Context

PHILIPPE LALANDA

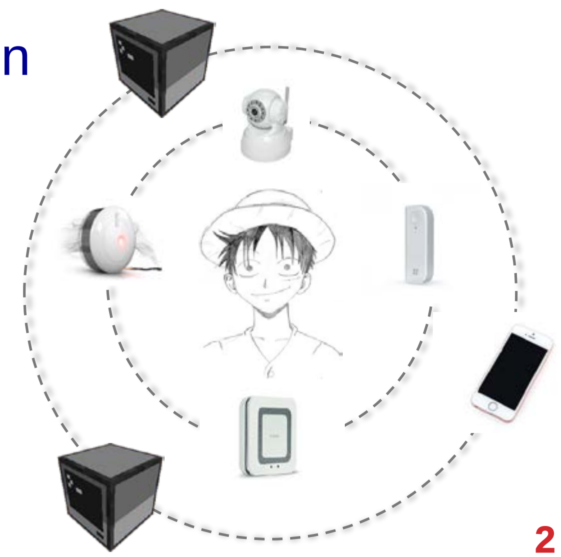
KOBE UNIVERSITY – AUGUST 2017

Pervasive computing

Pervasive computing promotes the integration of smart, networked devices in our living environments in order to provide us services.

Those services

- are context aware
- require minimal and natural interaction
- bring real added value
- are easy to administrate by end-users



Purpose of this lecture

Our goal is to:

- define the notion of context in pervasive computing
- define the notion of context-aware applications
- show how to structure and build a context
- develop different modeling techniques

Agenda

Example

Definitions

Global architecture

Context building

Challenges and conclusion

Example: “Keeping in touch”

Management of communication means with your family and friends

- Phone calls
- Messages
- Photos
- Coordinating activities
- Being aware of activities of friends and family

Problems with Keeping in Touch

Irrelevant messages

- Vacation mail, junk email

Interruptions

- During meetings, movies, dinner, driving

Lack of awareness on callee side

- Phone tag, time zone issue (oops!)

Information overload

- Can make it hard to find useful messages

The goal is to provide the right call/message at the right time and under the right form

5 Design Considerations

1. **Improving relevance**
2. **Minimizing disruption**
3. **Improving awareness**
4. **Reducing overload**
5. **Selecting channels**

5 Design Considerations

1. **Improving relevance**
   - Deciding when a communication is relevant to the person's current (or near future) situation.
   - For example, getting notification about an email from your travel agent regarding itinerary changes while packing to leave for the airport.
2. **Minimizing disruption**
3. **Improving awareness**
4. **Reducing overload**
5. **Selecting channels**

5 Design Considerations

1. **Improving relevance**
2. **Minimizing disruption**
   - Deciding when and how to notify people that they have a communication.
   - For example, your phone should vibrate and not ring, when you are at the symphony (unless it is truly urgent).
3. **Improving awareness**
4. **Reducing overload**
5. **Selecting channels**

5 Design Considerations

1. **Improving relevance**
2. **Minimizing disruption**
3. **Improving awareness**
   - Deciding what information and mechanisms can help people make intelligent communication decisions.
   - For example, the caller should be told you are at the movies before the call goes through.
4. **Reducing overload**
5. **Selecting channels**

5 Design Considerations

1. **Improving relevance**
2. **Minimizing disruption**
3. **Improving awareness**
4. **Reducing overload**
    - Deciding how to reduce the number of communications that don’t apply given your context.
    - For example, filtering out emails about going to lunch when you are away from the office (or already at lunch).
5. **Selecting channels**

5 Design Considerations

1. **Improving relevance**
2. **Minimizing disruption**
3. **Improving awareness**
4. **Reducing overload**
5. **Selecting channels**
   - Deciding which communication device should be used to get in touch with somebody.
   - For example, routing calls to your home phone instead of your cell phone when you are at home and cellular reception is poor.

To do so…

The “Keeping-In-Touch” application has to be aware of its context of execution

Then, it must adapt its behavior in order to provide the expected service

This is the essence of pervasive computing, but rather hard to achieve

Agenda

Example

Definitions

Global architecture

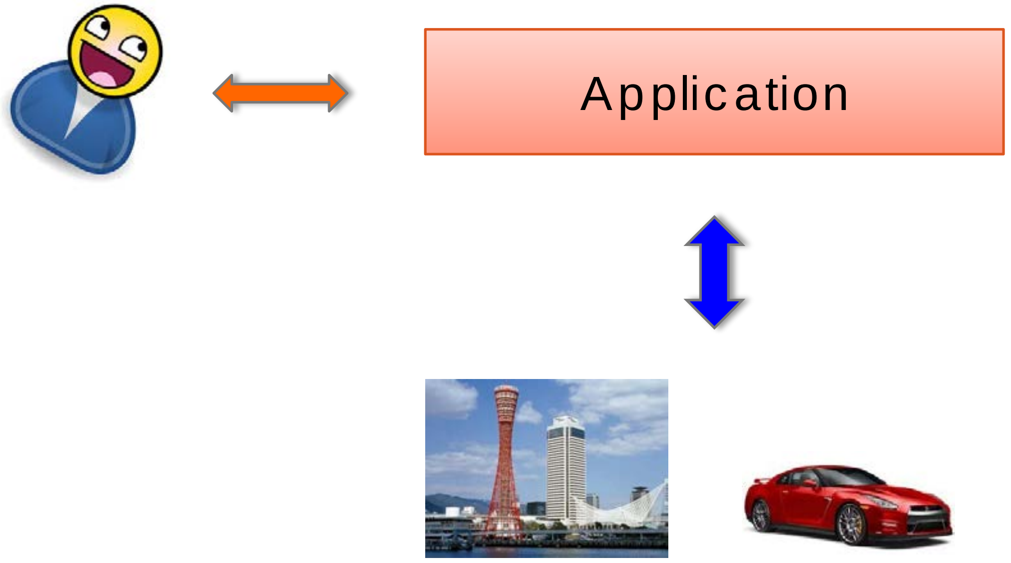
Context building

Challenges and conclusion

Context definition

Dey, 2001

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



Philippe Lalanda

Examples

- User identity
- Spatial information (location, orientation, speed, acceleration)
- Temporal information (time, date, season)
- Environmental information (temperature, air quality, light, noise)
- Social situation (who you are, who you are with, people nearby)
- Resources that are nearby (accessible devices, networks, hosts)
- Availability of resources (battery, display, network, bandwidth)
- Physiological measurements (blood pressure, heart rate, respiration rate, muscle activity, tone of voice)
- Activity (talking, reading, walking, running, sleeping)
- Schedules and agenda

Classification

Computing context

Application itself, network connectivity, communication bandwidth, nearby resources like printers, display, etc.

User context

People nearby, user’s profile, location, emotional state,, current activity

Physical context

Objects, locations, lighting, noise, traffic, conditions, temperature, etc.

Primary versus secondary context

Primary context (low level information)

Any information retrieved without using existing context and without performing any kind of sensor data fusion operations (e.g. GPS sensor readings as location information)

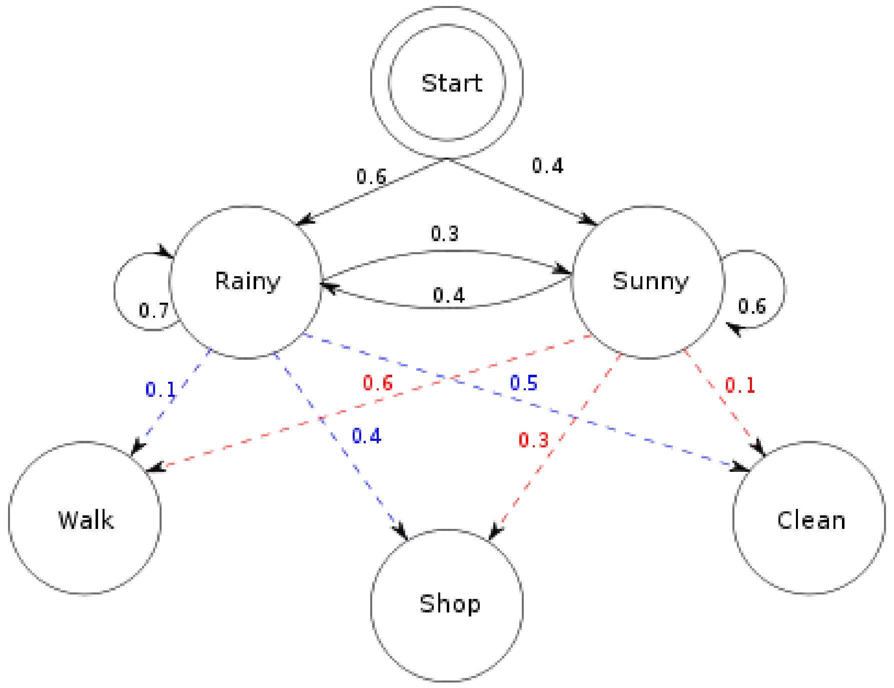
Secondary context (High level information)

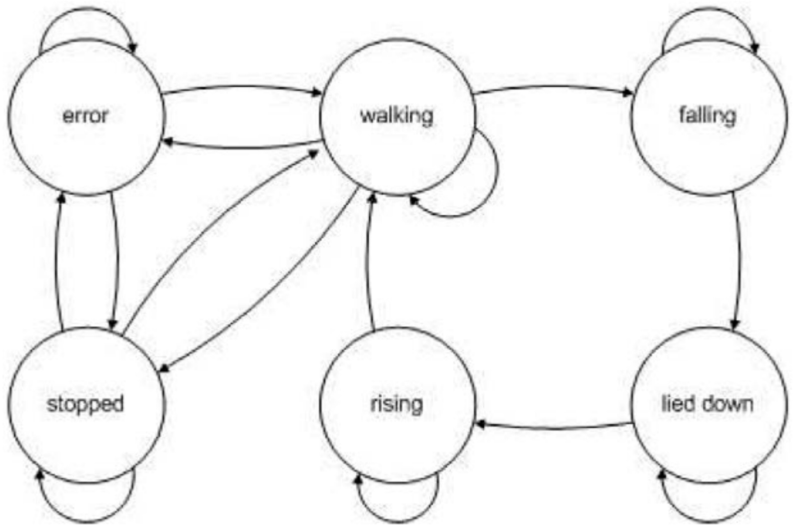
Any information that can be computed using primary context. The secondary context can be computed by using sensor data fusion operations or data retrieval operations such as web service calls

Example: activity

The activity of a person belongs to secondary context

- Computed from multiple sensors values
- Use of probabilistic models (Markov)





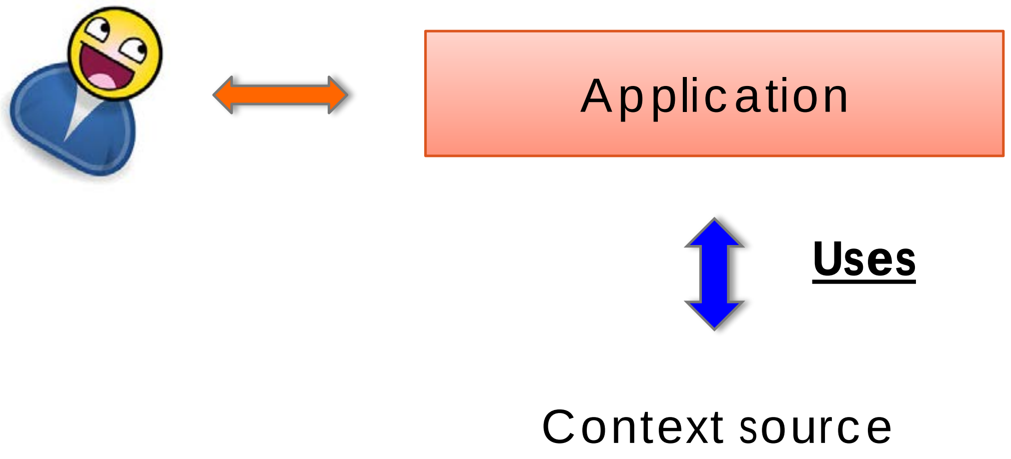
Context-aware application

Abowd, 2009

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task

Example

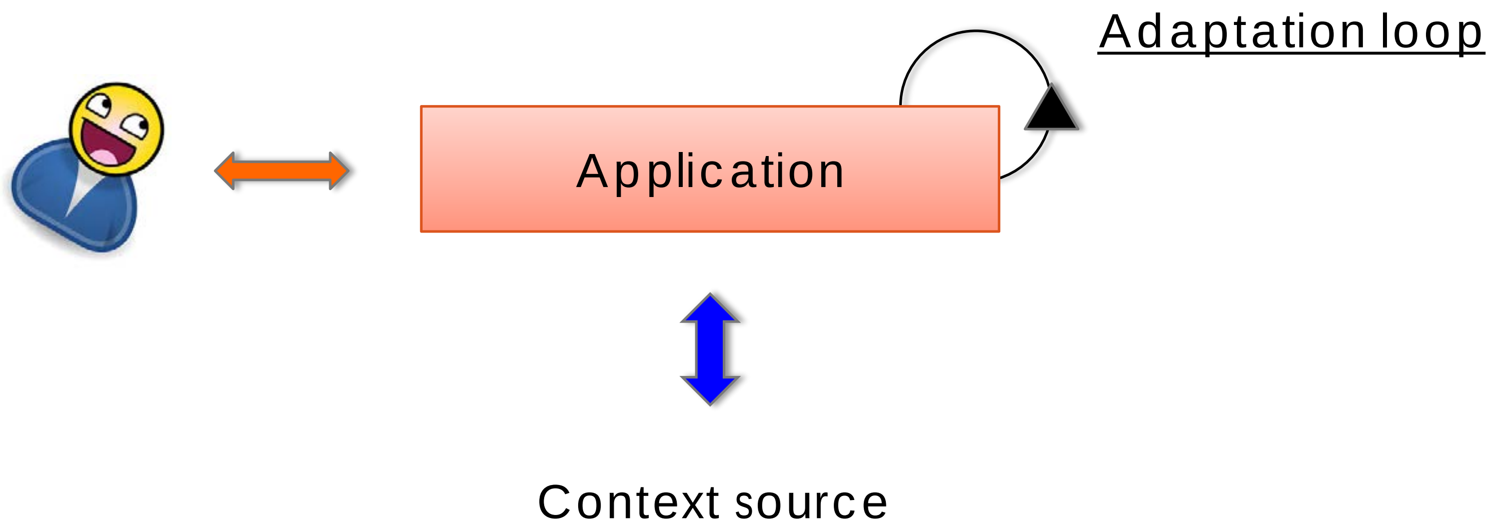
Smart phones screen goes brighter when exposed to light (using photo sensors) and goes dimmer on low battery.



Context-aware application - implication

Context-aware applications adapt according to the context in order to provide better services

Delivering the right service at the right moment



Level of context-awareness

Personalization

It allows the users to set their preferences and expectation. For example, users may set the preferred temperature so the heating system can maintain the specified temperature.

Passive context-awareness

The system monitors the environment and offers the appropriate options to the users so they can take actions. For example, when a user enters a super market, the mobile phone alerts the user with a list of discounted products.

Active context-awareness

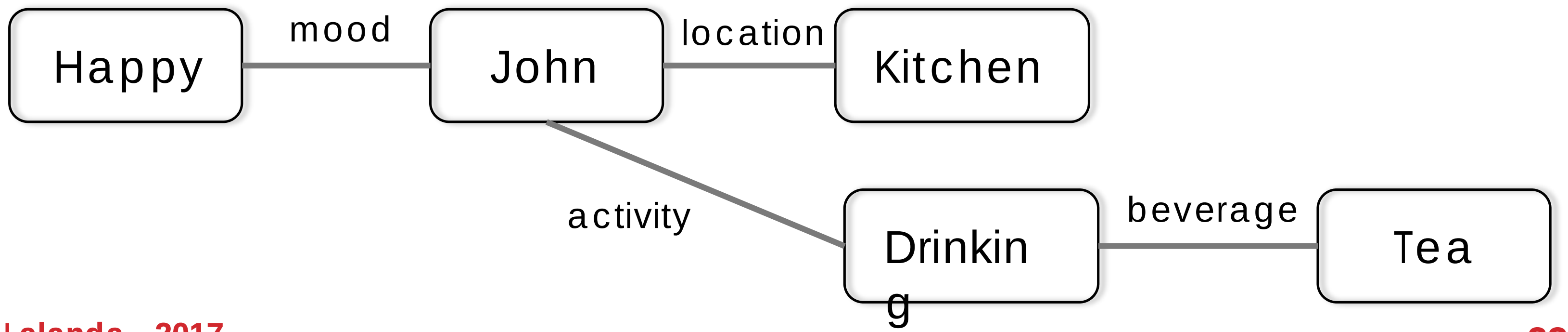
The system monitors the situation and acts autonomously. For example, if the smoke detectors and temperature sensors detect a fire, the fire brigade is called by the system.

Context model

Henricksen, 1999

A context model identifies a concrete subset of the context that is realistically attainable from sensors, applications and users and able to be exploited in the execution of the task.

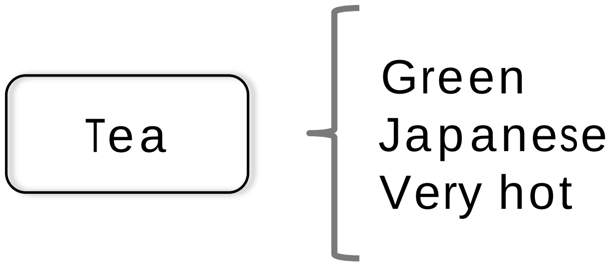
The context model that is employed by a given context-aware application is usually explicitly specified by the application developer, but may evolve over time"



Context attribute

Henricksen, 1999

A context attribute is an element of the context model describing the context. A context attribute has an identifier, a type and a value, and optionally a collection of properties describing specific characteristics



Context quality

A number of definitions and parameters that have been proposed.

Important parameters

- data validity
- context data precision
- context data up-to-dateness (freshness)

The context quality depends on the quality of the physical sensors, and quality of the delivery process

Synthesis

Context is a major source of information for pervasive applications

- It enables to manage the vast amount of information that surrounds the user
- It allows to discriminate what is important and what is not
- It helps us to adapt to our surroundings

But context

- is imperfect
- has temporal dimensions
- is hard to capture, represent and manage

Agenda

Example

Definitions

Global architecture

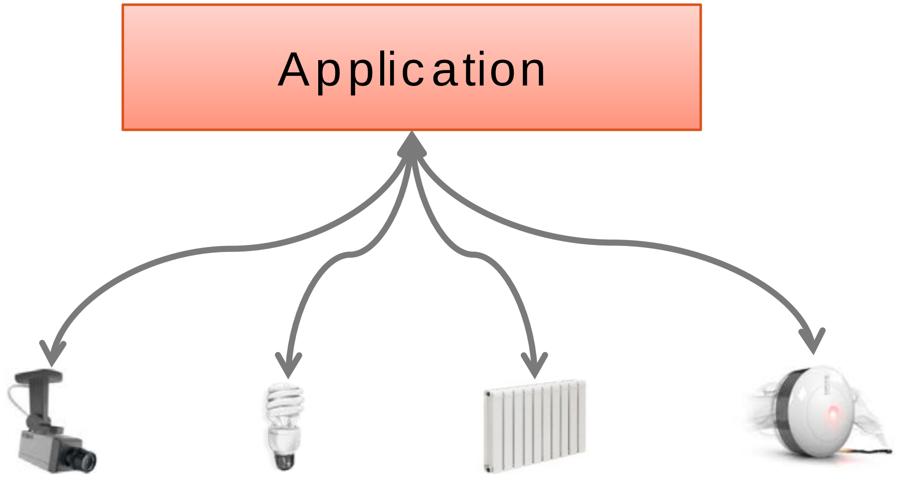
Context building

Challenges and conclusion

Getting contextual information - 1

Directly from sensors

- Context is directly acquired from the sensor
- Software drivers and libraries need to be installed locally
- This method is typically used to retrieve data from sensors attached locally



Benefits and limits

Efficient as it allows direct communication with the sensors

Have more control over sensor configuration and data retrieval process

Significant technical knowledge is required about devices internals and their configuration

Significant amount of time, effort, cost involved

Updating is very difficult due to tight bound between sensor and application

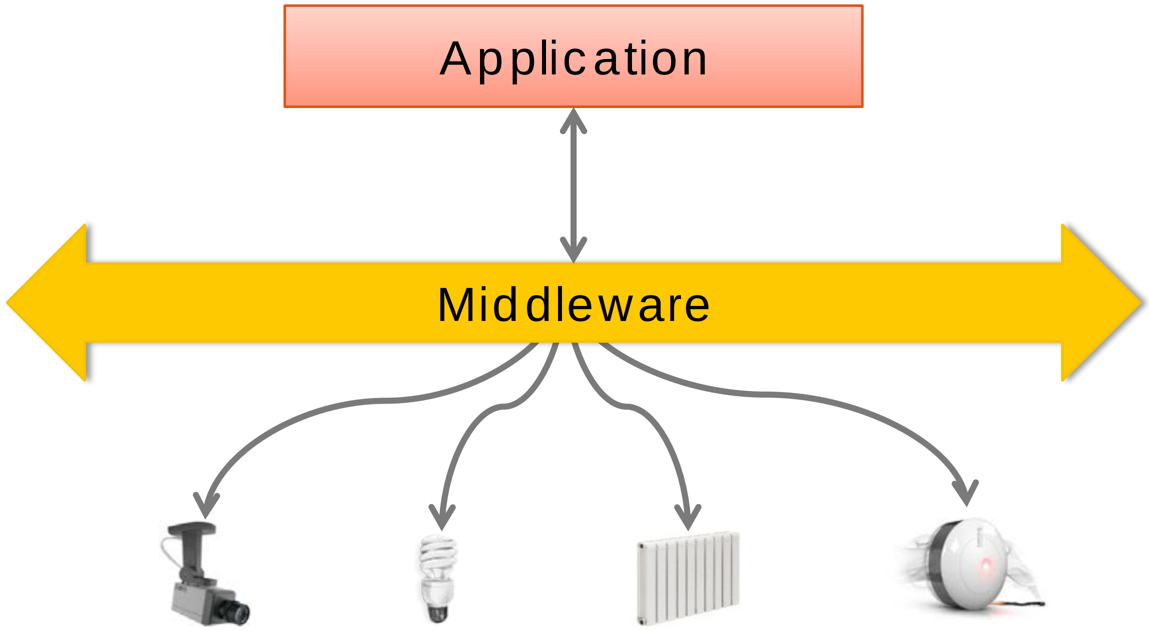
Can be used for small scale scientific experiments. Can also be used for situation where limited number of sensors are involved.

Getting contextual information - 2

With a middleware

- sensor data is acquired by middleware solutions.
- The applications retrieve sensor data from the middleware and not from the sensor hardware directly

Ex: publish-based



Benefits and limits

Easy to manage and retrieve context as most management tasks are facilitated by the middleware.

Can retrieve data faster with less effort and technical knowledge

Require more resources (processing, memory, storage) as middleware solutions need to be employed

Less control over sensor configuration

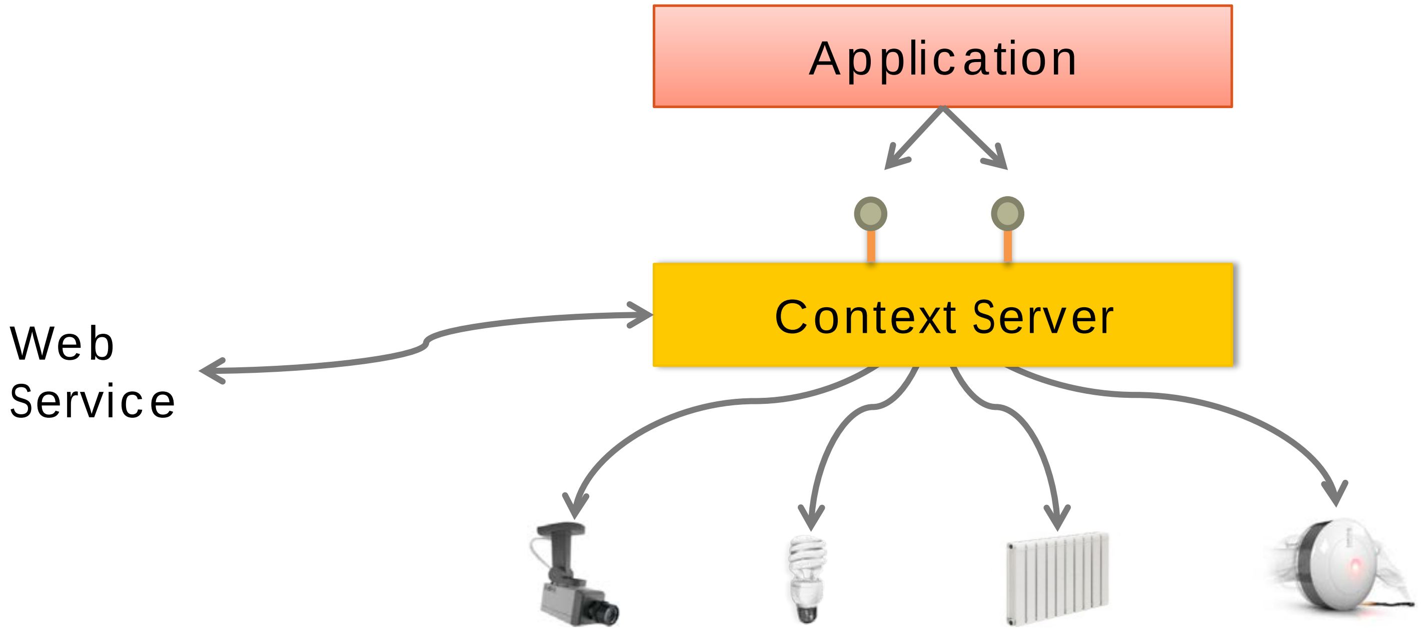
Moderately efficient as data need to be retrieve through middleware

Pervasive application will use this methods in most cases. Can be used in situations where large number of heterogeneous sensors are involved (on local network).

Getting contextual information - 3

With a context server

- Specific module in charge of acquiring context, from several sources
- Provides an API to access contextual information
- Can be distributed (if resources are scare for instance)



Benefits and limits

Less resources required

Can retrieve data faster with less effort and technical knowledge

No control over sensor configuration

Less efficient as the context need to be pulled from server over the network

Can be used in situations where significant amount of context are required but have only limited resources (i.e. cannot employ context middleware solutions due to resource limitations) that allows run the consumer application.

Frameworks

Principles related to context management frameworks

- Supporting technology
- Application programming interface (API)
- Event management
- Extensibility
- Scalability
- optimization

Agenda

Context and context-awareness

Example

Global architecture

Context building

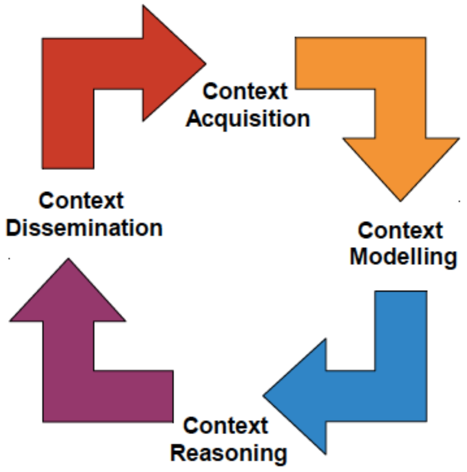
- structure
- acquisition
- modeling

Challenges and conclusion

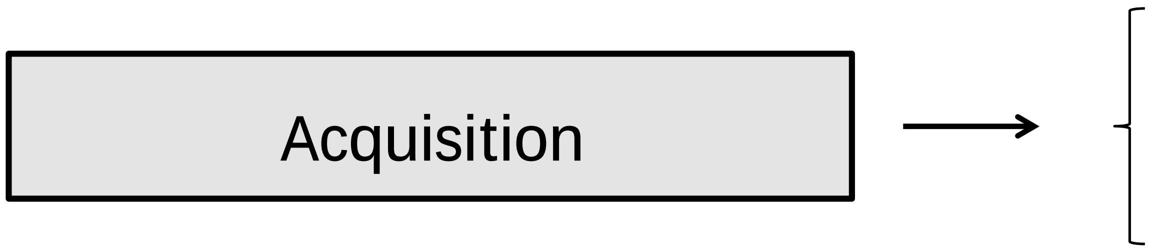
Building context-aware applications

Regardless of the global architecture, building a context requires to

- Acquire the necessary data
- Model and abstract the captured information
- Possibly, reason about the information
- Make the modeled information available to application
- (it is then the task of applications to adapt themselves)

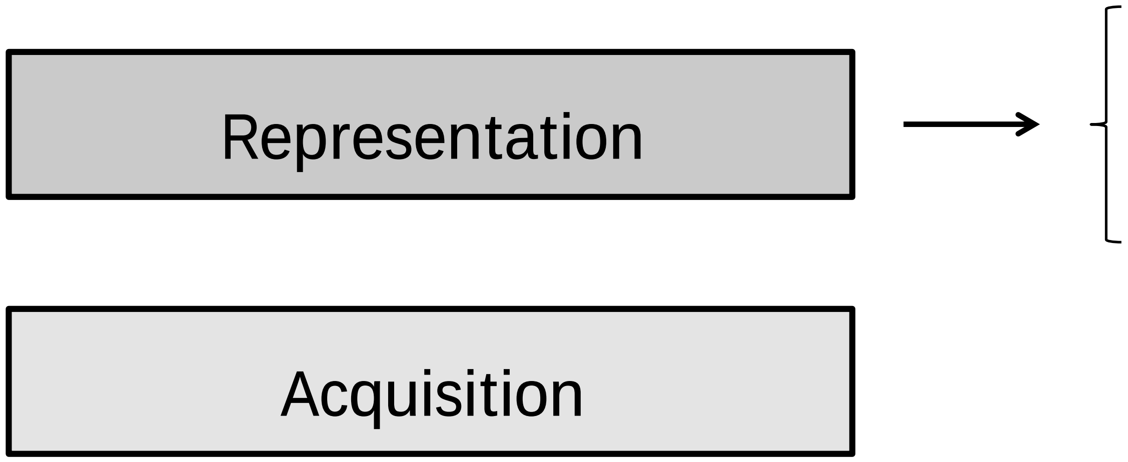


Layered approach



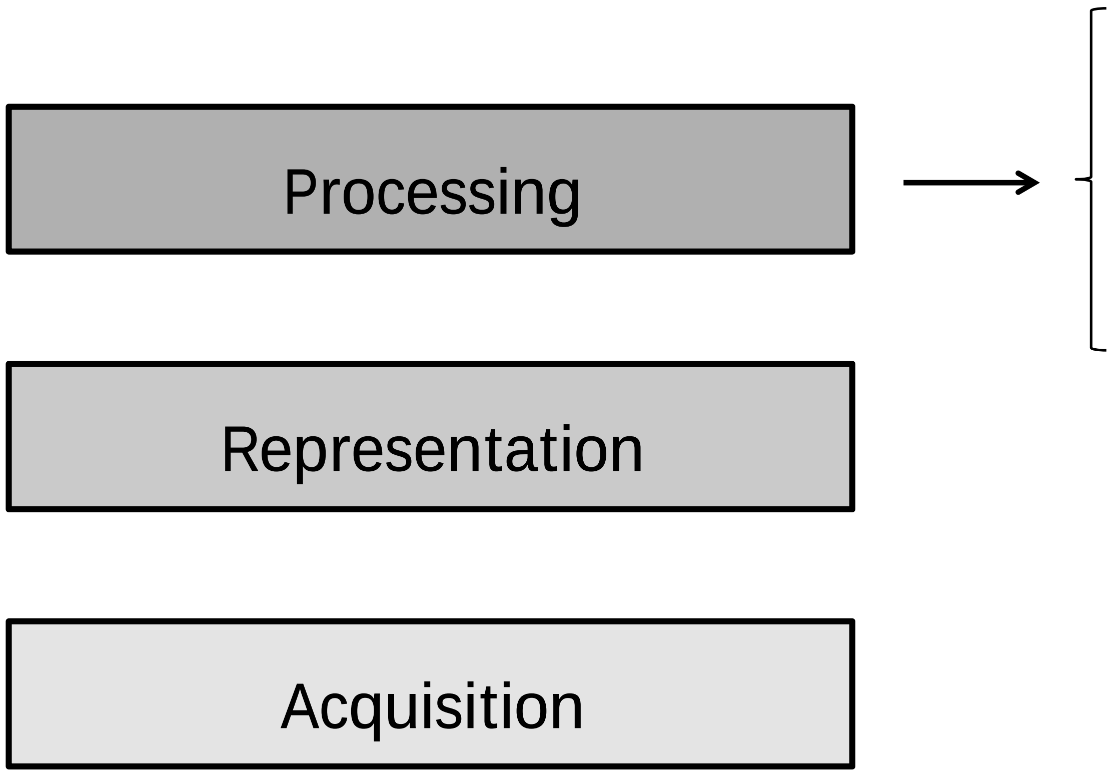
- Acquisition
- Single/Double way
- Synchronization

Layered approach



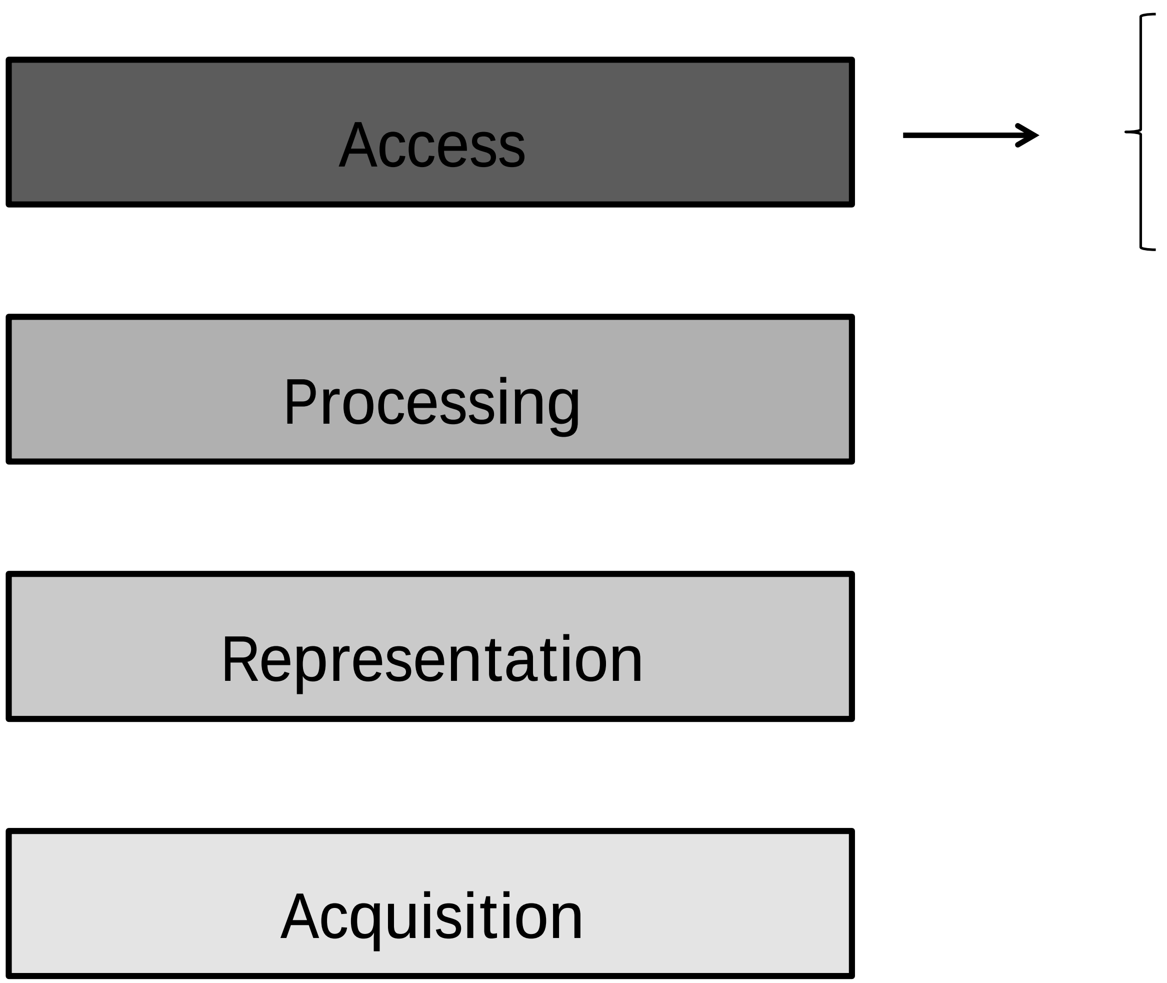
- Model
- Storage
- Historic

Layered approach



- Aggregation
- Mediation
- Inference
- Enrichment

Layered approach



- Query
- APIs
- Event

Agenda

Context and context-awareness

Example

Global architecture

Context building

- structure
- acquisition
- modeling

Challenges and conclusion

Context acquisition: responsibility

Pull

The software component which is responsible for acquiring sensor data from sensors make a request (e.g. query) from the sensor hardware periodically (i.e. after certain intervals) or instantly to acquire data

Push

The physical or virtual sensor pushes data to the software component which is responsible to acquiring sensor data periodically or instantly. Periodical or instant pushing can be employed to facilitate a publish and subscribe model

Context acquisition: frequency

Instant (threshold violation)

These events occur instantly. The events do not span across certain amounts of time. Open a door, switch on a light, or animal enters experimental crop field are some types of instant events. In order to detect this type of event, sensor data needs to be acquired when the event occurs. Both push and pull methods can be employed.

Interval

These events span a certain period of time. Raining, animal eating a plant, or winter are some interval events. In order to detect this type of event, sensor data needs to be acquired periodically (e.g. sense and send data to the software every 20 seconds). Both push and pull methods can be employed.

Context acquisition: process

Sense

The data is sensed through sensors, including the sensed data stored in databases

Derive

The information is generated by performing computational operations on sensor data. These operations could be as simple as web service calls or as complex as mathematical functions run over sensed data (e.g. calculate distance between two sensors using GPS coordinates)

Manually provided

Users provide context information manually via predefined settings options such as preferences (e.g. understand that user doesn't like to receive event notifications between 10pm to 6.00am).

Agenda

Context and context-awareness

Example

Global architecture

<u>Context building</u>

- structure
- acquisition
- modeling

Challenges and conclusion

Context modeling

Extensive research and proposals on that topic

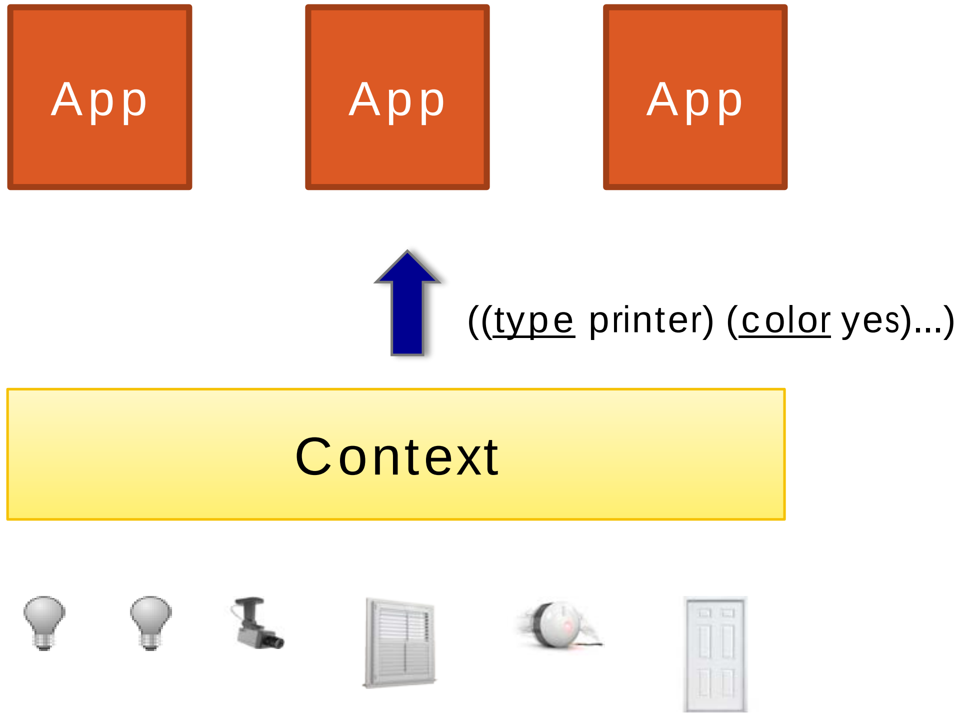
Let us examine the us of

- Key value
- Markup
- Objects (programming)
- Databases (relational)
- Logic
- Ontologies
- Entity/relation

Key-value based context

Contextual information is sent to applications through lists of key-value pair

- new list is sent when information is updated or when new information is captured (mobility)
- Applications provide means to receive the information and interpret it



Key-value - Benefits and limits

Simple

Flexible

Easy to manage when small applications

Strongly coupled with application

Not scalable

Not structured (no schema)

Hard to retrieve information

No way to represent relationships

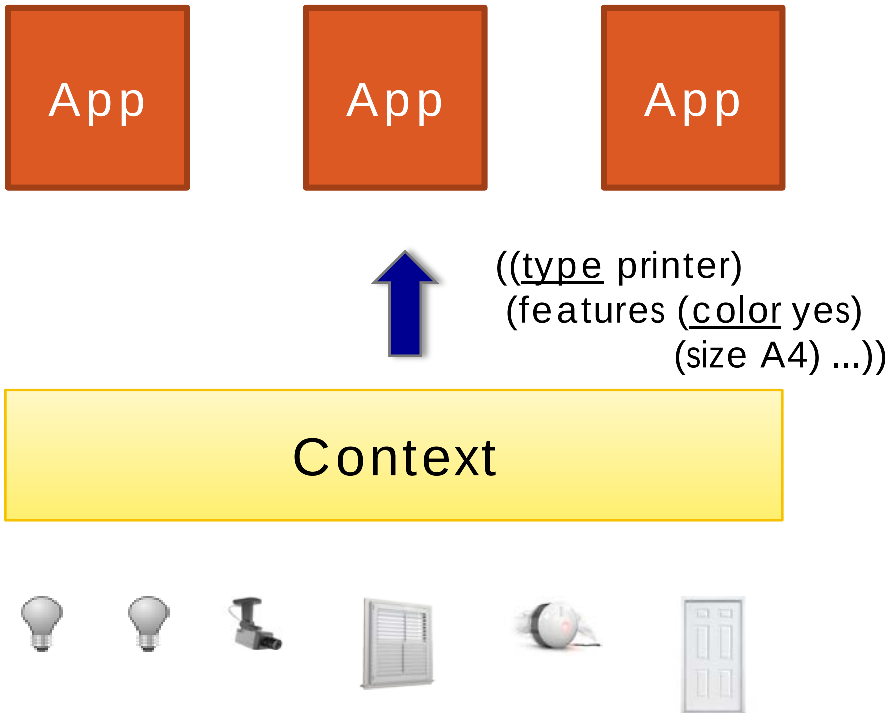
Can be used to model limited amount of data such as user preferences and application configurations. Mostly independent and non-related pieces of information.

This is also suitable for limited data transfer and any other less complex temporary modeling requirements.

Markup-based context

Contextual information is sent to applications through XML files (with keywords and nested lists)

- new file is sent when new or updated information
- Applications provide means to receive the information and interpret it



Markup - Benefits and limits

Flexible

More structured

Validation possible (schemas)

Tools available

Strongly coupled with application

Not scalable

Can be complex when many levels of information are involved

Information retrieval not always easy

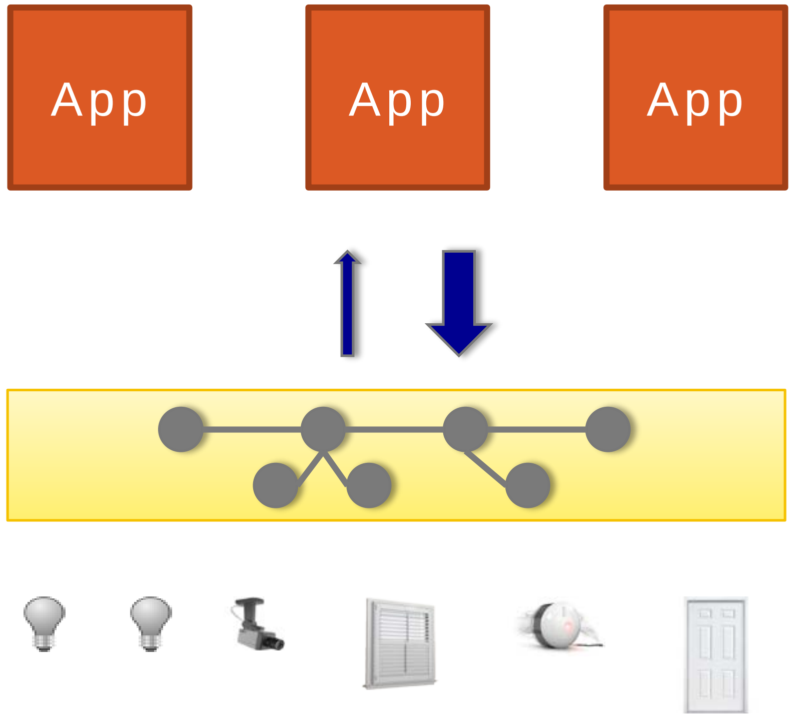
Can be used as intermediate data organization format as well as mode of data transfer over network. .

Can be used to decouple data structures used by two components in a system.

Object-based context

Contextual information is provided through object-oriented APIs

- Applications have to get the information
- Events can be sent when something changes (can be complex)



Objects - Benefits and limits

Allow relationships modeling

Can be well integrated with programming languages

Tools available

Hard to retrieve information

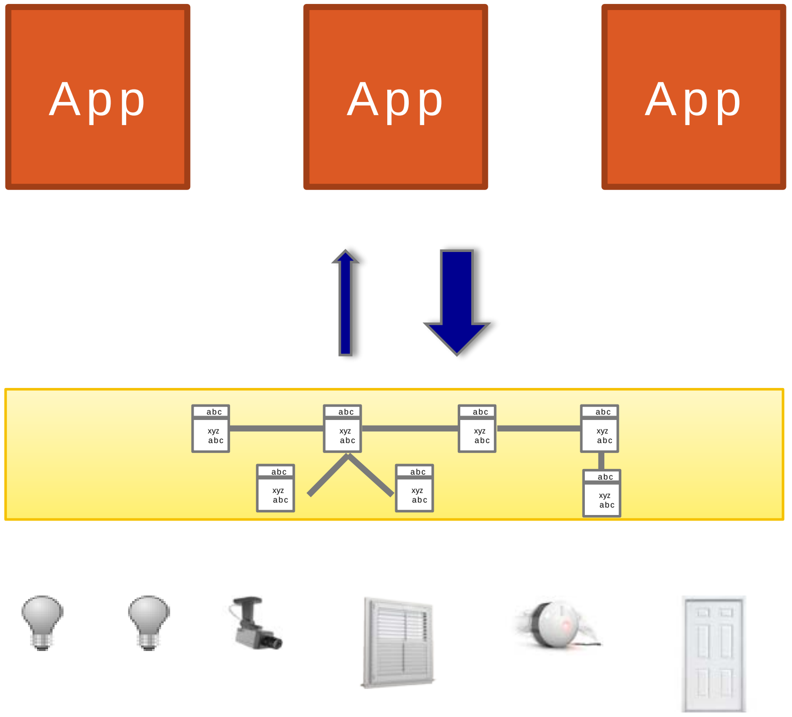
No standards but govern by design principle

Can be used to represent context in programming code level. Allows context runtime manipulation. Very short term, and mostly stored in computer memory.

Database context

Contextual information is provided through request server

- Applications have to get the information (with SQL in general)
- Events can be sent when something changes (extremely complex)



Database - Benefits and limits

Allow relationships modeling

Based on universal APIs

Different standards (for data representation) available

Processing tools available

Querying can be complex

Configuration may be required

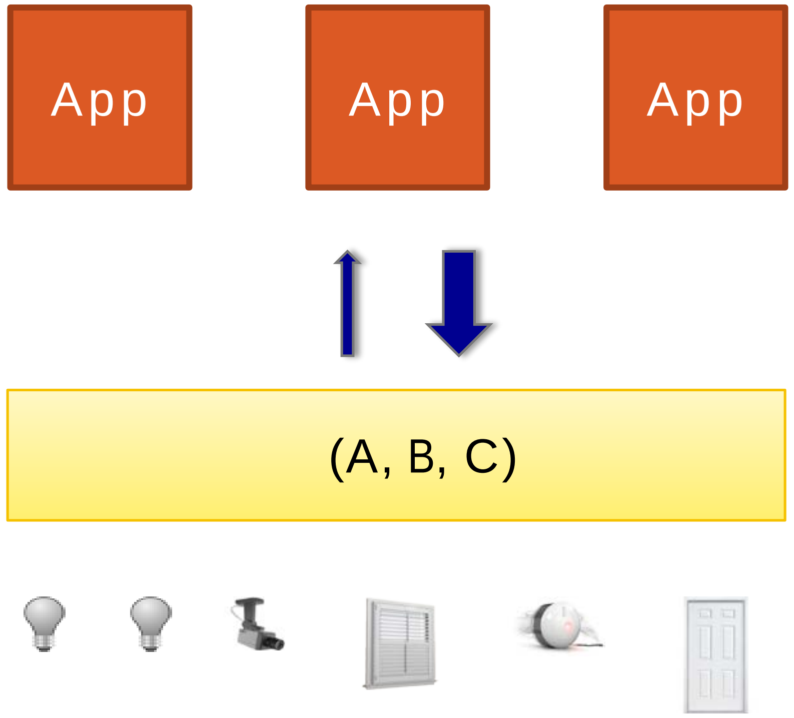
No standards but govern by design principle

Can be used for long-term and large volume of permanent data. Historic context can be used in database context.

Logic-based context

Contextual information is provided as facts (and sentences)

- Applications have to get the information
- Events can be sent when something changes (complex)



Logic- Benefits and limits

Allow to generate knowledge

Support logical reasoning

Processing tools available

Strongly coupled with applications

No standard

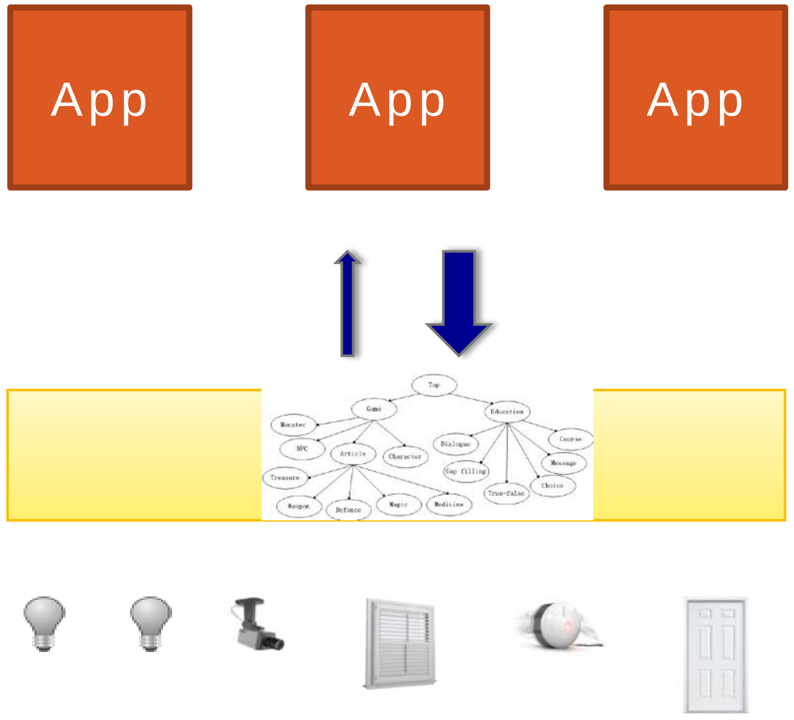
Hard to use

Can be used to <u>generate</u> high level context using low level context, model events and actions.

Ontology-based context

Contextual information is provided as a semantic graph

- Applications have to get the information
- Events can be sent when something changes (complex)



Ontology - Benefits and limits

Support semantic reasoning

Allow expressive representation of context

Application independent

Strong support by standards

Sophisticated tools available

Representation can be complex

Information retrieval can be complex and resource intensive

Reasoning can be long

Can be used to model domain knowledge and structure context based on the relationships defined by the ontology. Rather than storing data on ontologies (slow and hard to access), data can be stored in appropriate data sources (databases) while structure is provided by ontologies.

Agenda

Example

Definitions

Global architecture

Context building

Challenges and conclusion

Conclusion

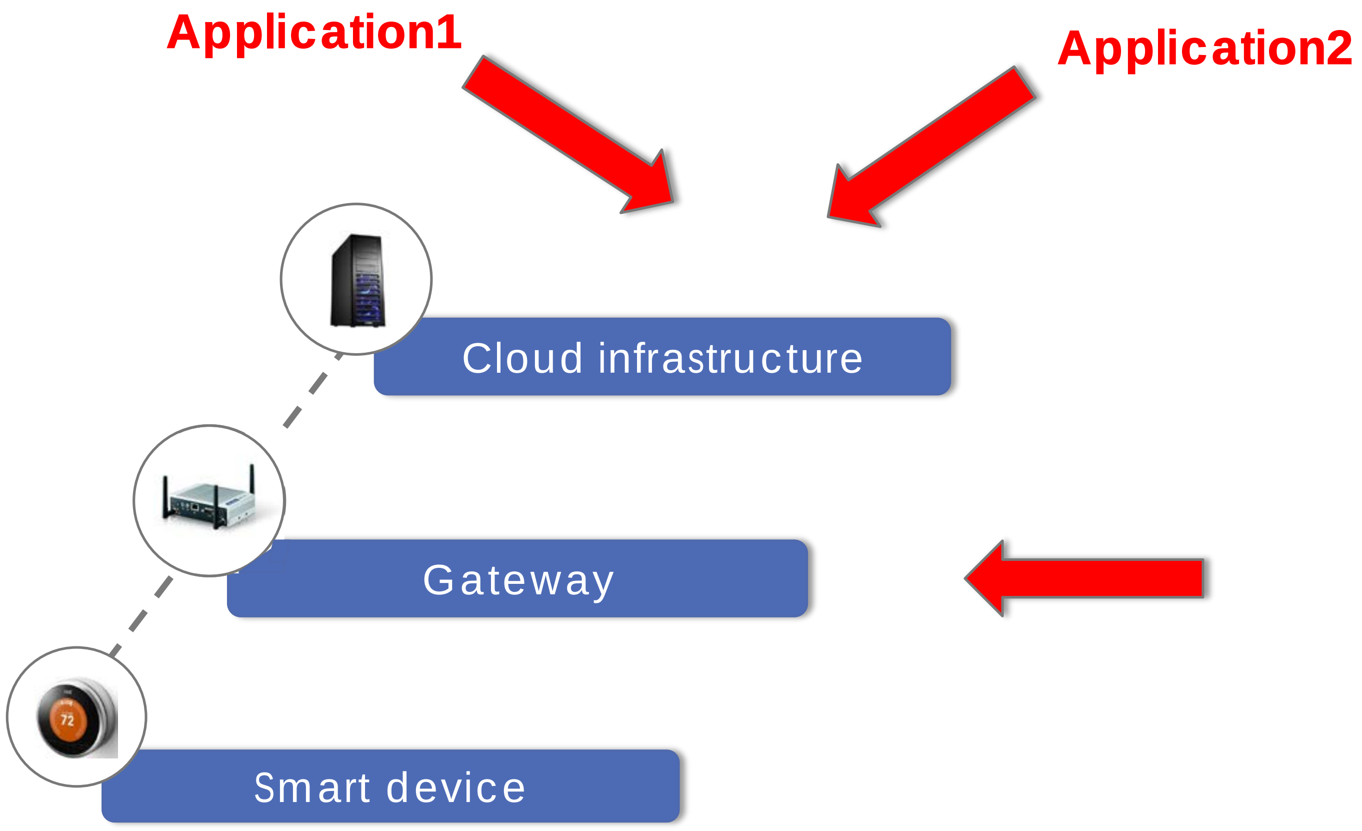
Different definitions of context available, but a common (intuitive) understanding

Important aspects

- Identity and location
- Activity
- Time
- Available resources

Different ways to represent context

Multiple contexts



Main goal

Build the context for the the application