Wearable Computing

Holger Kenn

Universität Bremen

WS 05/06

The Context Toolkit

Context

- Context
  - Sources
  - Handling Context
  - Using Context

The Context Toolkit

- Middleware for context processing
- Designed by Dey and Abowd
- Idea: Instead of redesigning context-aware applications from scratch, build them with a reusable middleware
- Uses GUI-Like metaphor: Context Widgets

Context according to Dey & Abowd

Context: any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects.

Dey & Abowd, Towards a better understanding of context and context- awareness, 2000

Context

- Entities: Places, People and Things
- Categories: Identity, location, status (or activity), time
- Application uses *context-aware functions* related to
  1. Presentation of information
  2. Execution of services
  3. Storage of context-information (attached to other stored items)

Examples for functions presenting information

- Map display of surroundings and points of interest (aka Location-based systems)
- in/out information of a group of users
- ambient information displays
- remote awareness of others (IMs, Skype)

Examples for functions executing services

- Teleport system: User's Desktop follows from Workstation to Workstation (Want et al, 1992) (commercial example: Sun Ray)
- Car navigation systems that recompute the path automatically on traffic information, wrong turns etc.
- Recording whiteboard senses ad-hoc meeting begin (Brotherton, Abowd and Truong, 1999)
- Mobile devices change settings according to context change
- Location-aware reminders

Examples for functions storing context information

- storing time of recording (digital cameras)
- storing location of recording (zoology application, Pascoe, Ryan and Morse)
- storing meeting information (people present, when, where meeting took place)
- Memory augmentation (Forget-Me-Not, Lamming & Flynn, 1994, Remembrance Agent, Rhodes, 1997)

Why handling Context ist difficult...

Day, Abowd and Salber claim:

> *Handling context is difficult for at least three reasons:*
>
> 1. *there are no guiding principles to support good software engineering practices*
> 2. *designers lack abstractions to think about context*
> 3. *context sensing is very often distributed and leads to complex distributed designs*

Requirements for Dealing with Context

Day, Abowd and Salber claim:

> *. . . we have identified a number of requirements that the framework must fulfill to enable designers to more easily deal with context. These requirements are*
>
> 1. *Separation of concerns*
> 2. *Context interpretation*
> 3. *Transparent, distributed communications*
> 4. *Constant availability of context acquisition*
> 5. *Context storage*
> 6. *Resource discovery*

Separation of concerns...

- Previous application hardcoded sensor data aquisition, signal analysis, context detection and processing. This requires the application programmer to know a lot. (See last problem sheet. . .)
- Can we handle Context like we handle User Input? Applications don't care how a letter is typed (or a button is clicked). . .
- Separation of concerns: Let the toolkit deal with the acquisition of context, let the application programmer deal with using context in his application.
- UI-Widgets are a mechanism for this: All widgets have a common external interface, query and callback mechanisms, an application can treat all widgets (i.e. contexts) the same.

Separation of Concerns

Day, Abowd and Salber claim:

> *By separating how context is acquired from how it is used, applications can now use contextual information without worrying about the details of a sensor and how to acquire context from it. These details are not completely hidden and can be obtained if needed.*

Context Interpretation

- Applications need high-level context information (i.e. User is in a meeting)
- Sensors provide low-level context information (User is in room x, sound level is moderate, there are 6 other users present, the user is sitting etc.)
- The low level context must be interpreted to form high-level context.
- This can be done in the application, but it would not be reusable
- Or it can be done in the toolkit, so that other applications can reuse the context interpretation.

Transport, Discovery

- Context is often acquired by different computer systems
- . . . and may be used by several applications at the same time
- A network-transparent mechanism for context transport is needed.
- The application programmer may not know which context sources are available
- Therefore, a discovery mechanism is needed.

Availability, Storage and History

- An event determining the current context may happen before the application asks for it
- The context acquisition system has to be running all the time
- Applications may need previous contexts for interpretation
- Old context information has to be stored
- Some interpretation is only possible by looking at a previous sequence of contexts
- Context History has to be stored, i.e. a sequence of old contexts
- This may need a lot of storage, so a good selection mechanism is needed.

Context Abstractions

The Context Toolkit abstracts context in the following way:

- Context Widgets : Provides Context, interface to a sensor
- Interpreters : Raises the level of abstraction
- Aggregators : Optimize the flow of data by combining local context sources
- Services : Execute actions, counterpart to widgets
- Discoverers : register capabilities available

Context Toolkit for ActiveBadge example

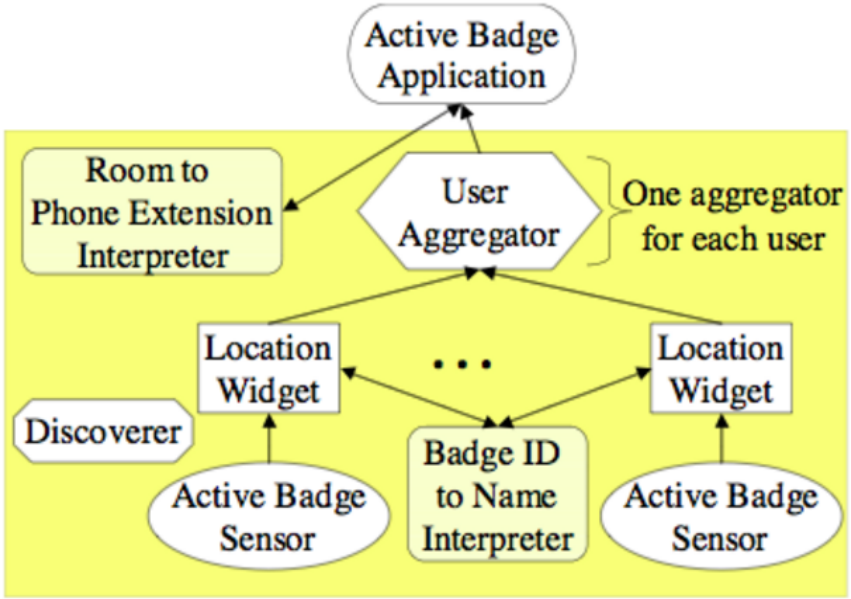

Image from Day, Abowd and Salber, J. of HCI

Context Toolkit for Intercom

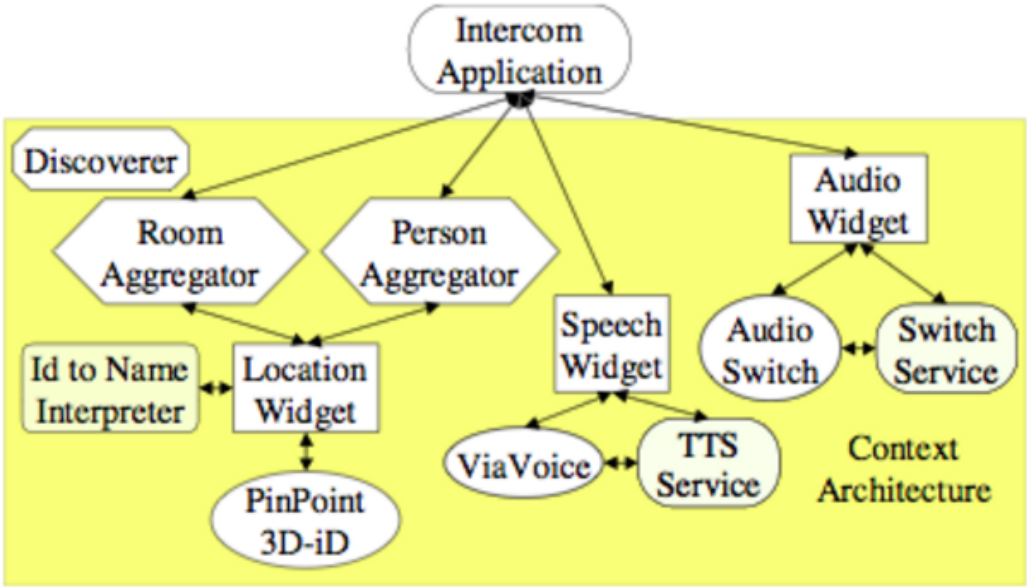

Image from Day, Abowd and Salber, J. of HCI

Design Methodology

The Context Toolkit authors suggests the following design methodology

- Identify Enitites
- Identify Context Attributes
- Itentify Quality of Service Requirements
- Choose Sensors
- Derive a Software Design

Summary

- Context Toolkit
  - Context Abstraction
  - Design Methodology