Lecture 11

Context Awareness of mobile applications

A. Context

- Context: all the elements that are significant in a given situation and can influence, for example, the execution of mobile applications; *apps become context-aware*:
  - device context;
  - user context;
  - environment context.

- Device-user-environment interactions create rich contextual information.
- We need a framework on how context information is acquired and delivered in a reliable, secure and with the right level of abstraction.

Generic Information Sources

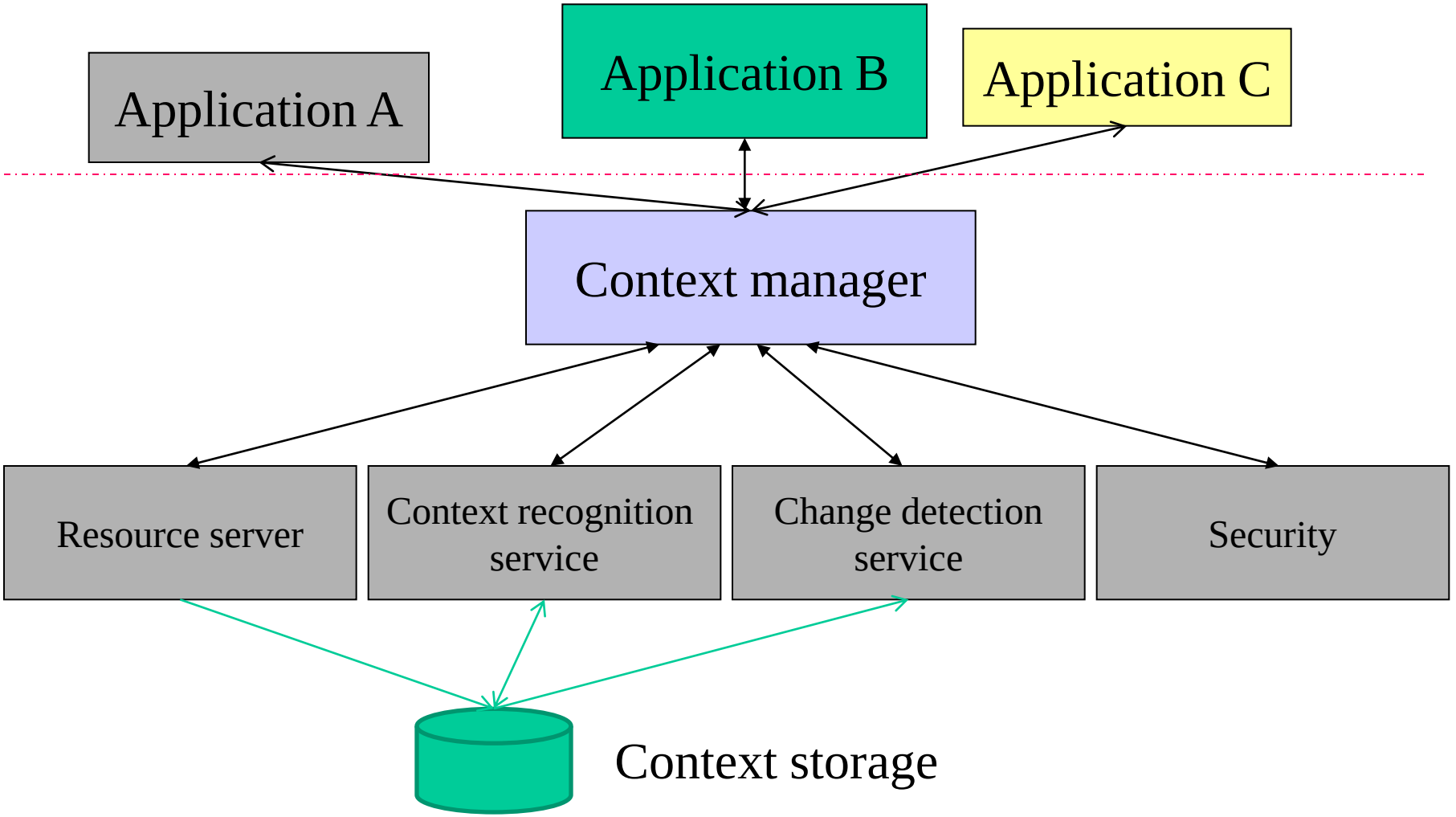
- Sensors: temperature, humidity, touch, microphone, channels for light, accelerometers, GPS units etc. Sensors can be internal, centralised external sensors, or ad-hoc external sensors, over Bluetooth and WLAN connections.
- Mobile device: running applications, processes, battery, other resources.
- User: activities/profiles, preferences, policies.
- Time, location.
- Internet, other networks.

- Rules:
  - the delivered data's abstraction level should be high enough;
  - the frequency should be low enough to be useful.

- Challenges:
  - There can be multiple and heterogeneous sources of interest providing data that may change rapidly and it is only partial.
  - Extracting relevant context information by fusing data from several sources is a complex task.

- Multidimensional data is used to create context descriptions.

B. The context framework



The context manager roles

- It creates and provides models of the context. This involves context reasoning:
  - *context inference* maps low-level data to high-level data;
  - *sensor data fusion* for reducing communication.
- It answers queries and subscriptions from applications.
- It provides a plug-in context registration service that allows applications to have access to the context models data.

The context manager work

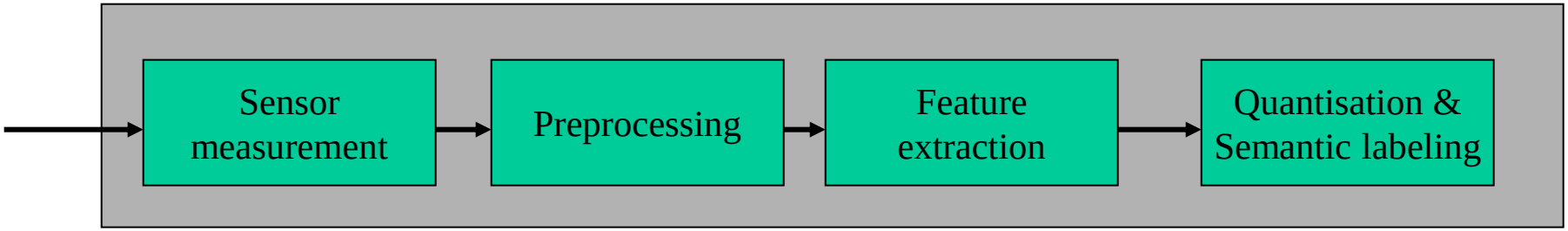
- A naïve Bayse classifier recognizes higher-level contexts from lower-level context atoms.

- The classifier that works within the context manager can serve as a plug-in context recognition service that takes a set of context atoms as input from the context manager, performs the classification, and returns a higher-level context to the manager.

- The manager stores the higher-level context produced by the recognition service and delivers it to clients, based on subscriptions.

The resource server

- The resource server connects to any context data source and outputs context information.



The pre-processing stage builds measurement data arrays that store a certain amount of samples and calculates generic features for each time interval.

Feature extraction produces more specific context features. The last stage binds the feature values to the real-world context, which has a meaning for a user/application.

The recognition service

- The quantisation phase produces context atoms (base context units) that applications can use, or recognition services can further refine.
- The resource server can use one of two methods for quantisation:
  - set crisp limits, resulting in a true/false labeling of context atoms (e.g, sound intensity = silent/moderate/loud);
  - use a fuzzy set that produces continuous valued fuzzy labeling (e.g., silent/0.6 + moderate/0.3 + loud/0.1).

- The context recognition service uses as input either a set time-stamped or a time series of context atoms and returns a single higher-level context.

C. The context ontology

- The ontology consists of a set of axioms. The axioms can be used to express relationships between entities. For example, a "Father" entity, could have a "hasDaughter" relationship, which points at a “Daughter" entity. A Context Reasoner (Inference Engine) can parse the axioms to make inferences about what is true and false.

- An ontology is an engineering artifact:
  - it has a specific vocabulary used to describe a certain reality,
  - and a set of axioms (explicit assumptions) regarding the intended meaning of the vocabulary.
- The aim is that an ontology describes a formal specification of a domain:
  - shared understanding of a domain of interest;
  - formal and machine manipulable model of a domain of interest.

- RDF – Resource Description Framework can be used as the description syntax ➔ there is a common structure for representing information.

- Each context can be described by several properties: context type, context value, confidence (optional property that refers to context's uncertainty), source, timestamp, attributes.

- The schema allows creating and handling composite contexts and context hierarchies.

Building ontologies

- determine how the world (domain) should work:
  - determine the classes and properties in the domain;
  - determine domains and ranges for properties;
  - determine characteristics of classes;
  - add individuals and relationships as necessary (some individuals belong here);
  - iterate until “good enough”;
  - package all this into an ontology;

  - is the ontology consistent?
  - are the classes coherent?

Context representation

- How to represent, store and infer context are key questions.
- Two approaches are significant:
  - the web ontology language (OWL);
  - the object-oriented model (JCAF).
- JCAF represents context in an Object Oriented manner. JCAF is an actual Java program that one runs on a system. The framework features an Entity class, a Context class, a ContextItem class, and a Relationship class.
- Each user of the system could be represented as an instance of the Entity class. Each Entity has one Context object, and the Context object contains several ContextItem objects, which represent the contextual data. Each ContextItem object is indexed by a Relationship object, describing the relationship the ContextItem has with the Entity.

The Web Ontology Language (OWL)

- W3C standard for representing ontologies in the semantic web.
- The OWL API can be used to represent context as a document, marked up in XML. It supports various different OWL syntaxes such as the RDF syntax.
- An Ontology contains various different types of entities and axioms, which can be used to represent user context.
- OWL is made up of classes, properties (object and data properties), and individuals (named or anonymous), which are used to represent the entities and axioms in an ontology.
- Different entities are bound together by logical axioms, which can be used to infer relationships, in the form of linked data.

The Web Ontology Language (OWL)

- W3C standard for representing ontologies in the semantic web.
- The OWL API can be used to represent context as a document, marked up in XML. It supports various different OWL syntaxes such as the RDF syntax.
- An Ontology contains various different types of entities and axioms, which can be used to represent user context.
- OWL is made up of classes, properties (object and data properties), and individuals (named or anonymous), which are used to represent the entities and axioms in an ontology.
- Different entities are bound together by logical axioms, which can be used to infer relationships, in the form of linked data.

Context inference

- The OWL API provides an OWLReasoner interface, which is implemented by an Inference Engine.

- Using the OWLReasoner, one can query an ontology in a document, by specifying for what data or object property should be returned, and the values for those properties are received back from the reasoner as a collection of literal values (Integers, Floats, Strings).

- JCAF does not support any form of context inference in its current form.

Context storage

- The OWL API has the ability to write out ontologies into XML files, marked up in various supported syntaxes. These files can be loaded by the OWL API for parsing, reading, and writing. The files themselves can be stored on any cloud based file system for access by the Context Processor.

- JCAF does not use any form of storage beyond the running time of the JCAF Context Service. Once the program has terminated, all Entities and their Contexts are lost.

Sensor-based context ontology

Context type	Context value
Environment:Sound:Intensity	{Silent, moderate, loud}
Environment:Light:Intensity	{Dark, normal, bright}
Environment:Light:Type	{Natural, artificial}
Environment:Temperature	{Cold, normal, hot}
Environment:Humidity	{Dry, normal, humid}
User:Activity:PerMovement	{FreqOfWalking, FreqOfRunning, notAvailable}
Device:Activity:Placement	{AtHand, notAtHand}
Context type (higher level)	**Context value**
Environment:Location:Building	{Indoors, outdoors}