an example to demonstrate how the PIM part of a PSIM is created.

A Pervasive Service Scenario

- An imaginary future pervasive service called SNAMU (transparent Service & Network Access for Mobile User) – see the next two slides
- The service tends to provide a means for a nomadic user to maintain a secure access to his/her enterprise/home network transparently after the user has subscribed to the pervasive service from the service provider.
- SNAMU also allows its user to dynamically search for new services and to use services in a context-aware manner.
SNAMU illustration

Part of the PIM of SNAMU
Transformation of SNAMU PIM to PSM/Java

- A MicroJava PSM and its corresponding transformation to MicroJava (a micro version of the Java programming language) have been implemented in XMF.
- Each element in the SNAMU PIM is a sub-class of XCore::Class. As every element in the XCore package has a mapping to a corresponding element in the MicroJava PSM, then the mapping from SNAMU PIM to MicroJava PSM is straightforward.
- One of the mechanisms provided by the XMF tool to transform PIM to PSM/MicroJava is XMap. The diagram in next slide illustrates how XMap was used to provide this transformation by means of mappings.
- A mapping consists of a sequence of clauses describing pattern matches. These clauses take an object with a specific structure and match it with another object of another structure.
Policy-based PS Decision Making

- So far the source code of classes that are locally available to SNAMU service has been generated.
- These classes constitute the building blocks for creating pervasive services.
- However, before discussing how a pervasive service is created, another important component of pervasive services, namely the service decision making engine needs to be designed.

A GUI for Pervasive Service Creation

1. GUI for service creation
2. Policy-based rules
3. Service information
4. Decision-making interface
5. Implementation details
### PS Execution Environment

- **SNAMU**'s execution is event-based and usually follows the following procedure:
  - the change of the context information (and in turn the conditions of certain **SNAMU** policies) generates certain event which triggers the reasoning of the service decision engine over the **SNAMU** policies;
  - then certain action(s), or no action if none of the conditions of the policies is satisfied, will be taken.

![Pervasive Service Execution Environment](image)

### Agenda

- Pervasive Computing
- Fundamentals on Perv. Serv. Engineering
- Pervasive Service Creation & Deployment
- **Service Discovery in Wireless Environment**
- Service Offloading
- Q&A
Service Discovery Overview

Service discovery is an essential mechanism broadly utilized by researchers and competing industries to pursue dynamic and automatic software system composition, configuration and adaptation.

However, the current situation of service discovery to some extent hinders the service supporting systems from pursuing their maximum effectiveness.

- various service description languages, thus limiting the scope of global service discovery
- they are primarily bound to the architecture of the individual discovery mechanism
- all fail to provide incentive for service composition

Discovered service components as a result of any service discovery will typically be used to compose new services (including integrating with the current user services). Therefore, model-based service discovery (MBSD) can be a good alternative.

Current Service Discovery Mechanisms

For wired networks:

- Jini: a Java-based technology introduced by the Sun Microsystems.
  - The service discovery is based on unicast after knowing the presence of service brokers.
  - Jini technology uses the Java RMI (Remote Method Invocation) to provide service delivery (i.e., moving code around the network).
- UPnP (Universal Plug and Play): the UPnP Forum, headed by Microsoft, is in charge of the standard’s developments.
  - UPnP involves advertisement, discovery, and control of networked devices, services.
  - Service description in XML and the control messages are expressed as a collection of SOAP (Simple Object Access Protocol) objects.

For wireless Networks

- DEAPspace: developed by IBM, evolves the service discovery from wired networks to wireless networks (i.e., single-hop mobile ad hoc networks).
- Konark: furthers DEAPspace to multi-hop wireless ad hoc networks.
Model-based SD: Methodology

- The most salient point of this methodology is the fact that it reuses the models that are generated during static service creation/development stage and makes them available to the dynamic stages of the service lifecycle such as service discovery and service re-composition.
- This is reflected in MBSD from the following two main aspects.
  - Firstly, rather than manually creating service description as it is in almost all service discovery mechanisms, the XML (eXtensible Mark-up Language) -based service description in MBSD can be automatically generated from these models and as such providing a more precise description of services.
  - Secondly, the model information carried in MBSD service description is used to regenerate original models that can be used, either entirely or partially, by other parts of mobile applications.

Model-based Service Discovery - MBSD

- MBSD adopts a fully distributed architecture over multi-hop wireless mobile networks, i.e., each mobile node is a service provider and is its own service broker.
- XML is employed by MBSD to describe the service. However, rather than creating these description manually, MBSD makes re-use of the service information already existing in the service models.
- Furthermore, the content structure of its service registry is well organized to reflect the domain-specific feature of mobile applications.
Service Registry

- Service information in service registry should be stored in a manageable (e.g., well classified) and user-understandable manner.
- MBSD adopts a tree-based service registry where more generic service classifications reside at the higher levels of the tree and service classifications become more specific as the tree levels go down.
- The registry on each service provider can start from different levels but as a branch of a generic mobile service tree.

![Service Registry Diagram]

Agenda

- Pervasive Computing
- Fundamentals on Perv. Serv. Engineering
- Pervasive Service Creation & Deployment
- Service Discovery in Wireless Environment
- Service Offloading
- Q&A

**Service offloading in Wireless Environment**

- Offloading service can seamlessly offload part of a PS’s tasks from a mobile host to some nearby resource-rich PCs (called surrogates).
- Service components are instrumented on-the-fly either statically or dynamically.
- A PS is partitioned into one local partition (to be executed in the mobile host) and one or more remote partitions (for remote execution in surrogates).
- Surrogates are discovered by using Combined Routing and Surrogate Selection (CRoSS) algorithm.
- Offloading decisions are made dynamically adapting to real mobile computing environments.
- By using offloading services, the execution time of PSs can be shortened and the energy (e.g., battery time) in the mobile devices can be preserved.

---

**Proposed System Architecture for Service Offloading**

![Diagram of proposed system architecture for service offloading.]
Case Study: Language Translation

Contact, Q&A

Dr Kun Yang
Reader of Pervasive Networks and Services
School of Computer Science & Elec. Eng. (CSEE),
University of Essex, Wivenhoe Park, Colchester,
CO4 3SQ, UK
- Email: kunyang@essex.ac.uk
- http://privatewww.essex.ac.uk/~kunyang/