Context-aware computing

Introduction

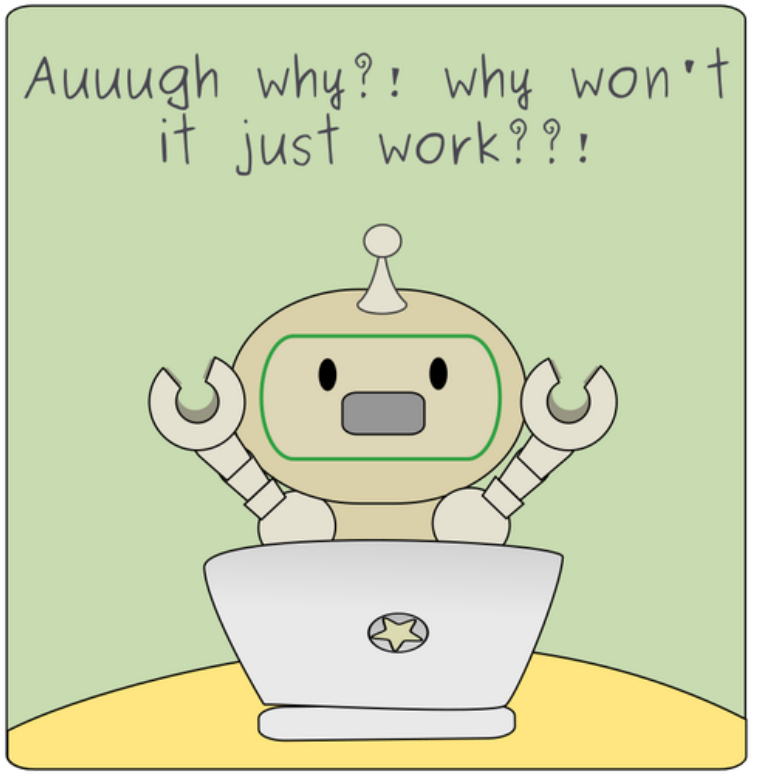
- When humans speak with humans, they are able to use information apparent from the current situation, or context, to increase the conversational bandwidth

- This ability to convey ideas does not transfer well when humans interact with computers

Copyright:
http://arnoldzwicky.s3.amazonaws.com/BrowmanTalk.jpg

Introduction

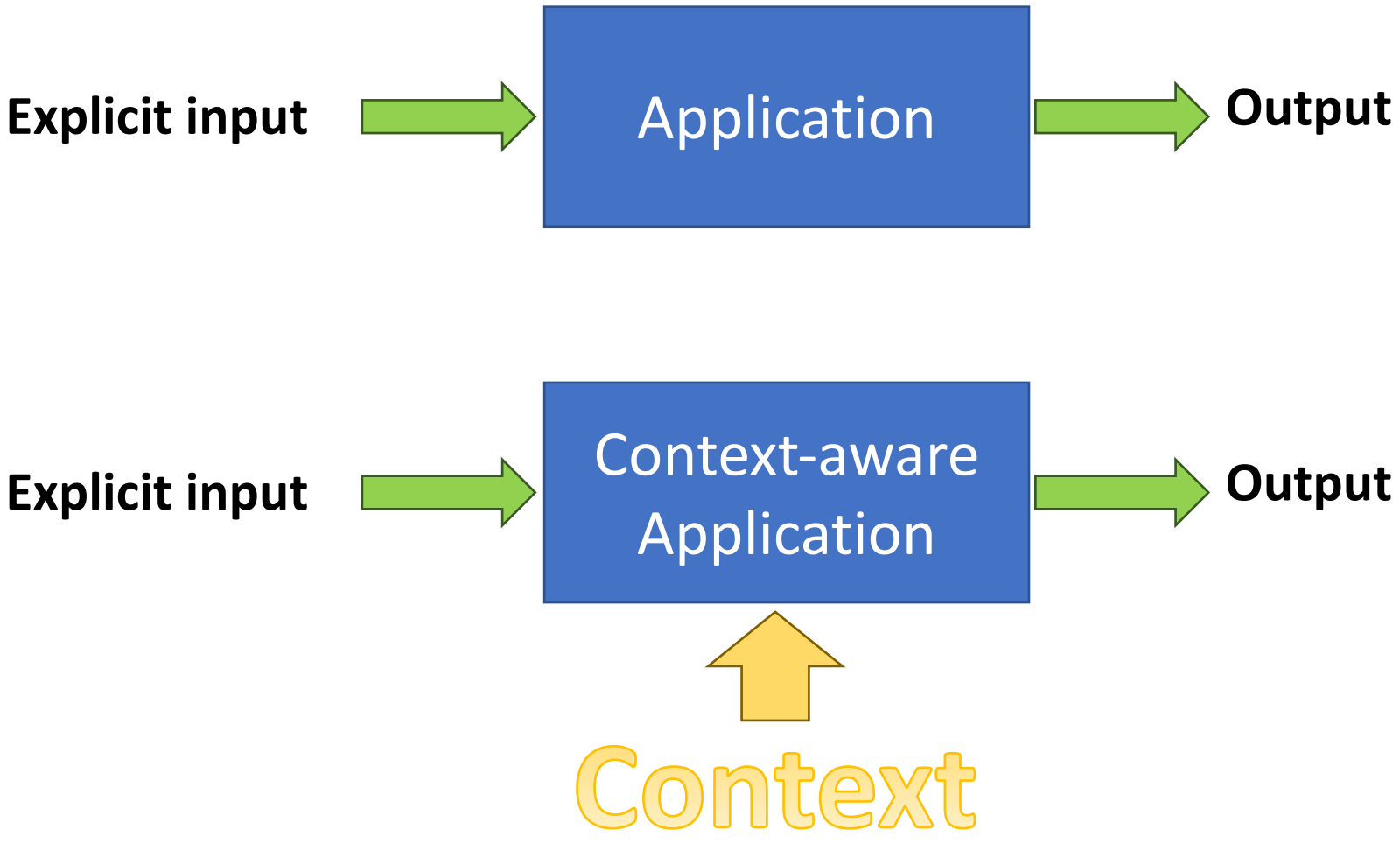
- **Context-aware computing**: using context as an implicit cue to enrich the impoverished interaction from humans to computers, making it easier to interact with computers
- A step towards *calm* technology: an approach to ubiquitous computing, where computing moves back and forth between the center and periphery of the user's attention



Introduction

- Applications that use context, whether on a desktop or in a mobile or ubiquitous computing environment, are called ***context-aware***
- Availability of commercial, off-the-shelf sensing technologies is making it more viable to sense context in a variety of environments
- Mobility creates situations where the user's context, such as his or her location and the people and objects around him/her, is more dynamic

Introduction



Introduction

- Wide range of possible user situations: we need to have a means for the services to adapt appropriately, in order to best support the human–computer and human–environment interactions
- Context-aware applications are becoming more prevalent and can be found in the areas of
  - wearable computing
  - mobile computing
  - adaptive and intelligent user interfaces
  - augmented reality
  - intelligent environments

What is context?

We all have a general and maybe vague idea of what context is

- *Schilit and Theimer (1994):* refer to context as location, identities of nearby people and objects, and changes to those objects
- *Brown et al. (1997):* define context as location, identities of the people around the user, the time of day, season, temperature, etc
- *Ryan et al. (1998):* define context as the user's location, environment, identity, and time
- *Dey (1998):* enumerated context as the user's emotional state, focus of attention, location and orientation, date and time, and objects and people in the user's environment
- These definitions define context by example and are difficult to apply
- Some consider context to be the user's environment, whereas others consider it to be the application's environment

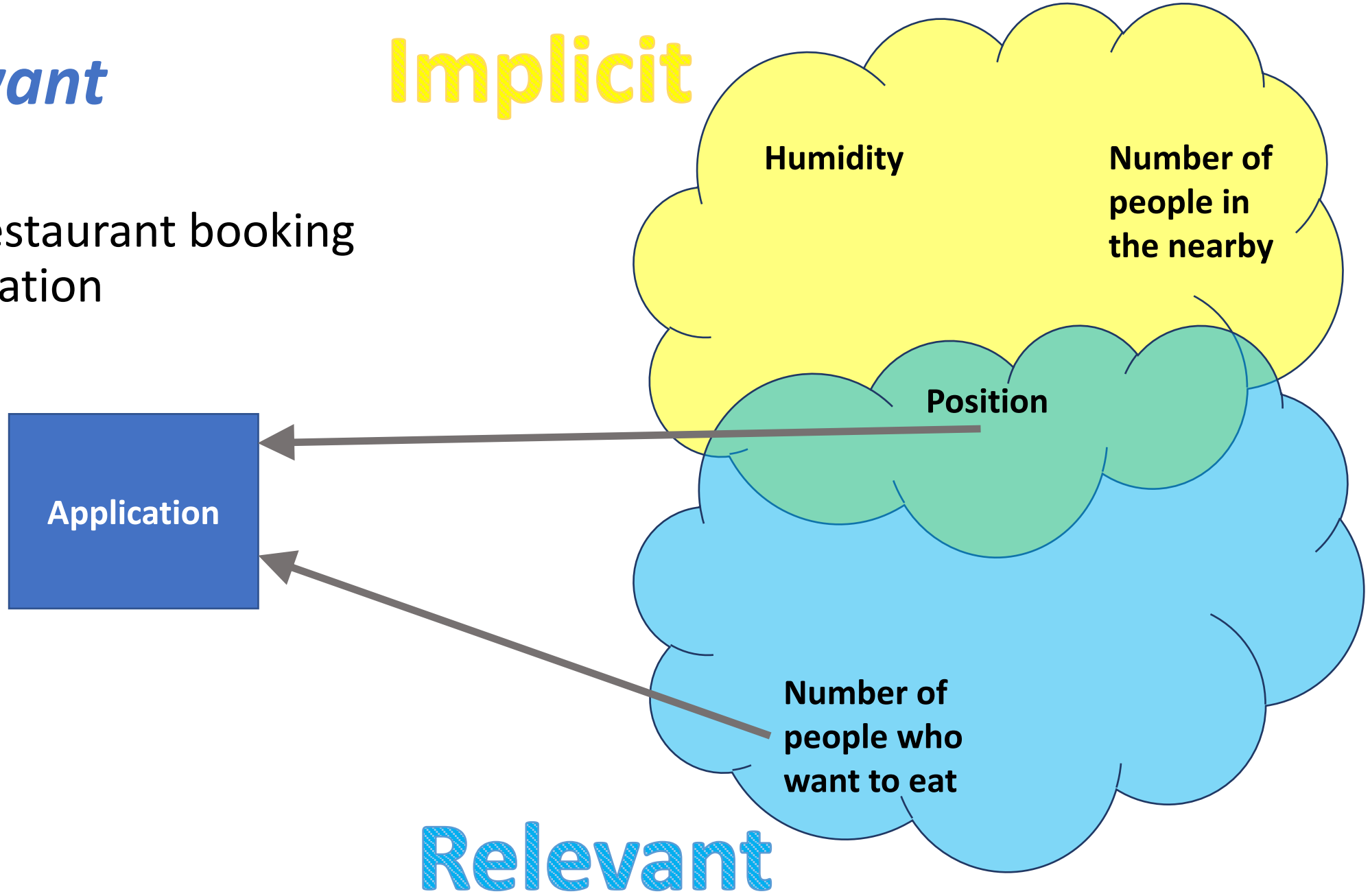
Context, a definition

“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 application themselves.”

Dey and Abowd (2000a)

Relevant

- In a restaurant booking application



Context: sensing

- Environmental sensing is relatively straight-forward
  - Use sensors for temperature, humidity, pressure, etc
- Human sensing is a little harder (ranked easy to hard)
  - **When**: time (Easiest)
  - **Where**: location
  - **Who**: identification
  - **What**: running, eating, cooking (meta task)
  - **Why**: reason for actions (extremely hard!)
  - ***How***: (Mood) happy, sad, bored (gesture recognition, cameras)

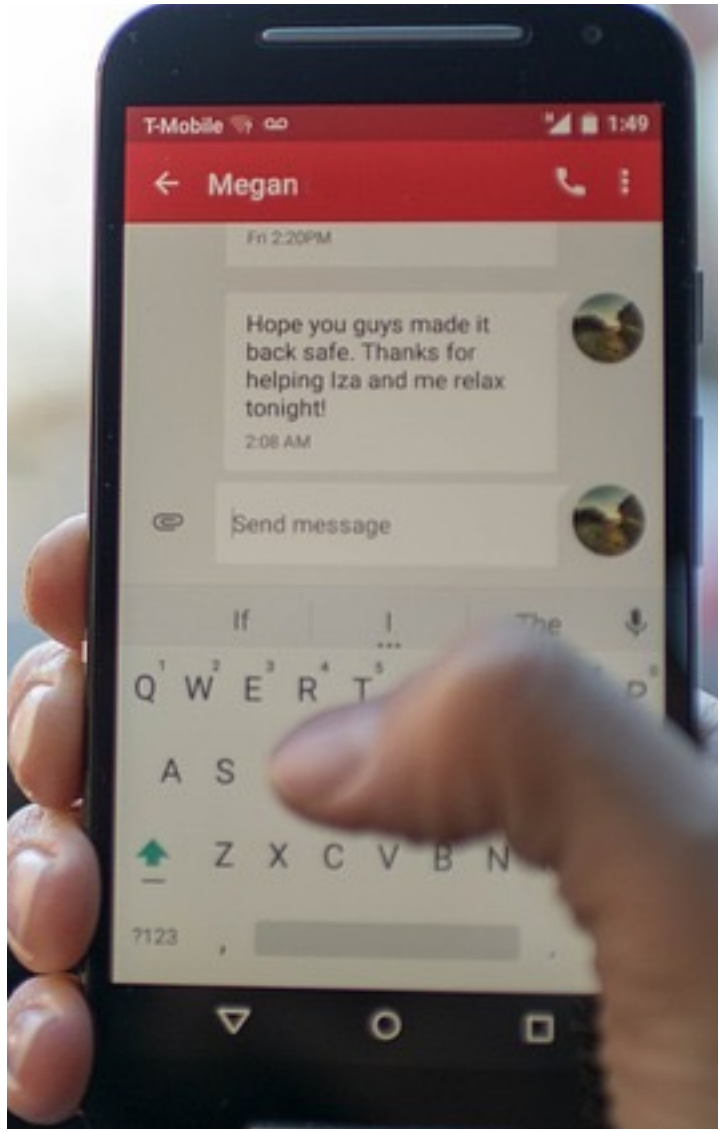
- Pervasive computing integrates
  - location sensing, user identification, emotion sensing, gesture recognition, activity sensing, user intent

Context

- Certain types of context that are, in practice, more important than others
- These are **location** (where), **identity** (who), **time** (when), and **activity** (what)
- Pointers to other sources of contextual information:
  - given a person's identity, we can acquire many pieces of related information: phone numbers, addresses, email addresses, birth date, list of friends, relationships to other people in the environment, etc
  - given an entity's location, we can determine what other objects or people are near the entity and what activity is occurring near the entity

Context

- **Physical**
  - Measured by hardware sensors
  - Examples: humidity, pollution level, noise, position, light, …
- **Logical**
  - Acquired by monitoring the user's interaction with the application or derived from physical context
  - Examples: user's goal, tiredness, mood, communication pattern, …


T-Mobile
1:49
Megan
Fri 2:20PM
Hope you guys made it back safe. Thanks for helping Iza and me relax tonight!
2:08 AM
Send message


T-Mobile
1:49
Megan
Fri 2:20PM
Hope you guys made it back safe. Thanks for helping Iza and me relax tonight!
2:08 AM
Send message

Use of context

There are three categories of common usages:

- **Presentation** of information and services to the user
  - A list of printers in the nearby is updated as the user moves in a building
- Automatic **execution** of a service
  - A reminder is triggered as the user passes in the nearby of a grocery
  - Silent mode is turned on during class lectures
- **Tagging** of context to information for later retrieval
  - Humidity and temperature during a running session

Designing contex-aware applications

- **Specification**: what context-aware behaviors your application will have and in which situations each behavior should be executed
- **Acquisition**:
  - determining HW and/or SW sensors required to acquire the context identified
  - using the API to communicate with the sensor, and define how to be notified when changes occur
  - store the context, combine it with other context
- **Delivery**: specifying how context should be delivered from the sensors to the (possibly remote) applications that will use the context
- **Reception**: application specifies what context it is interested in (possibly indicating from which sensors) and receiving that context
- **Action**: analyzing all the received context to determine what context-aware behavior to execute, and then execute actions

Designing contex-aware applications

Example:

- **Specification**: Light up a LED depending on the activity level in a remote room (if level > X …)
- **Acquisition**: Write software to analyze frame-to-frame changes in video camera image of remote location
- **Delivery**: Make the percentage change in activity available to interested applications, using publish-subscribe approach
- **Reception**: Application wants to receive all changes in activity above a particular threshold
- **Action**: Application analyzes received activity changes, and lights up the appropriate LED (e.g. low activity: red, moderate activity: yellow, high activity: green)

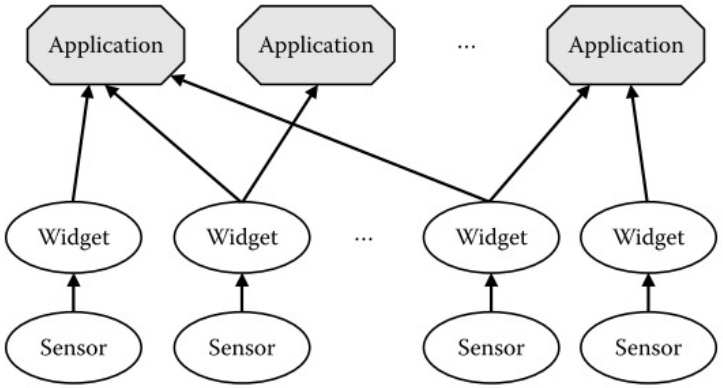
Building context-aware applications

Three main alternatives used for building applications:

- no support
- a widget- or object-based system
- a blackboard-based system

Building context-aware applications: widgets

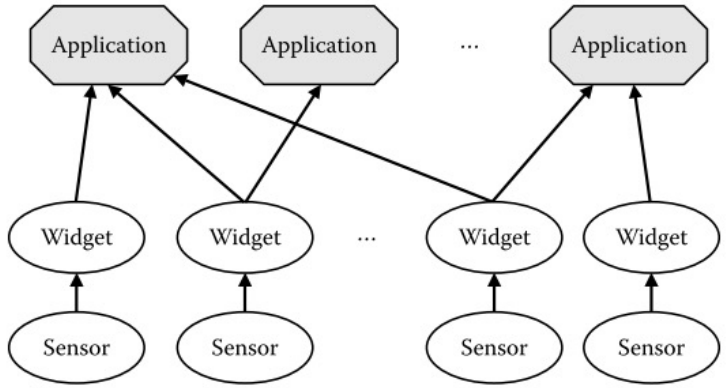
- **Context widget**: a software component that provides applications with access to context information
- GUI widgets insulate applications from presentation concerns, similarly context widgets insulate applications from context acquisition concerns by wrapping sensors with a uniform interface
- Separation of concerns: they hide the complexity of the actual sensors used from the application
  - Whether the presence of people is sensed using IR sensors, floor sensors, video image processing, or a combination of these should not impact the design of the application
- Abstract context information to suit the expected needs of applications
  - Widget that tracks the location of a user within a building notifies the application only when the user moves from one room to another, and does not report less significant moves to the application



From «Ubiquitous Computing Fundamentals».

Building context-aware applications: widgets

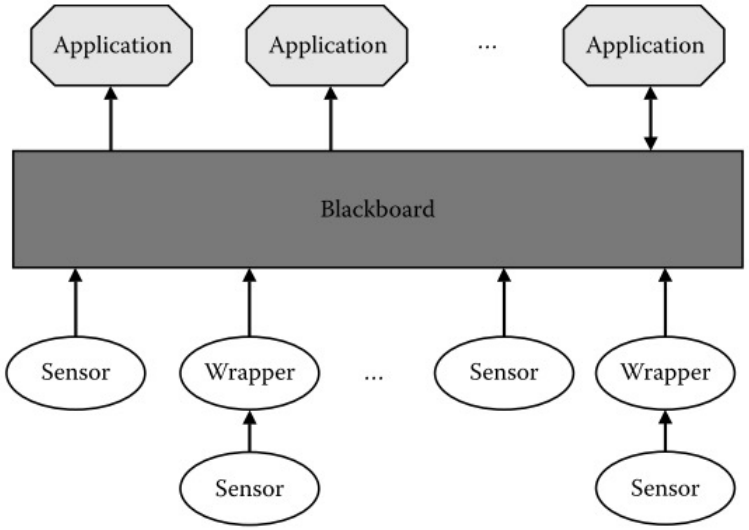
- Access to context data through querying and notification mechanisms
- Reusable and customizable building blocks of context sensing
  - Widget that tracks the location of a user can be used by a variety of applications, from tour guides to industrial automation
- Context widgets can be combined in ways similar to GUI widgets
  - E.g. a *presence* widget senses the presence of people in a room. A *meeting* widget may be built on top of a *presence* widget and assume a meeting is beginning when two or more people are present



From «Ubiquitous Computing Fundamentals».

Building context-aware applications: blackboard

- Blackboards originated from the Linda programming language and tuple space model from the early 1980s
- Components can place information into the storage system, and read or remove this information
- Blackboards may have the ability to notify components when information of interest has been added to the blackboard
- Blackboard approaches sometimes inefficient particularly as the amount of data in the tuple space grows and searching for specific tuples becomes harder



Issues when building contex-aware applications

Context is a proxy for human intent

- Context awareness is about understanding human intent
- Applications would use this human intent to adapt appropriately by providing information or taking actions
- However, context information is only a **proxy** for this intent
- Museum tour guide: a user standing in front of a particular painting may not represent interest in the artpiece, maybe user has her back turned to the exhibit and is having a conversation with a friend there
- Solutions:
  - Additional context is required to understand human intent
  - Applications should express a certainty in their beliefs about their sensed information and, if that certainty is not above an appropriate threshold, ask for confirmation of this information before taking action

Issues when building contex-aware applications

Context ambiguity

- Many sources of ambiguity or errors: context sensors can sense incorrectly, fail, or be unsure about what they sensed; context inferencing systems can inaccurately reach conclusions about a situation
- One approach to dealing with context ambiguity is to combine multiple disparate sources of the same type of context to improve the accuracy or dependability of the provided context
- This is commonly known as sensor fusion
- Another possibility: context does not change completely randomly, but instead shows some coherence over time

Issues when building contex-aware applications

Rules vs Machine Learning

- Context-aware applications are commonly designed from a set of **if then** rules
  - if the application senses a situation, then it should perform a particular action
  - intuitive, easy to build
- Rule-based systems also have some disadvantages:
  - Possible conflicts between rules due to hidden dependencies between rules
  - difficult to understand the impact of adding or removing a rule when the number of rules grows larger
- Machine Learning (ML)
  - developer collects data on the types of situations that a user will experience and the types of adaptation desired
  - ML applied to learn the probabilistic relationships between the situations and adaptations, rather than have these relationships be hardcoded and deterministic
  - It may require a large amount of data to learn

Issues when building contex-aware applications

Privacy

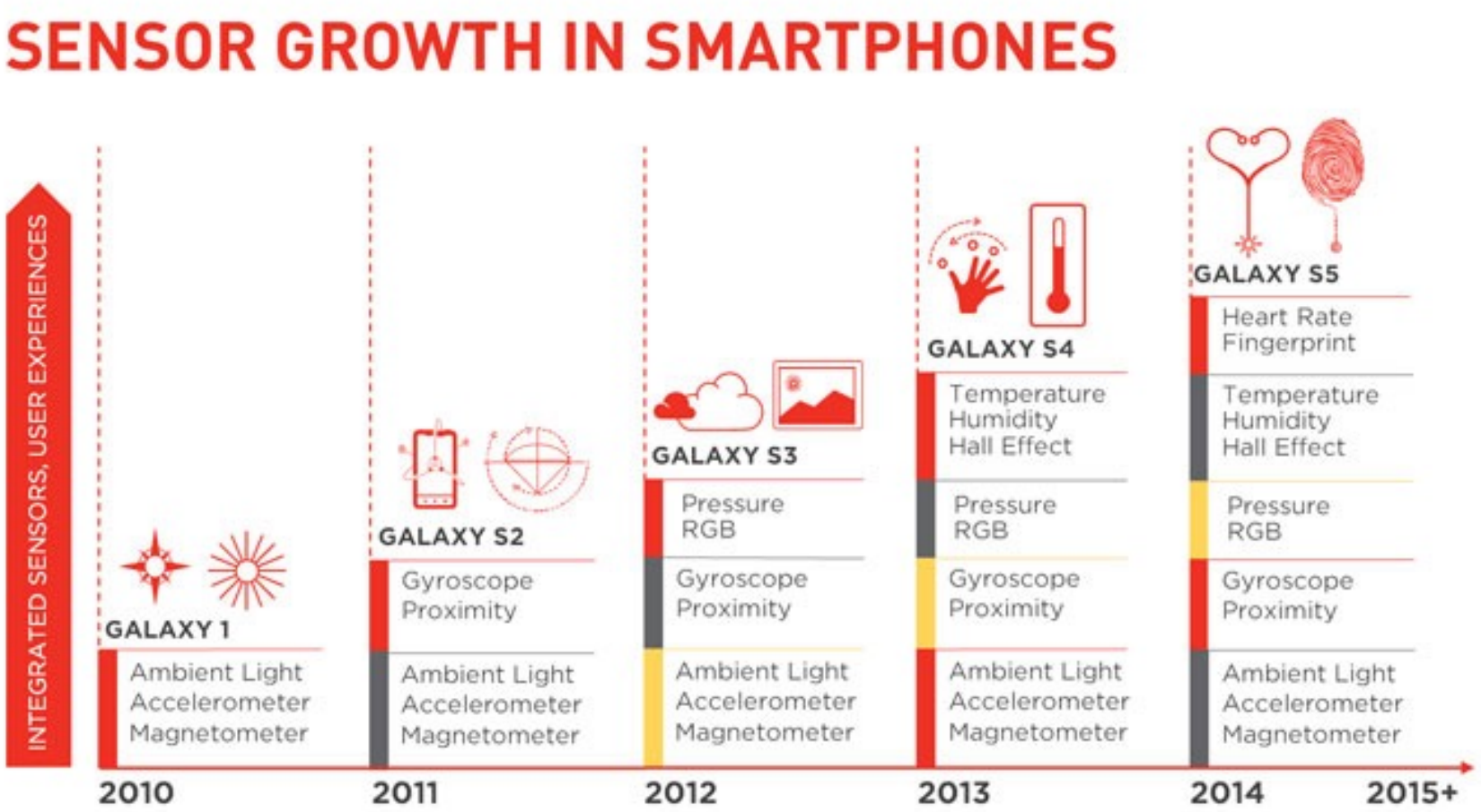
- There is a real concern that context information could either be disseminated inappropriately or disseminated to a component that is not completely trustworthy, and may disseminate the information further
- Developers of context-aware applications need to ensure that a user's privacy is maintained and that information is not being used inappropriately, or in a manner that the user feels inappropriate
- Make clear **why** some specific information is needed
  - E.g. the position of the user is needed to better estimate the amount of burned calories
  - Some systems (Android) show the user the permissions required by apps: statistics show that the ones which better explain why sensitive information is needed are less likely disinstalled

Issues when building contex-aware applications

Intelligibility of applications

- Context primarily consists of **implicit** input: more challenging for a user to understand that an application took some action based on nonexplicit input, to understand what nonexplicit input that was, and to understand what action was even taken
- Provide appropriate **feedback** to users to indicate that some action has been taken or why
  - Challenging for rules-based systems, but even more challenging for machine learning–based systems

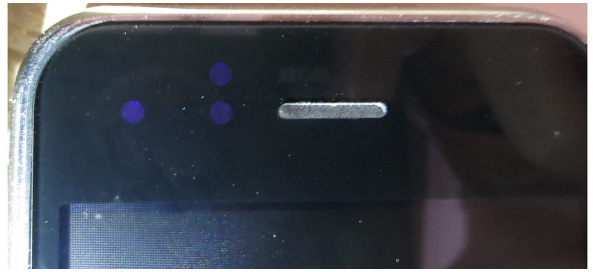
Sensing (context) with smartphones



Source: qualcomm.com

Sensing context: simple examples

- Screen orientation is changed as smartphone is rotated
- Smartwatch screen is turned on as user's arm is rotated and brought to eyes
- Screen brightness is automatically adjusted depending on the amount of environmental light
- The screen is automatically turned off when the phone is close to the face of the user

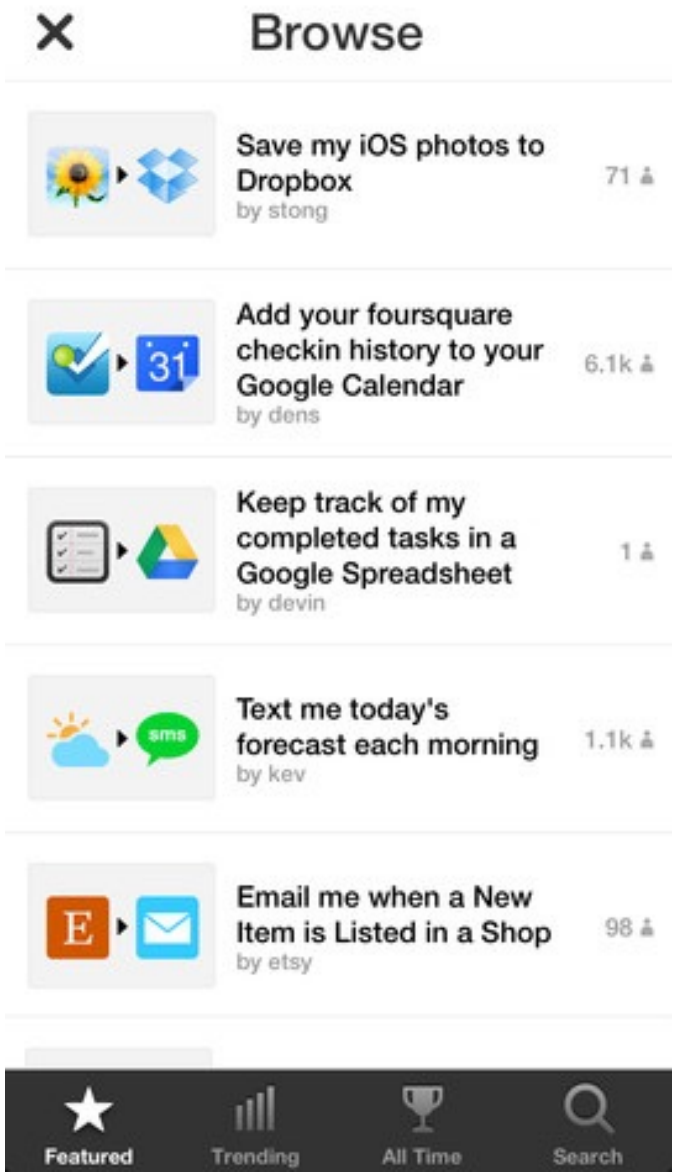


Sensing context: simple examples

- User’s location, directions from current position, shops in the nearby, bus stops, etc.

Rule-based actions

- If This Then That (IFTTT)



Benefits of using context

- Applications/systems can be pro-active
  - Adjust a car seat depending on the user
  - Automatic reminders
- Filter information
  - Stores that are far from current position are generally not relevant
- Intelligent environments
  - Turn heating off if the user is far from home
  - Discover and use nearby resources (e.g. speakers and screens)
- Reduce the cognitive load of users
  - Automatically turn off notifications when in meetings

References

- Mobile Computing CSE 40814/60814, U. of Notre Dame
- Ubiquitous Computing Fundmantals, Krumm ed, Ch. 8
- CS 528 Mobile and Ubiquitous Computing, WPI