Dude, Where's My Car?
And Other Questions in Context-Awareness

Jason I. Hong
James A. Landay

Group for User Interface Research
University of California at Berkeley

The Context Fabric:
Infrastructure Support for Context-Aware Computing

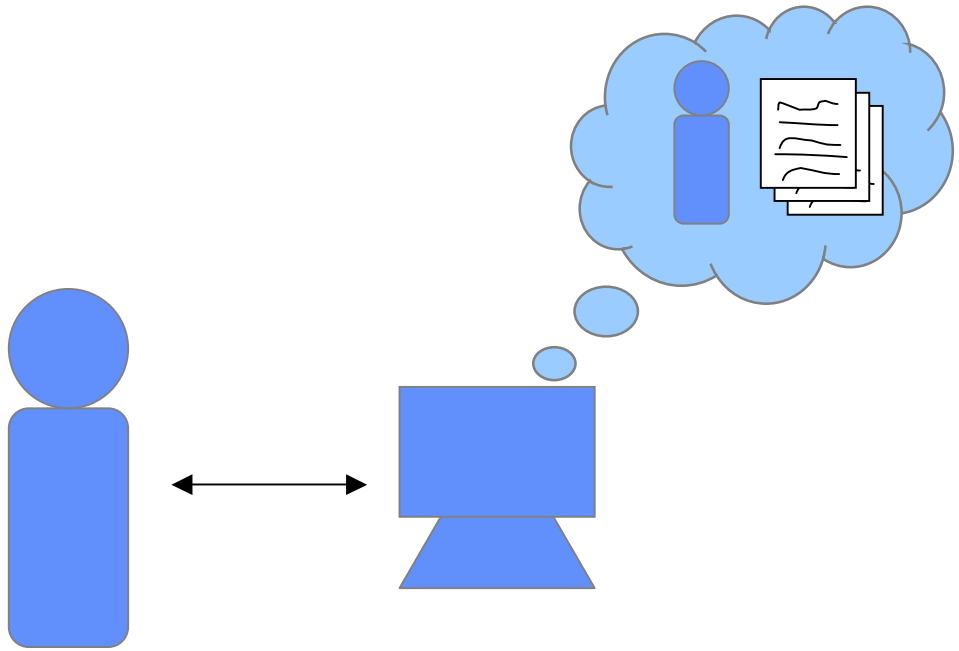
Jason I. Hong
James A. Landay

Group for User Interface Research
University of California at Berkeley

What's Context?

- Context?
  - Situated Action, Activity Theory, Distributed Cognition, Linguistics, Embodiment
- Computational view
  - Increase input channels into computer
  - Push towards implicit acquisition
  - Create better models to take advantage of input
  - Using the input + models in useful ways
- Focus is on physical world, distributed, implicitly acquired context

Motivation



We sort of know how to handle context-awareness here

GUIR

Motivation

GUIR

Motivation

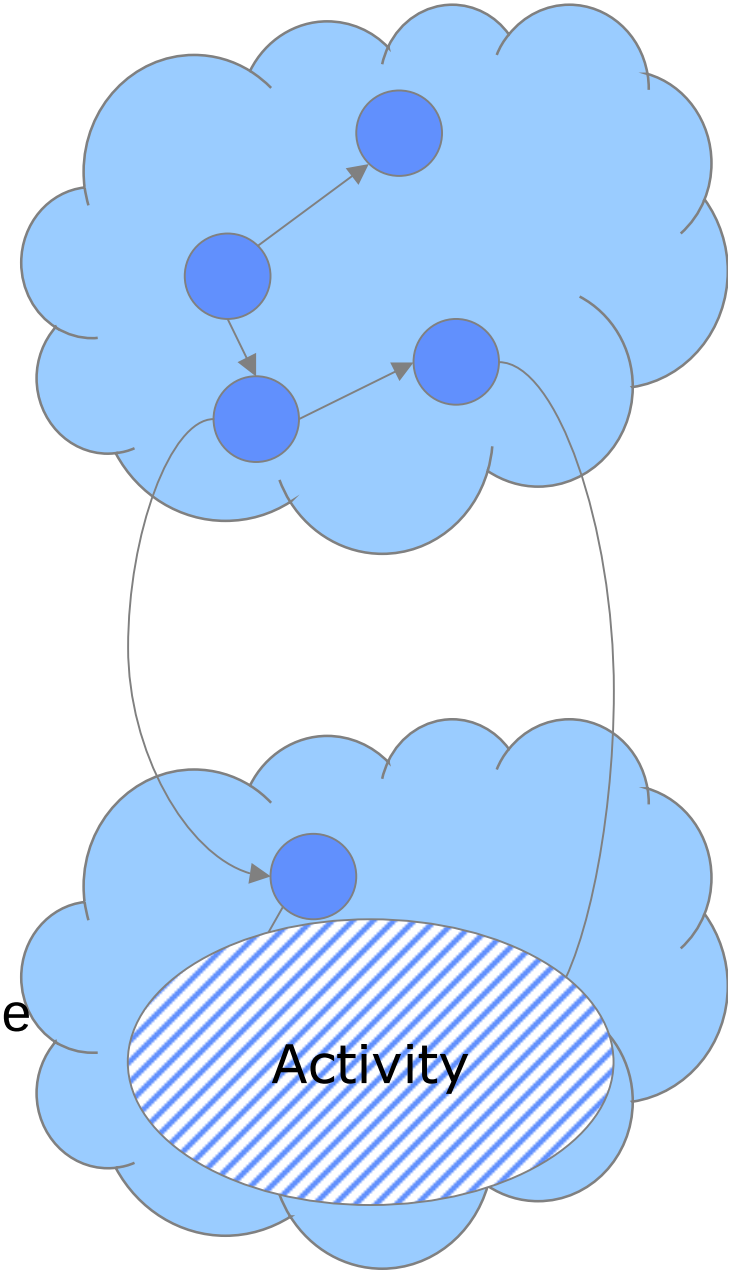
- Still really hard to build context-aware apps!
  - Same context data can come from many sources
  - Context data is highly distributed (emergent)
  - Need more expressive data models
  - These context models have not addressed security and privacy concerns
  - Difficult to program applications in an environment that is constantly changing in terms of sensors, services, and context data
- Context Toolkit
  - Operating system view – abstraction to hardware, programming it
  - Database view – how data is modeled, distributed, protected, and used

Proposed Solution

- Infrastructure approach
  - TCP / IP analogy
  - Data formats and network protocols, services out there that you can rely on being there
  - Be agnostic of sensor, CPU, OS, programming lang, network, discovery service, service platform
  - Have to be able to evolve and incrementally deploy
- Three things
  - Distributed data model of people, places, things
  - Context Specification Language (like SQL)
  - Context service as API into this all (per device)

Data Model

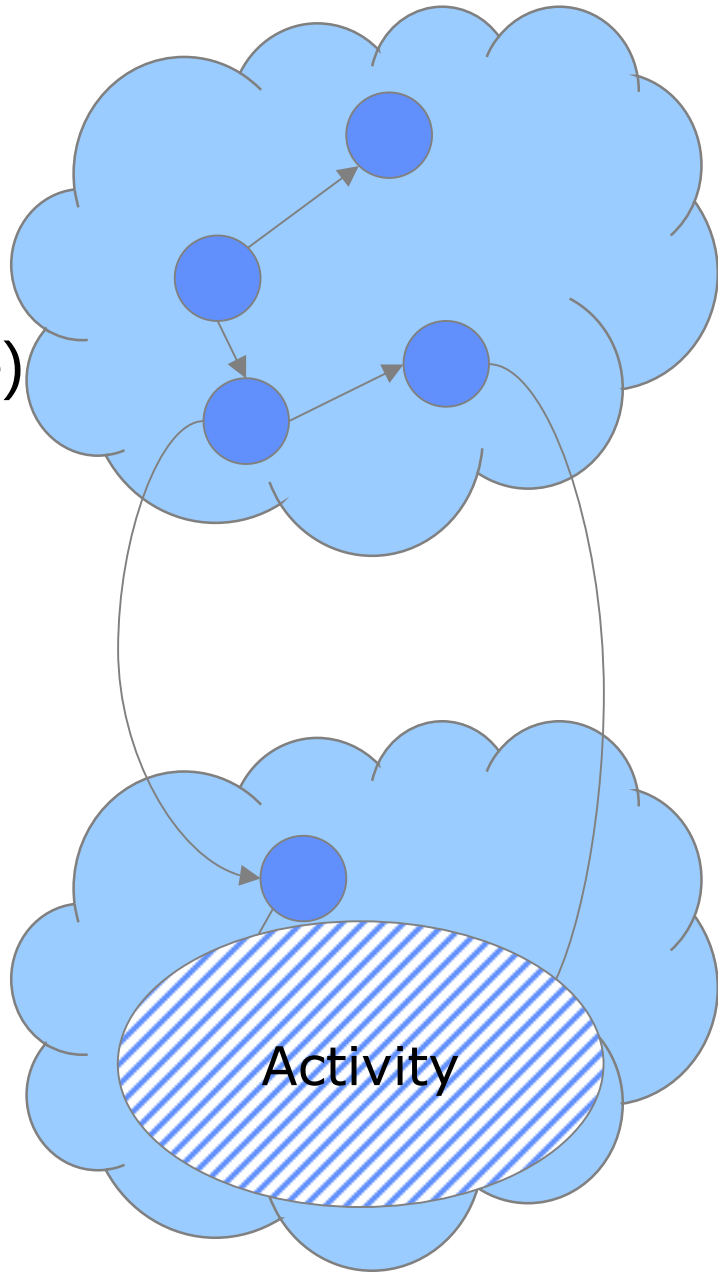
- Three questions:
  - How is context data represented?
  - Where does it live?
  - How is it shared?
- Semi-structured data
  - No universal ontology, situation dependant
  - Support multiple schemas simultaneously
- Multiple tuple-spaces
  - Each device has a space (no servers)
  - CAP theorem, weak consistency
  - Local / Private data goes to your space
  - Different levels of trust for each
  - Multiple views of context
    - Mine, yours, theirs



GUIR

Data Model

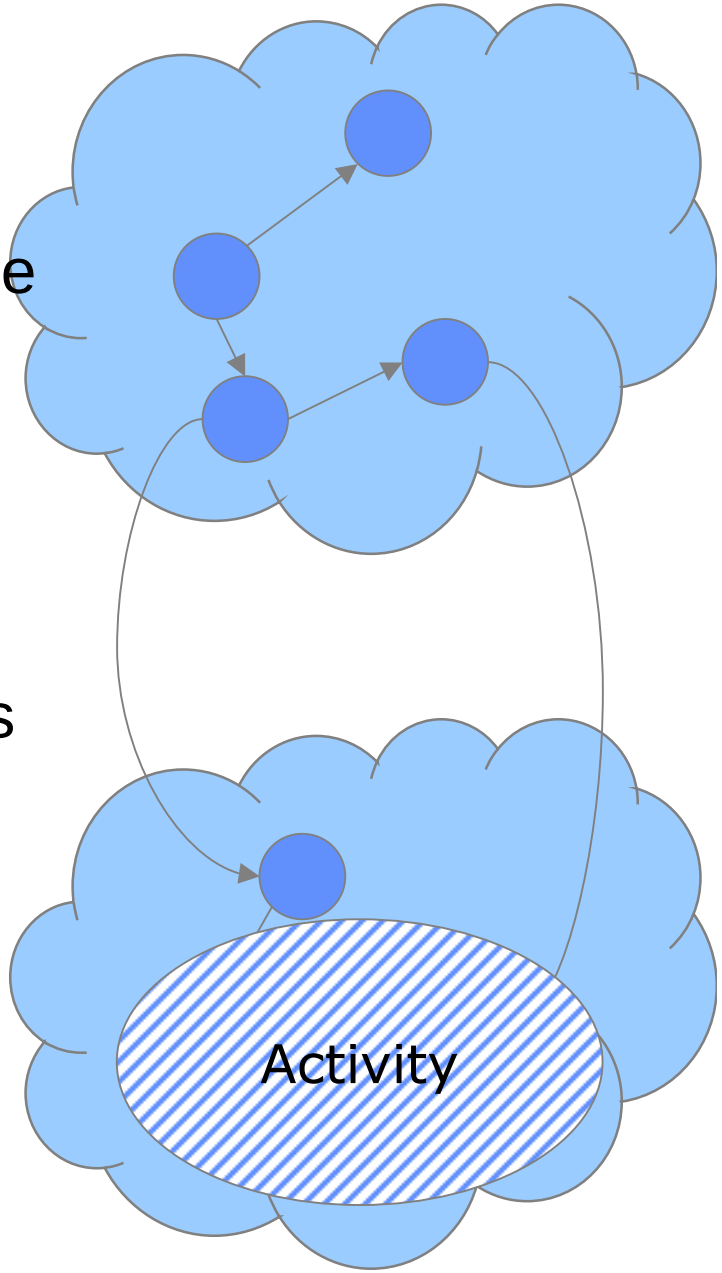
- Entities
  - People, places, things
  - (Haven't figured out space, time)
  - Certificates, Access control, Views on data
- Attributes
  - Tons of metadata
  - Source? Trusted? Age? Delta? Precision? Accuracy?
- Relationships
- Aggregates
  - Indexes, Active Maps, Action, Workgroup, Histories



GUIR

Data Model

- Advantages
  - Separates acquisition, model, usage
  - More resilient to failure
  - Multiple schemas provide flexibility
  - Context data lives separately from process, application, device
  - Templates for basic privacy policies
    - Family, friends, co-workers, strangers



GUIR

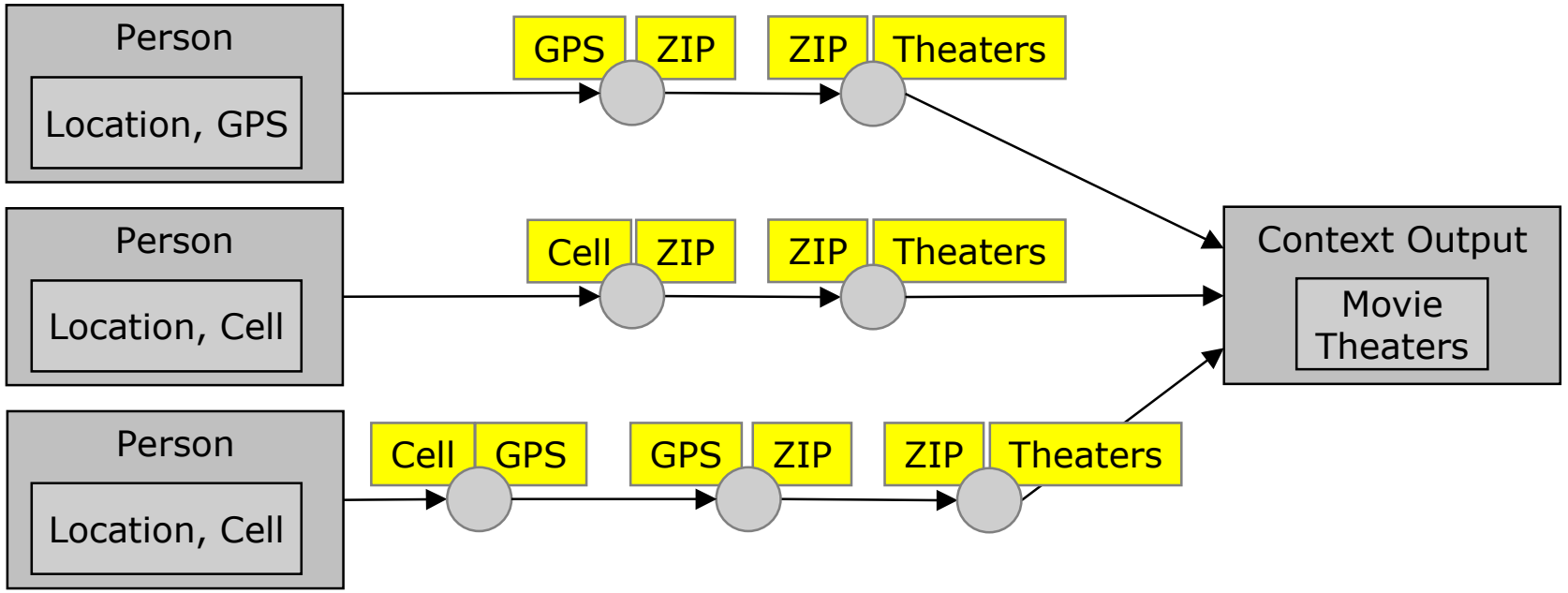
Context Specification Lang

- Problem: Difficult to coordinate data and services to get the right context data procedurally
- Idea: Declaratively specify what you need
- Query
  - "What are the nearby movie theaters?"
  - “How many people are in the room right now?”
- Events
  - “Notify me every time a person enters the room.”
  - (Like programming the physical world)
- Still vague, still in progress
  - Don't want to solve Natural Language Problem!
  - Basic templates for common types of queries
    - "What are the nearest X?" "Where is Y?"

GUIR

Context Service

What are the nearby movie theaters?



- Interpreters
- Data type transducers
- Fusers
- Filters
- Introspection: What's going on?

GUIR

Speakeasy:
Supporting the Ubiquitous Computing User Experience

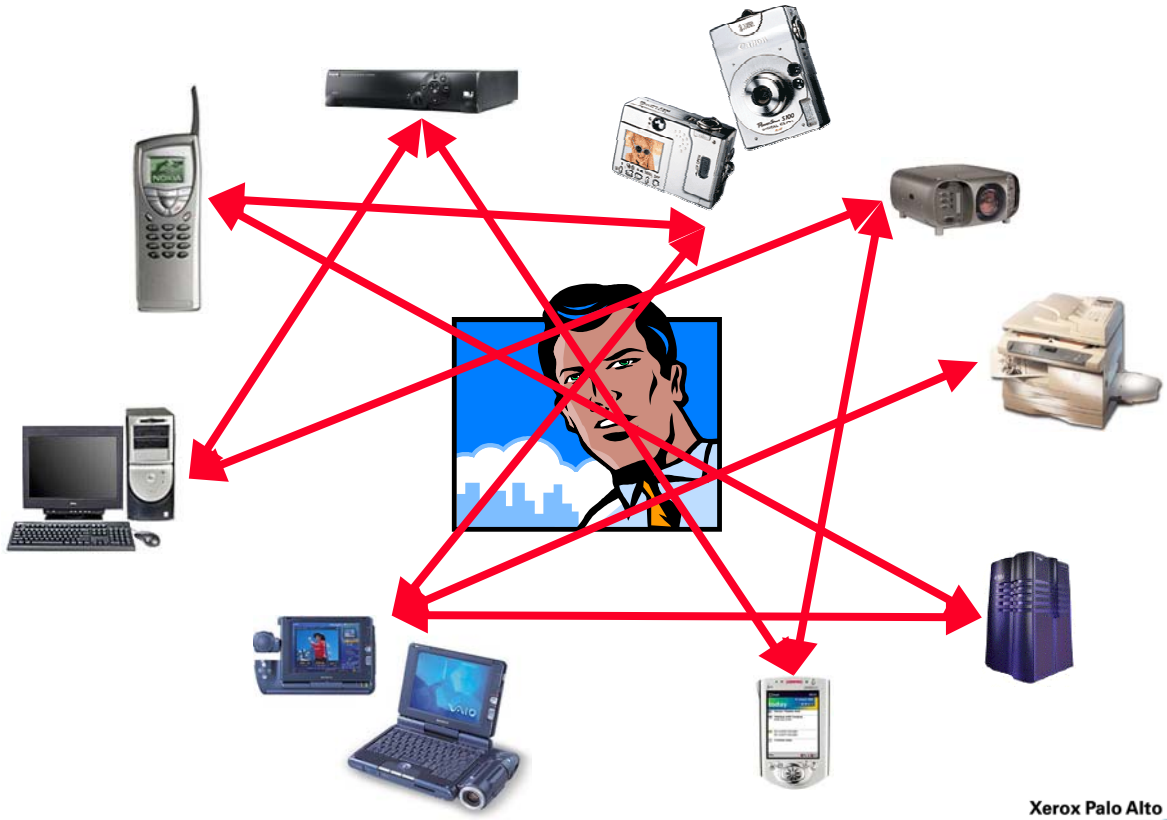
Mark Newman, Keith Edwards, Jana Sedivy, Chris Neuwirth, Karen Marcelo, Trevor Smith, Jason Hong

Xerox Palo Alto Research Center
parc

Motivation

The era of ubiquitous computing is upon us

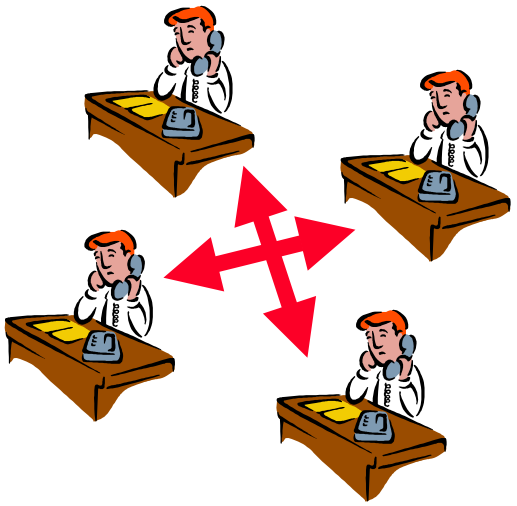
– many devices per person, becoming interconnected



The Speakeasy Vision

Enable Network Effects

– analogous to phone, fax, web…



Radical Interoperability

– what if anything can talk to anything?
– every new device or service adds value

Deal with Complexity

– support users' sensemaking
– what can I do in this world?
– what the heck is going on?

Interoperability

No need to write specifically for a new component

- Interact with components you've never heard of
  - Interact with *types* of components you've never heard of

Our approach: mobile code + standard interfaces

- (+ discovery + shared network)

Identify the *minimal set of interfaces.* So far…

- Data transfer & transformation
- Status & notification
- …
- Context
- User interface

User Experience

Context

- Modeled as People, Places, Things (Components)
- People
  - What components have I used before?
  - What components belong to me?
- Places
  - What components are in this place?
- Components
  - Where am I?
  - What can I do?
  - How have other people used me?

Key Idea

- Provide key information to people, not infer