Definitions

- **Autonomic** [Horn, 2001, Kephart et al., 2003]
  Autonomous and able to pro-active adaptation to context changes.

- **Autonomicity**: [Kephart et al., 2003]
  Ability to be autonomous and to self-adapt to context changes.

- **Cloud computing** [Mell et al., 2011]
  Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics (On demand self service, Broad network access, Resource pooling, Rapid elasticity, Measured service), three service models (Software as a Service, Platform as a Service, Infrastructure as a Service), and four deployment models (Private cloud, Community cloud, Public cloud, Hybrid cloud).

- **Context contract:**
  An artefact expressing requirements into clauses on the interactions between a Mu-Broker and a context data producer or consumer; it is either a context consumer contract or a context producer contract.
Definitions (cont.)

- **Context data**
  Context data represent either useful raw data that have been directly acquired by a Context Manager through sensing, observing, or measuring some facts, or they represent data that have been processed, organized, structured or presented so as to make them meaningful and useful. Context data are equivalent to observations.

- **Context data distribution** [Bellavista et al., 2012]
  Context data distribution is the (distributed) middleware function that makes possible the injection of context data in the system and the automatic delivery of this data to all entities that have expressed any form of interest in them.

- **Context data type**
  The context data type corresponds to the type of context data. A context data type is equivalent to an observable.

- **Context manager** [Coutaz et al., 2005]
  A context manager is a software entity computing high-level information from various sources. Its functionalities includes:
• Context data acquisition,
• Context data processing (fusion, aggregation, interpretation, inference),
• Context data presentation to context-aware applications.

**Internet of Things** [Chen, 2012]
In the future, digital sensing, communication, and processing capabilities will be ubiquitously embedded into everyday objects, turning them into the Internet of Things (IoT, or machine-to-machine, M2M). In this new paradigm, smart devices will collect data, relay the information or context to each other, and process the information collaboratively using cloud computing and similar technologies. Finally, either humans will be prompted to take action, or the machines themselves will act automatically.
Middleware

[Krakowiak, 2009]
The software layer that lies between the operating system and the applications on each site of the system. Its role is to make application development easier, by providing common programming abstractions, by masking the heterogeneity and the distribution of the underlying hardware and operating systems, and by hiding low-level programming details.

[Issarny et al., 2007]
Middleware is a software layer that stands between the networked operating system and the application and provides well known reusable solutions to frequently encountered problems like heterogeneity, interoperability, security, dependability

Multiscalability:
Ability to cover several scales into at least one dimension such as device, network, geography, user, or administration
Definitions (cont.)

- **Observable** [Taconet et al., 2010]
  Abstraction which defines something to watch over (observe). (raw observable / high level observable) (e.g. Suzanne’s terminal battery level is an observable).

  An observable is equivalent to a context data type.

- **Observable Entity** [Taconet et al., 2010]
  Element representing a physical or logical phenomenon (person, concept, etc.) to which "observables" may be associated. (e.g. Suzanne’s terminal is an entity)

- **Observation** [Taconet et al., 2010]
  An observation represents the state of an observable at a given time.

  An observation is equivalent to a piece of context data.
Ontology [Maedche et al., 2001]
An ontology is a formal specifications of concepts in a domain of interest. It defines a common vocabulary to share knowledge. Formally, an ontology can be defined as the tuple: \( O := \{ C, R, Hc, rel, Ao \} \) where:
- \( C \) is a finite set of concepts
- \( R \) is a finite set of relations
- \( Hc \) is a direct relation \( Hc \subseteq C \times C \), which is called concept hierarchy or taxonomy. For example \( Hc(C_1, C_2) \) specifies that \( C_1 \) is a subconcept of \( C_2 \)
- \( rel \) is a function : \( R \rightarrow C \times C \). It relates concepts non-taxonomically. For example, \( rel(R) = (C_1, C_2) \) specifies that \( C_1 \) and \( C_2 \) have relation \( R \).
- \( A^\circ \) is a set of axioms. \( A^\circ \) is expressed in an appropriate logical language, e.g. first order logic.

Privacy:
Capacity of control about what, how, when, where and with whom to share information under what circumstances.

Quality-of-context ability:
Ability to qualify context data through metadata (QoC metadata).
Definitions (cont.)

- **QoC metadata:** Metadata that characterize context data and that can be used to determine the worth of the data.

- **Situation** [Bouzeghoub et al., 2007]
  A situation is a set of semantic relations between concepts (in one context dimension or between several context dimensions) which are valid and stable during an interval of time, where the term dimension here, means a context type like location, time, activity, etc.

- **Software architecture** [Bass et al., 1998]
  A software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

- **Software component** [Szyperski, 1998]
  A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to third-party composition.
**Viewpoint:** [ISO/IEC/IEEE, 2011]
A work product establishing the conventions for the construction, interpretation and use of views and associated architecture models to frame specific system concerns
References

Software architecture in practice.
Addison-Wesley.


A Situation-Based Delivery of Learning Resources in Pervasive Learning.

Challenges and opportunities of internet of things.
In Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific, pages 383–388.

Context is Key.
48(3):49–53.

Ibm autonomic computing manifesto.
Systems and software engineering — Architecture description.

A Perspective on the Future of Middleware-based Software Engineering.

The vision of autonomic computing.

Middleware Architecture with Patterns and Frameworks.

Ontology Learning for the Semantic Web.
IEEE Intelligent Systems, 16(2):72–79.

The NIST Definition of Cloud Computing.
NIST Special Publication 800-145, National Institute of Standards and Technology, Gaithersburg, USA.
*Component Software Beyond Object Oriented Programming.*

Taconet, C. et al. (2010).