Chapter 1

Sensors

Recap

Slide Human Computer Interaction II:

- Design Principles
- Theories
  - Levels-of-analysis
  - Stages-of-action
  - GOMS
  - Widget-level
  - Context-of-use
  - Object Action Interface models

1.1 Sensors

Slide Sensors and Wearables:

- User
- Place
- Task

- Environment

User: Identification, Pose, Movement, Physical Condition

Place: Symbolic: (Room, Building, Address, City, Country) Coordinates: Own, G-K, WGS84, Distance to beacons, Position on Map, Map Building

Task: Current task, position in workflow, Execution of task (Quality Control)

Environment: Temperature, Light, Comfort, Health Risks, Task-related

---

Slide Sensing the User:

- Presence of a user
- Actions and Tasks
- Body and Mind

Presence of the user: Energy saving?

Actions and Tasks: HCI, Task detection

Body and Mind: Exhaustion

Slide Simple example: Presence of the user:

- Useful for the wearable computer: Save energy
- Useful for other systems: Communicate the presence of a user
- Useful for the user: No “on”-Switch...

Slide Detecting User Presence:

- Tactile: Design switches that detect

  Pro: Easy to implement (binary input)

  Con: Can be fooled.
- Temperature: Design sensors that detect body heat

Pro: Harder to fool (but still possible), easier to integrate

Con: harder to interface, computation needed

- Motion: Detect body motion

  Pro: Even harder to fool

  Con: even harder to interface, computation and signal analysis needed

---

Slide The details . . . :

- Task: Detecting users of a body-worn wearable computer
- Mechanics: User's Body presses switch when user is present
- Textile: pressure-sensitive textiles
- Electronics: Schmitt trigger circuit
- Computer I/O: Binary input
- System Software: event-driven, similar to ACPI switches
- Application Software: Event API

Slide Sensors and signals:

- A *sensor* is a transducer that is used for measurement
- (A *transducer* is a device that converts one form of energy to another)
- Sensors measure a property of the physical world . . .
- . . . and produce a corresponding signal that can be processed.
- Evaluating the signal at one fixed point in time is called a *measurement*
- A number of measurements taken at different times is called a *time series*
- Typically, these measurements are taken *sequencially at fixed equal time intervals*, i.e. once every second

---

Slide Signals and Computers:

- Typically, sensors produce electrical signals
- A property of the electrical signal (current, voltage, frequency, pulse width, phase) corresponds to the property measured by the sensor.
- In order to use the signal in a computer, the relevant property of the signal needs to be converted into a binary representation.
- This process is called A/D-conversion. It is performed by an A/D-converter.
- Typical A/D-converters convert the voltage of a signal into a binary representation.

Slide Sampling:

- Performing the digitalization of a signal at a fixed point in time is called sampling. Sampling is a form of measurement.
- Signals are continous, a binary representation produced by an A/D-Converter is discrete.
- Multiple signal values reproduce the same binary representation. This phenomenon is called *quanitzation*
- Through quantization, information is lost, resulting in the so-called *quantization error* for a single measurement and *quantization noise* for the whole signal.

Slide How often do we need to sample:

- Sampling in regular, fixed time intervals T produces a time series of binary values.
- The frequency $\frac{1}{T}$ is called sampling frequency.
- Question: How often do we have to sample? Answer: Depends on the signal
- The faster and the more often the signal changes, the more often we have to sample.
- Rule of thumb: Sample slightly more than twice as often as the period of the highest frequency component in the signal.
- Example: CD-Player Signals ≤ 20kHz, sample rate 44.1kHz

Slide Classification of sensors:

- Measured property
- Type of measurement (absolute vs. relative)
- Dimensionality
- limiting factors: precision, noise, frequency response,...

Slide Measurable properties:

- Distance and Position
- Motion
- Light
- Temperature

Slide Distance and Position:

- Signal Attenuation
  - Propagation of a signal in a medium
  - Assumption: propagation path attenuates signal
  - Original signal strength s_1, received signal strength s_2
  - Attenuation depends on signal properties and path properties
- Time-Of-Flight measurement
  - Propagation of a signal in a medium
  - Assumption: constant propagation speed v
  - Signal transmitted at time t_1, received at time t_2.
  - $d = (t_2 - t_1)v$

Slide Position from distance and angle:

- Triangulation

  - Known distances from three known points
  - Known distance differences from four points
  - Example: GPS
- Angle-of-arrival
  - Position from three angles to known points
  - Two angles sufficient for 2D positioning
  - Example: Ships

Slide Motion:

- Differential position
- Doppler
  - Frequency of received signal changes with relative motion
  - Self-transmitted or remote signal
- Inertial Measurement
  - Motion can be measured through accelleration
  - Accelleration can be measured
  - linear acelleration and rotation

Slide From Sensors to Context:

- Sensors measure aspects of the real world
- Sometimes, multiple sensor measurements are combined to measure another aspect
- Example: Position derived from time difference, angles, proximity to beacons
- Question: How do we determine the “right” thing for the wearable to do when we have (combined) sensor information
- Example: Location-based services, supporting mobile workers
- In general: How can we combine all the sensor measurements to a concept that we can use to write an application program for the wearable?

1.2 Context

Slide Context?!?:

- Activity recognition and task context: What is the user doing?
- Physiological monitoring and user context: How is the user doing?
- Environment and situational context: Where is he and what is going on around him?

Slide How to identify the context?:

- Prior knowledge: User model, configuration, external data sources Disadvantage: tedious to aquire, outdated, wrong, expensive
- Explicit interaction: Asking the user (“Press OK to continue”) Disadvantage: impact on the user, lowers user acceptance
- Sensors
  Disadvantage: open research question

[fragile] **Slide Dealing with Context:**

- Context has both a qualitative and quantitative nature
- Sensors give only quantitative information (“The temperature at sensor position 3 is 29.4 degrees celsius”)
- Problem: Making qualitative from quantitative information (“It’s too hot.”)
- Solution: Explicit programming

```
if (position(3)->TempSensor()->curentValue() > 29) {
   setTemperatureContext(CONTEXT_TEMP_TOO_HOT);
}
```

- Question: Implementation complexity? Application Domain Expertise?
- Better Solution: Supervised machine learning (see problem sheet)
- Question: Reusability?

[fragile] **Slide Processing context:**

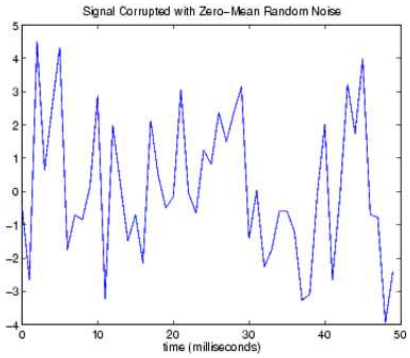
1. Reading sensors
2. Combining sensors
3. Detecting simple context
4. Combining contexts
5. Using context information

[fragile] **Slide Detecting simple contexts:**

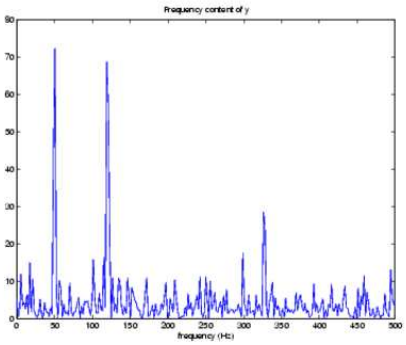
1. Assign “meaning” to sensor values
2. Most simple approach: Tresholds (as seen before)
3. Often needed: filtering
   - temporal filtering: combine several measurements from different times
   - simple example: low-pass-filter (limits high-frequency noise)
   - Integration, Derivatives
   - Time domain → Frequency Domain: Fourier Transform
4. Signal analysis: Pattern recognition, state machines, Hidden Markov Models, Artificial Neural Networks, . . .
5. Combining contexts: first order logic, temporal logic, spatial logic

---

Slide FFT example:



Slide FFT example:



[fragile] **Slide Using Context in an application:**

- Query-based

```
if (getContext(CONTEXT_TEMP) == CONTEXT_TEMP_TOO_HOT)
   {...}
```

- Event-based (Publisher-Subscriber-Pattern)

```
RegisterContextSubscriber(CONTEXT_TEMP,
        CONTEXT_TEMP_TOO_HOT,
        TempContextSubscriberCallback);
...
void TempContextSubscriberCallback(context current)
{
 cout << "Too hot!" << endl;
}
```

[fragile] **Slide Context processing frameworks:**

- Question: How to integrate all steps of context processing
- Context Toolkit (Salber, Dey and Abowd)
  - object-oriented toolkit
  - uses widget and event concept from GUIs

  - organized context detection, but does not define specific mechanism

- Context Recognition Network (CRN) Toolbox: Banach, Kunze, Lukowicz, Amft
  - organizes filters and classifiers for sensor data
  - Supports reuse: filters and classifiers can be parameterized and combined by application programmers
  - Separation of concerns: Experts program and parameterize filters, application programmers use combined filters
  - no learning, experts conduct experiments and build filters

---

1.3 The Context Toolkit

Slide 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

Slide 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 _

full citation: Dey A. K. & Abowd, G. D. (2000a). Towards a better understanding of context and context- awareness. Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness, affiliated with the CHI 2000 Conference on Human Factors in Computer Systems, New York, NY: ACM Press.

Slide 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)

From A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications

Identity refers to the ability to assign a unique identifier to an entity. The identifier has to be unique in the namespace that is used by the applications. Location is more than just position information in 2-D space. It is expanded to include orientation and elevation, as well as all information that can be used to deduce spatial relationships between entities, such as co-location, proximity, or containment. For example, the information that an object is upside down is location information.

Location also applies to places. Places can be located in a frame of reference such as geographical coordinates or relative spatial relationships, and thus have a location.

Status (or activity) identifies intrinsic characteristics of the entity that can be sensed. For a place, it can be the current temperature, or the ambient light or noise level. For a person, it can refer to physiological factors such as vital signs or tiredness, or the activity the person is involved in, such as reading or talking. Similarly for a group of people, status is a characteristic of the group such as their enthusiasm or global mood, or a description of their activity, such as attending a lecture, or having a meeting. For software components, status basically refers to any attribute of the software component that can be queried. Typical examples are the uptime or load of a CPU, the existence or state of files in a file system, or the state of a software component such as an application.

Finally, time is context information as it helps characterize a situation. It enables us to leverage off the richness and value of historical information. It is most often used in conjunction with other pieces of context, either as a timestamp or as a time span, indicating an instant or period during which some other contextual information is known or relevant. However, in some cases, just knowing the relative ordering of events or causality is sufficient.

[. . .]

This classification introduces three categories of functions, related to the presentation of information, the execution of services, and the storage of context information attached to other captured information for later retrieval.

Slide 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)

Slide 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

Mobile Device context awareness: Harrison B. L., Fishkin, K. P., Gujar, A., Mochon, C. & Want, R. (1998). Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. Proceedings of the CHI'98 Conference on Human Factors in Computer Systems, 17-24. New York, NY: ACM Press.

Hinckley K., Pierce, J., Sinclair, M. & Horvitz, E. (2000). Sensing techniques for mobile interaction. To appear in the Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST 2000), New York, NY: ACM Press.

Schmidt A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. V. & Velde, W. V. d. (1999). Advanced interaction in context. Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC '99), 89-101. Heidelberg, Germany: Springer Verlag.

Schmidt A., Takaluoma, A. & Mantyjarvi, J. (2000). Context-aware telephony over WAP. Personal Technologies, 4(4), 225-229.

Location-aware reminders: Beigl M. (2000). MemoClip: A location based remembrance appliance. Personal Technologies, 4(4).

Marmasse N. & Schmandt, C. (2000). Location aware information delivery with comMotion. Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K), Heidelberg, Germany: Springer Verlag.

Slide 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)

Slide 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

---

Slide 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

Slide 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.

Slide 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.

Slide 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.

Slide 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.

Slide 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.

Slide 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

Slide Context Toolkit for ActiveBadge example:

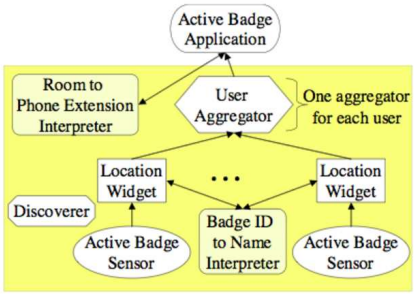


Image from Day, Abowd and Salber, J. of HCI

Slide Context Toolkit for Intercom:

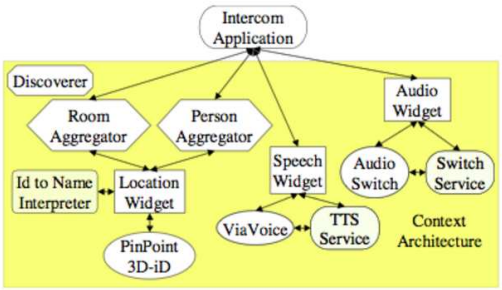


Image from Day, Abowd and Salber, J. of HCI

Slide 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

From Day, Abowd and Salber, J. of HCI:

8.4. Applying the Design Methodology Starting from the usage scenarios presented above (1.1), we must first identify the entities that are likely to provide context information to the Conference Assistant application. We then look at their context attributes, and the requirements for each attribute. Then we examine our choice of sensors to acquire the specified context information. Finally, we derive a software architecture for the application from this information on context sources and attributes.

Identifying Entities Object Oriented Analysis (OOA) methods provide useful support to identify objects. Among these objects, users are usually context entities. In the case of the Conference Assistant, both attendees and presenters are of interest. Places, rooms for the Conference Assistant, are also modeled as context entities. Physical and computational objects must be handled with care; good candidates for context entities are objects that are highly dynamic. The conference schedule for example, or a conference track are static objects and do not qualify. Indeed in our scenario, we do not account for changes in the

conference schedule, although that would be an obvious extension. On the other hand, a presentation, or the current slide or Web page displayed by the presenter are objects of interest for context modeling. Also of interest are objects whose content we wish to capture and annotate with context information, such as a question asked by an attendee, or notes taken by the user. Identifying Context Attributes Once we have identified context entities, we look at candidate context attributes for each entity. Again OOA methods can provide insight by identifying attributes of context entities' corresponding objects. Then, we must decide if a given attribute is context-related or not. To identify context-related attributes, we use the context categories of Section 2.2 as a guide. For each entity, attributes that fit into context categories are interesting context attributes. For example, for an attendee, contact information, interests, and colleagues are derived from identity. The room where the attendee currently stands is location information. The time when the user moves into and out of room is needed by the retrieval application that shows the user a timeline of her whereabouts. Finally the current level of interest of the user falls into the status category. For a presenter, we only need the name of the person, that is, her identity. When it comes to the room entity, we notice something interesting. The room is only used to characterize the location of the user, and the location where a presentation is held. As a matter of fact, our scenario does not involve attributes of the room per se, except its identity. Since we have to distinguish between the different rooms where presentations are being held, we need to identify the rooms in some way. However, one could imagine that a different application would involve context attributes of the room. For example, we might be interested in monitoring the headcount in all rooms at all time, to help planners for the next edition of the conference. In that case, we would identify an "occupancy" attribute for each room. For a presentation, we are concerned with its location, that is, the room it is held in. The current slide or Web page of a presentation has an identity (its slide number or URL), secondary identity information (a title) and status information (a thumbnail shown on the user's handheld, according to the scenario). Question and user notes share the same attributes both the slide or Web page they relate to and the time at which the question was asked or the note was taken are of interest. Question or note contents are captured and annotated with context information. We need to be able to relate the note either a slide or Web page and display the note in the timeline provided by the retrieval

application. Similarly, we annotate questions with time information and the slide or Web page the question was about (if any). Figure 16 summarizes the identification of context entities and their context attributes for the Conference Assistant application.

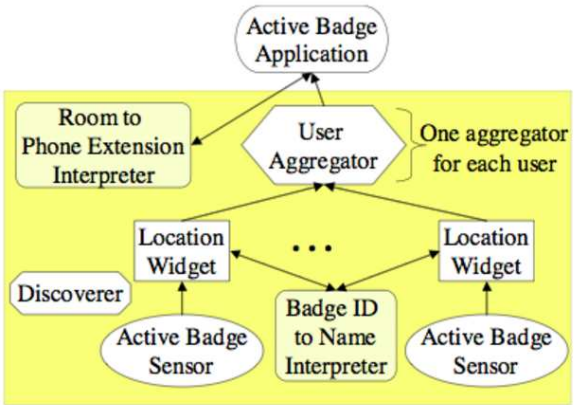


Figure 16. Context entities and attributes for the Conference Assistant application. Attributes are derived context types.

Identifying Quality of Service Requirements We now look at the quality of service con-

straints for each context attribute. In this part of the analysis, we focus on dynamic attributes. The names of the attendee and presenter as well as the room name or the room a presentation occurs in do not present particular challenges with regard to quality of service. They are set once and for all, and they might be acquired once by the application at startup. On the other hand, the location of the user is highly dynamic, as well as the context attributes of the current slide or web page, and the level of interest of the user. The user's location must be available to the application whenever the user is on the conference grounds. Thus, we require that coverage for the user's location is at least the space occupied by the conference. This space does not need to be contiguous. In our demonstration setup in our lab, we used two areas at the opposite ends of the lab and a room across the corridor as our three rooms where the three tracks of a simulated conference are held. The requirements are less strict for resolution: we want to be able to differentiate the user's location among the three rooms she's visiting. But the location of the user within the room is not useful in our scenario. Thus, room-level resolution is sufficient. However, if we extended the application to help conference attendees navigate between different rooms, a much higher level of resolution would be required. We must specify high accuracy and reliability requirements: Incorrect or missing user location would seriously jeopardize the usefulness of the Conference Assistant. For frequency and timeliness, our requirements must match the expected behavior of the users. We expect attendees to move at normal pace between rooms and stay in a room several minutes at least. The retrieval application, which will allow users to re-live their experience, displays a schedule whose resolution is no more than five minutes. But when a user moves into a room, we want to display almost immediately the information about the presentation currently held in the room. The user might accept a delay of a couple of seconds for this information to be displayed on her handheld. As a consequence, we may set our requirements for frequency at one update every second minimum and timeliness at one second maximum. Context attributes of a slide or a web page are presented to the user on the handheld device. They are attached to notes taken by the user. Coverage and resolution apply to the slides' context attributes. Partial coverage would mean the context information available to the application would be constrained (e.g., only slides with no color pictures, or titles shorter than ten characters). Limited resolution would mean, for example, that only every other Web page jump would be transmitted to the application. Of course, either limited coverage or limited resolution would cripple the Conference Assistant. We thus require 100resolution. We also set high requirements for accuracy and reliability for similar reasons. Finally, we must evaluate our needs in terms of update frequency and timeliness. As a first approximation, we can set both the update frequency and timeliness to about one second. We want to avoid glaring discrepancies between the display shown to the audience and the display presented on the handheld. When the application is being deployed, these quality of service requirements would most definitely need to be backed with theoretical data or user testing. The level of interest is set by the user, and is chosen among three values (low, medium and high). The level of interest is communicated to colleagues of the user. Since the level of interest is a discrete variable with very few values, required coverage and resolution are 100available, respectively. Similarly, very high accuracy is required. However, we can tolerate less than perfect reliability. This feature is not essential for the operation of the Conference Assistant. Frequency and timeliness are also not of the highest importance and a typical figure of a few minutes for both would be acceptable. Choosing Sensors At this stage of the analysis, we have identified the pieces of context that we need to acquire and the quality of service requirements for each of them. We now examine the sensing techniques available for each context attribute and use quality of service requirements to choose an appropriate set of sensing techniques. Except for the user's location, our choice of sensing techniques is severely limited. Information about the current slide or Web page can only be acquired through inter-application communication mechanisms such as COM/DCOM. The level of interest of the user is input manually by the user. Other techniques could be used but were just not practical in our setup. Computer vision techniques could extract relevant infor-

mation from a slide, although reliability may be an issue. Physiological sensors may be able to determine the level of interest of the use, although reliability may also be an issue, as well as the intrusiveness of the sensors. We have a number of techniques available for acquiring the user's location. Among them, we considered Olivetti's Active Badges (Want et al., 1992), the Dallas Semiconductor iButtons, and Pinpoint 3D-iD. All systems satisfy our requirements in terms of coverage, resolution and

accuracy. With Active Badges, users wear IR-transmitting badges whose signal is picked by fixed receivers. With iButtons, users carry iButtons devices and dock them explicitly to indicate their change of location; adding docks extends the coverage and a room-level resolution is easily achieved. With the Pinpoint system, users carry a RF active tag whose signal is received by a set of antennas (seven in our setup). Triangulation between the signals received at the antennas provides the tag's location. We have set high requirements for reliability but all systems have potential drawbacks. Active Badges require line of sight between the badge and a reader installed indoors. In a room potentially packed with attendees, we have no guarantee that the transmission will be successful. In this regard, a RF-based system like the Pinpoint system is preferable. But it suffers from other potential reliability problems: the tag orientation is crucial for its operation. Providing a carefully designed tag holder could alleviate this problem. Finally, the iButtons suffer from another kind of reliability problem: since they are operated manually by the user, we run the risk that users will forget to indicate their change of location. In our experience with iButtons and the Pinpoint system, they both satisfy our frequency and timeliness requirements. In summary of this analysis, the Pinpoint system is the most appropriate given our requirements. However, the complexity involved in the initial setup of the Pinpoint system led us to use iButtons in our first prototype. The iButtons provide one significant advantage over the PinPoint system, that of providing the users control over disclosing their location to the system and others. Only motivated users, those that want to collaborate with their colleagues or want to take advantage of the advanced services that require user location, will provide their location. Both systems provide slightly different information. The Pinpoint system provides the identity of the tag as a numeric ID and a zone number, according to a zone map defined statically. With the iButtons, each dock provides the identity of the docking iButton as a numeric ID. When using several docks, we attach location information to each dock. Deriving a Software Design To derive a software design from the analysis of context entities and attributes, we apply the rules introduced in Section 3.3. We first map entities to aggregators, then context attributes to context widgets, and finally identify the need for interpreters to abstract or infer context information. According to Figure 16, our model for the Conference Assistant application consists of the following entities: Attendee, Presenter, Room, Presentation, Slide and Web page. Each of these entities has a number of attributes, also presented in Figure 16. According to our mapping rules, the software design of the application contains one aggregator component for each entity, and one context widget for each attribute of an entity. We apply them to produce the aggregators and widgets sections of the software components list shown in Figure 17.

accuracy. With Active Badges, users wear IR-transmitting badges whose signal is picked by fixed receivers. With iButtons, users carry iButtons devices and dock them explicitly to indicate their change of location; adding docks extends the coverage and a room-level resolution is easily achieved. With the Pinpoint system, users carry a RF active tag whose signal is received by a set of antennas (seven in our setup). Triangulation between the signals received at the antennas provides the tag's location. We have set high requirements for reliability but all systems have potential drawbacks. Active Badges require line of sight between the badge and a reader installed indoors. In a room potentially packed with attendees, we have no guarantee that the transmission will be successful. In this regard, a RF-based system like the Pinpoint system is preferable. But it suffers from other potential reliability problems: the tag orientation is crucial for its operation. Providing a carefully

designed tag holder could alleviate this problem. Finally, the iButtons suffer from another kind of reliability problem: since they are operated manually by the user, we run the risk that users will forget to indicate their change of location. In our experience with iButtons and the Pinpoint system, they both satisfy our frequency and timeliness requirements. In summary of this analysis, the Pinpoint system is the most appropriate given our requirements. However, the complexity involved in the initial setup of the Pinpoint system led us to use iButtons in our first prototype. The iButtons provide one significant advantage over the PinPoint system, that of providing the users control over disclosing their location to the system and others. Only motivated users, those that want to collaborate with their colleagues or want to take advantage of the advanced services that require user location, will provide their location. Both systems provide slightly different information. The Pinpoint system provides the identity of the tag as a numeric ID and a zone number, according to a zone map defined statically. With the iButtons, each dock provides the identity of the docking iButton as a numeric ID. When using several docks, we attach location information to each dock. Deriving a Software Design To derive a software design from the analysis of context entities and attributes, we apply the rules introduced in Section 3.3. We first map entities to aggregators, then context attributes to context widgets, and finally identify the need for interpreters to abstract or infer context information. According to Figure 16, our model for the Conference Assistant application consists of the following entities: Attendee, Presenter, Room, Presentation, Slide and Web page. Each of these entities has a number of attributes, also presented in Figure 16. According to our mapping rules, the software design of the application contains one aggregator component for each entity, and one context widget for each attribute of an entity. We apply them to produce the aggregators and widgets sections of the software components list shown in Figure 17. accuracy. With Active Badges, users wear IR-transmitting badges whose signal is picked by fixed receivers. With iButtons, users carry iButtons devices and dock them explicitly to indicate their change of location; adding docks extends the coverage and a room-level resolution is easily achieved. With the Pinpoint system, users carry a RF active tag whose signal is received by a set of antennas (seven in our setup). Triangulation between the signals received at the antennas provides the tag's location. We have set high requirements for reliability but all systems have potential drawbacks. Active Badges require line of sight between the badge and a reader installed indoors. In a room potentially packed with attendees, we have no guarantee that the transmission will be successful. In this regard, a RF-based system like the Pinpoint system is preferable. But it suffers from other potential reliability problems: the tag orientation is crucial for its operation. Providing a carefully designed tag holder could alleviate this problem. Finally, the iButtons suffer from another kind of reliability problem: since they are operated manually by the user, we run the risk that users will forget to indicate their change of location. In our experience with iButtons and the Pinpoint system, they both satisfy our frequency and timeliness requirements. In summary of this analysis, the Pinpoint system is the most appropriate given our requirements. However, the complexity involved in the initial setup of the Pinpoint system led us to use iButtons in our first prototype. The iButtons provide one significant advantage over the PinPoint system, that of providing the users control over disclosing their location to the system and others. Only motivated users, those that want to collaborate with their colleagues or want to take advantage of the advanced services that require user location, will provide their location. Both systems provide slightly different information. The Pinpoint system provides the identity of the tag as a numeric ID and a zone number, according to a zone map defined statically. With the iButtons, each dock provides the identity of the docking iButton as a numeric ID. When using several docks, we attach location information to each dock. Deriving a Software Design To derive a software design from the analysis of context entities and attributes, we apply the rules introduced in Section 3.3. We first map entities to aggregators, then context attributes to context widgets, and finally identify the need for interpreters to abstract or infer context information. According to Figure 16, our model for the Conference Assistant application consists of the following entities: Attendee, Presenter, Room,

Presentation, Slide and Web page. Each of these entities has a number of attributes, also presented in Figure 16. According to our mapping rules, the software design of the application contains one aggregator component for each entity, and one context widget for each attribute of an entity. We apply them to produce the aggregators and widgets sections of the software components list shown in Figure 17.

Aggregators
Attendee
Presenter
Room
Presentation
Slide
Web page
Widgets
Name (Attendee)
Room (Attendee)
Times of departure/arrival (Attendee)
Level of interest (Attendee)
Name (Presenter)
Name (Room)
Room (Presentation)
Number (Slide)
Title (Slide)
Thumbnail (Slide)
URL (Web page)
Title (Web page)
Thumbnail (Web page)
Current slide (Question)
Time (Question)
Current slide (User note)
Time (User note)
Interpreters
Tag ID to Name (Attendee)
Zone ID to Room (Attendee)
iButton ID to Name (Attendee)

Figure 17. Software components derived for the Conference Assistant. Entity name in parentheses indicates the entity the attribute relates to. In the Conference Assistant application, we do not use complex context inference. However, we need interpreters to abstract information acquired from our iButton and/or Pinpoint sensors. We assume that we may use either of the two sensing technologies during the life cycle of the application. As a consequence, we need interpreters on one hand to transform an iButton ID to an attendee name, and on the other hand, to transform both tag ID and zone ID provided by the Pinpoint system to attendee and room names. We will now show how the list of components shown in Figure 17 can be simplified, and we will examine the tradeoffs involved. First, let us consider aggregators that provide a single piece of context information acquired directly through a context widget, like Presenter, Room, and Presentation. Since aggregators are intended to aggregate information from multiple sources, an aggregator with only one source of information is of limited usefulness. Such an aggregator can be removed from the software design. The gain is mostly in simplicity of implementation and performance. The main drawback concerns future extensibility. If new context attributes of the Presentation entity, such as the number of attendees and their overall interest were to be added, the aggregator would make these new attributes easier to access. As a result, we can remove the Presenter, Room, and Presentation Aggregators and let the application interact directly with their respective widgets. If attributes of an entity are clearly static throughout the expected usage period of the application, they do not need to be acquired through context widgets. As a result, the Name Widget associated with the Room Aggregator can also be removed, since the room name does not change.

1.4 CRN Toolbox

Slide Context Recognition Network Toolbox:

- Written by David Bannach (ETH/UMIT/Uni Passau)
- Implements a network of readers, filters, classifiers, recorders, writers
- Distributed: can pass data over network links
- Java editor, Interface to Matlab

Slide CRN Components:

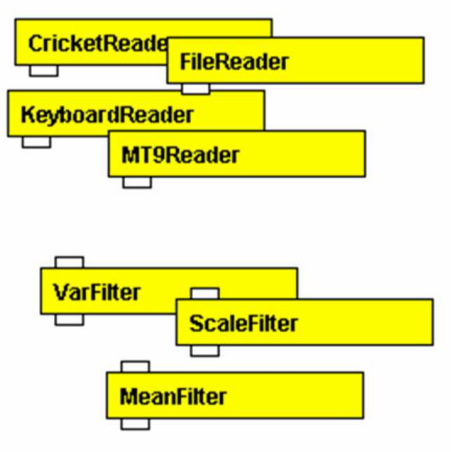


Image from D. Bannach, Uni Passau

Slide CRN Components:

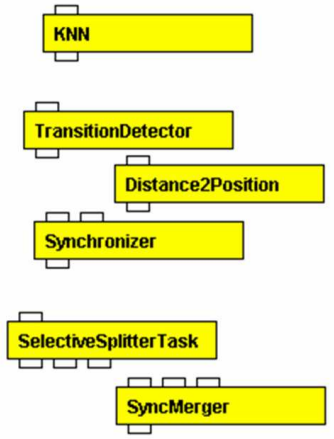


Image from D. Bannach, Uni Passau

Slide CRN Components:

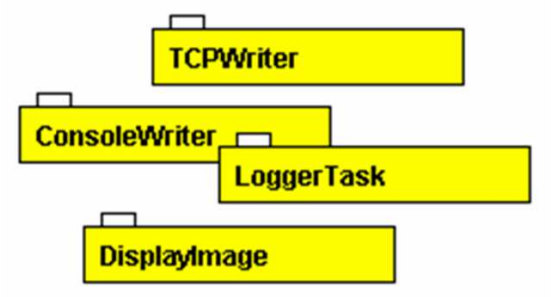


Image from D. Bannach, Uni Passau

Slide CRN Toolbox Chain:

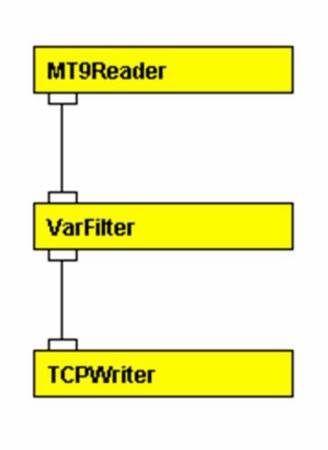


Image from D. Bannach, Uni Passau

Slide CRN Editor:

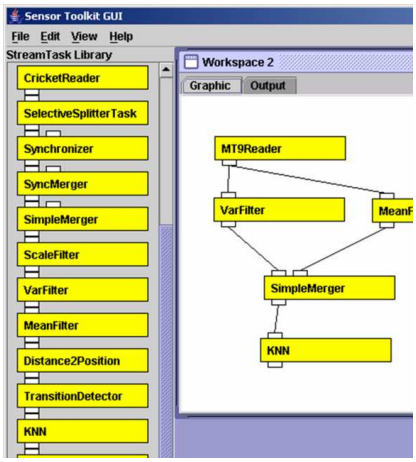


Image from D. Bannach, Uni Passau

Slide CRN Matlab Interface:

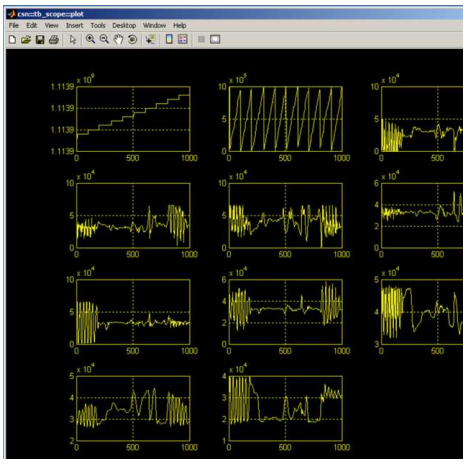


Image from D. Bannach, Uni Passau

Slide Summary (1):

- Sensors
  - Integration into wearables
  - Sensor Types
  - Details: position

Slide Summary (2):

- Context
  - Sources
  - Handling Context
  - Using Context
- Context Toolkit
  - Context Abstraction
  - Design Methodology
- CRN Toolbox
