Context-Aware Computing

Ramakrishna Padmanabhan
February 12, 2013
Overall Theme

- What kinds of applications can we create that make good use of context?
- How can we obtain location?
- What kind of software engineering framework do we need to build these applications?
Context-Aware Computing Applications
Context-Aware Computing Software

- **Definition** - S/w that exploits changing environment by taking context into account

- **Categories**
  - Example applications
  - Some challenges
Proximate Selection

- Emphasize nearby objects

- Applications -
  - Select nearby i/p or o/p devices
  - "Beam" a document to a person
  - Find out about nearby restaurants

- Challenges -
  - Display alphabetical and proximity info
  - How often should location be updated?
Automatic Contextual Reconfiguration

- When context changes, adapt.

- Applications -
  - Multi-user drawing program with virtual whiteboard. Context can be per-room, or per-set_of_people

- Challenges -
  - Rapid change in context -> frequent reconfiguration
Contextual Information and Commands

- Interpret commands based upon context

- Applications -
  - Print to nearest printer
  - Display files based upon users' locations
    - Only if users are in specific locations can they see files

- Challenges -
  - Security and Authentication
Context-triggered Actions

- Actions that take place automatically when context changes

- Applications -
  - Play song as soon as I walk into my office
  - Flexible Reminders

- Challenges -
  - Tradeoff between timely execution and predictable behavior
Strengths and Weaknesses

- Pioneering work in context-aware computing
  - PARCTAB is like an IPad!
  - Good example applications
  - Location services with beacons
- Paper briefly mentions migration of computing...
  - Do processes migrate from place to place?
  - Do files migrate? If so, how?
- Cloud and social networks - how have they changed things?
WhereWare
Locate people and objects

- Explores technical, business and societal considerations
- What are the different ways of locating things?
- What are the services that get enabled as a result?
- What obstacles need to be overcome?
What technologies are available?

- **Outdoors**
  - **GPS**
    - Need to calculate position by contacting 3 out of 24 possible GPS satellites
    - Requires line of sight to satellite
    - Calculation on end device. Accuracy: \(~8m\)
  - **Cellular**
    - Basic cell based approach? Low accuracy
    - Time difference of arrival with multiple BS
    - Accuracy: \(~120m\)
  - **Assisted GPS**
    - Cell towers help device communicate with satellite
What technologies are available?

- Ceilings, walls and floors interfere with accuracy of GPS and cellular.
- Indoors
  - Wi-Fi
    - Use signal strengths to find position
    - Solves up and down problem
    - Accuracy: 1 to 5m
    - Problems: Signal Interference and Security
  - Ultrawideband
    - Tiny energy bursts over large frequency spectrum
    - Still in its infancy
Problems to be solved

- Need high accuracy, low power consumption and low price

- Business Considerations
  - Companies may not share their services with each other

- Privacy
  - Even for 'honest' people, is it always Ok for others to know where they are?
A Software Engineering Framework for Context-Aware Pervasive Computing
Problems with context-aware applications

- Context-aware applications not in the market yet
  - High application development overheads
  - Privacy and usability concerns
  - Need to identify compelling uses of context-awareness

- Problems with existing models
  - Not formal enough or expressive enough
  - Don't handle imprecision, user preferences
How should Context be Modelled?

- Design a formal, expressive context model - CML
  - Four types of info: sensed, static, user-supplied and derived
  - Design a context model using fact types and relationships
  - Translate context model into relational model that can be queried
  - Handle unknowns and ambiguity
  - Take into account user preferences and assign them scores and ranks
Abstractions for Context-Aware Programming

● Situation Abstraction
  ○ Situations can be thought of as predicates built using relations from the CML
  ○ Situations can be combined
    ■ Increased reuse
    ■ Complex situations can be formed

● Branching
  ○ Provide abstraction to prevent code-change as context evolves

● Triggering
  ○ Asynchronous actions as context changes
Strengths and Weaknesses

- Worthy goals
  - Practical approach to a broad problem
  - Providing formal methods to express context
  - Take into account imprecision and ambiguity

- Is the emphasis on user preferences too high...?
  - How often do users customize their apps?
Thank you!