Unit V

WEB OF THINGS

The future Internet will be driven by smart objects that communicate using open Web technologies."

– Dominique Guinard and Vlad Trifa

Unit Overview

This unit introduces the evolution from the Internet of Things (IoT) to the Web of Things (WoT), where smart devices are integrated using standard Web technologies such as RESTful APIs and Linked Resources. It explores the design of RESTful Smart Things, the concepts of the Real-Time Web of Things, and techniques for discovering, describing, and sharing smart devices. The unit further discusses the Semantic Web, Semantic Web Services, their processes and lifecycle, and the role of Ontology Engineering in enabling interoperability and intelligent decision-making in IoT systems. Finally, it examines ontology from organizational, IT system, and data perspectives to support semantic integration in IoT applications.

Objectives of the Unit

- To understand the evolution from the Internet of Things (IoT) to the Web of Things (WoT).
- To explain the principles of designing RESTful Smart Things using Web technologies.
- To study the modeling of smart device functionalities as Linked Resources.
- To understand the concepts of the Real-Time Web of Things and its future developments.
- To explore methods for discovering, describing, and sharing smart things.
- To understand the concepts and architecture of the Semantic Web.
- To study Semantic Web Services, including their processes and lifecycle.
- To understand Ontology and Ontology Engineering methodologies.
- To analyze the applications of Ontology Engineering in IoT.
- To examine ontology from organizational, IT system, and data perspectives.

Learning Outcomes

After completing this unit, students will be able to:

- Explain the transition from IoT to the Web of Things.
- Design RESTful Smart Things using Web standards and APIs.
- Model IoT device functionality as Linked Resources.
- Describe the architecture and future trends of the Web of Things.
- Explain methods for discovering and sharing smart devices on the Web.
- Discuss the principles and applications of the Semantic Web.
- Explain the lifecycle and processes of Semantic Web Services.
- Apply Ontology Engineering methodologies for IoT applications.

Importance of Studying this Unit

- Provides a strong foundation for integrating IoT devices with Web technologies.
- Helps understand RESTful architecture for scalable IoT applications.
- Enables interoperability among heterogeneous IoT devices using Web standards.
- Introduces Semantic Web technologies for intelligent information processing.
- Enhances knowledge of semantic interoperability using ontologies.
- Supports the development of smart, connected, and intelligent IoT systems.
- Facilitates efficient discovery, sharing, and management of smart devices.
- Improves decision-making through semantic knowledge representation.
- Prepares students for advanced research in Web of Things and Artificial Intelligence.
- Equips learners with skills required for developing next-generation smart applications.

Key Concepts

- Internet of Things (IoT)
- Web of Things (WoT)
- Evolution from IoT to WoT
- REST (Representational State Transfer)
- RESTful Smart Things
- RESTful APIs
- Web Technologies for IoT
- Linked Resources
- Resource-Oriented Architecture (ROA)
- Hypermedia
- Future of the Web of Things
- Real-Time Web of Things
- Smart Thing Discovery
- Smart Thing Description
- Smart Thing Sharing
- Device Interoperability
- Web Standards
- Semantic Web
- Semantic Web Architecture
- Resource Description Framework (RDF)
- RDF Schema (RDFS)
- Web Ontology Language (OWL)

From the Internet of Things to the Resource Oriented Architecture and Best Practices

5.1From the Internet of Things to the Web of Things

- As more and more devices are getting connected to the Internet, the next logical step is to use the World Wide Web and its associated technologies as a platform for smart things.
- Cool Town project, Kindberg et al. (2002) proposed to link physical objects with Web pages containing information and associated services. Using infrared interfaces or bar codes on objects, users could retrieve the URI of the associated page simply by interacting with the object.
- Another way to use the Web for real-world objects is to incorporate smart things into a standardized Web service architecture (using standards, such as SOAP, WSDL(Web Services Description language), UDDI(Universal Description, Discovery, and Integration)a way to publish and discover information)
- Instead of these heavyweight Web services ((SOAP/WSDL, etc)often referred to as WS-* technologies, recent "Web of Things" projects have explored simple embedded Hypertext Transfer Protocol (HTTP) servers and Web 2.0 technologies.

- In fact, recent embedded Web servers with advanced features can be implemented with only 8 KB of memory and no operating system support, and can therefore run on tiny embedded systems, such as smart cards.
- So far, projects and initiatives, subsumed here under the umbrella term “Internet of Things”, have focused mainly on establishing connectivity in a variety of challenging and constrained networking environments.
- A promising next step is to build scalable interaction models on top of this basic network connectivity and thus focus on the application layer.
- In the WOT concept, tiny Web servers are embedded into smart things and the REST architectural style is applied to resources in the physical world.

- The essence of REST is to focus on creating loosely coupled services on the Web, so that they can be easily reused.
- The services that smart things expose on the Web usually take the form of a structured XML document or a JavaScript Object Notation (JSON) object, which are directly machine-readable.
- These formats can be understood not only by machines, but are also reasonably accessible to people.
- With this smart things can not only communicate on the Web, but also provide a user-friendly representation of themselves.
- This makes it possible to interact with them via Web browsers and thus explore the world of smart things with its many relationships.

Web Of Things definition:

Web of Things is all about making devices accessible over the web using web protocols like HTTP, Web Socket, JSON etc. agnostic to any thing below the application layer, so that any device can be part of the universal WOT regardless of what protocols it uses to connect to internet.

IoT	WoT
Devices can be connected with any form of internet	WoT is made to handle and use the potential of IoT
It deals with actuators, sensors, computation, communication Interfaces.	It deals with web servers and Protocols.
Digitally Augmented objects make IoT	WoT is made up of the applications that are made for Io Devices.
Every IoT devices have a different Protocol	A single protocol is used for multiple/various IoT devices.
Programming is difficult because of multiple protocols	Programming is easy so it doesn't have multiple protocols.
All the protocols and standard are private and it cannot be accessed publicly	WoT can be accessed freely by anyone, anytime.

- Dynamically generated real-world data on smart objects can be displayed on such “representative” Web pages, and then processed with Web 2.0 tools.
- For example, things can be indexed like Web pages via their representations, users can “google” for them, and their URI can be emailed to friends or it can be bookmarked.
- The physical objects themselves can become active and publish blogs or inform each other using services, such as Twitter.

5.2 Designing RESTful Smart Things

- The “Web of Things” can be realized by applying principles of Web architecture, so that real-world objects and embedded devices can blend seamlessly into the Web.
- The main contribution of the “Web of Things” approach is to offer a foundation for the next step beyond basic network connectivity.
- Now, the use of REST as a universal interaction architecture, that interacts with smart things can be built around universally supported methods and a set of guidelines to Web-enable smart things and illustrate them with concrete examples of implemented prototypes.

5.2.1 Modeling Functionality as Linked Resources

- The central idea of REST revolves around the notion of a resource as any component of an application that is worth being uniquely identified and linked to.
- On the Web, the identification of resources relies on Uniform Resource Identifiers (URIs), and representations retrieved through resource interactions contain links to other resources, so that applications can follow links through an interconnected web of resources.
- Clients of RESTful services are supposed to follow these links, just like one browses Web pages, in order to find resources to interact with.
- In the case of the Sun SPOT, each node has a few sensors (light, temperature, accelerometer, etc.), actuators (digital outputs, LEDs, etc.), and a number of internal components (radio, battery). Each of these components is modeled as a resource and assigned a URI.
- For instance, typing a URI such as

http://.../sunspots/spot1/sensors/light

in a browser requests a representation of resource light of the resource sensors of spot1

- Resources are primarily structured hierarchically and each resource also provides links back to its parent and forward to its children.
- As an example, the resource

  http://.../sunspots/spot1/sensors/

  provides a list of links to all the sensors offered by spot1.
- This interlinking of resources that is established through both, resource links and hierarchical URI, is not strictly necessary, but well-designed URIs make it easier for developers to "understand" resource relationship and even allow non-link based "ad-hoc interactions".
- In a nutshell, the first step when Web-enabling a smart thing is to design its resource network, Identification of resources and their relationships are the two important aspects

5.5 Advanced Concepts: The Future Web of Things

- Even though there are many web standards and design principles that are leveraged for smart things, there are many open challenges remain.
- Three such challenges discussed are
  1. Needs for real-time data of many smart things applications.
  2. Finding and understanding services available in a global Web of Things.
  3. Mechanisms for sharing smart things.

5.5.1 Real-Time Web of Things

- HTTP is a stateless client/server protocol, This interaction model is well-suited for control-oriented applications where clients read/write data from/to embedded devices.
- However, this client initiated interaction models seem inappropriate for bi-directional event-based and streaming systems, where data must be sent asynchronously to the clients as soon as it is produced.
- Many scenarios must deal with real-time information to combine stored or streaming data from various sources to detect spatial or temporal patterns, as is the case in many environmental monitoring applications.
- As mentioned before, using syndication protocols, such as Atom, improves the model when monitoring, since devices can publish data asynchronously using AtomPub on an intermediate server or Smart Gateway.

- Nevertheless, clients still have to pull data from Atom servers. Web streaming media protocols have enabled transmission of potentially infinite data objects, such as Internet radio stations.
- Sensor streams are similar to streaming media in this respect. However, streaming media mainly support play and pause commands, which is insufficient for sensor streams where more elaborate control commands are needed.
- The **Extensible Messaging and Presence Protocol (XMPP)** is an open standard for real-time communication based on exchanges of XML messages, and powers a wide range of applications including instant messaging.
- Although widely used and successful, XMPP is a fairly complex standard, which is often too heavy for the limited resources of embedded devices used in sensor networks.
- An alternative type of Web applications that attempt to eliminate the limitations of the traditional HTTP polling has become increasingly popular.

- This model, called Comet (also called HTTP streaming or server push), enables a Web server to push data back to the browser without the client requesting it explicitly.
- Since browsers are not designed with server-sent events in mind, Web application developers have tried to work around several specification loopholes to implement Comet-like behavior, each with different benefits and drawbacks.
- One general idea is that a Web server does not terminate the TCP connection after response data has been served to a client, but leaves the connection open to send further events.
- The recent developments in Web techniques have allowed to build efficient and scalable publish/subscribe systems, It is suggested that a Web-based pub/sub model could be used to connect sensor networks with applications. PubSubHubbub (PuSH) is a simple, open pub/sub protocol as an extension to Atom and RSS.
- Parties (servers) speaking the PuSH protocol can get near-instant notifications (via callbacks) when a feed they are interested in is updated.

5.5.2 Finding and Describing Smart Things

- Another major challenge for a global Web of Things is searching and finding relevant devices among billions of smart things that will be connected to the Web.
- Finding them by browsing HTML pages with hyperlinks is literally impossible in this case, hence the idea of searching for smart things.
- Searching for things is significantly more complicated than searching for documents, as things are tightly bound to contextual information, such as location, are often moving from one context to another, and have no obvious easily indexable properties, such as human readable text in the case of documents.
- Beyond location, smart things need a mechanism to describe themselves and their services to be (automatically) discovered and used.
- But what is the best way to describe a thing on the Web so that both, humans and machines, can understand what services it provides?

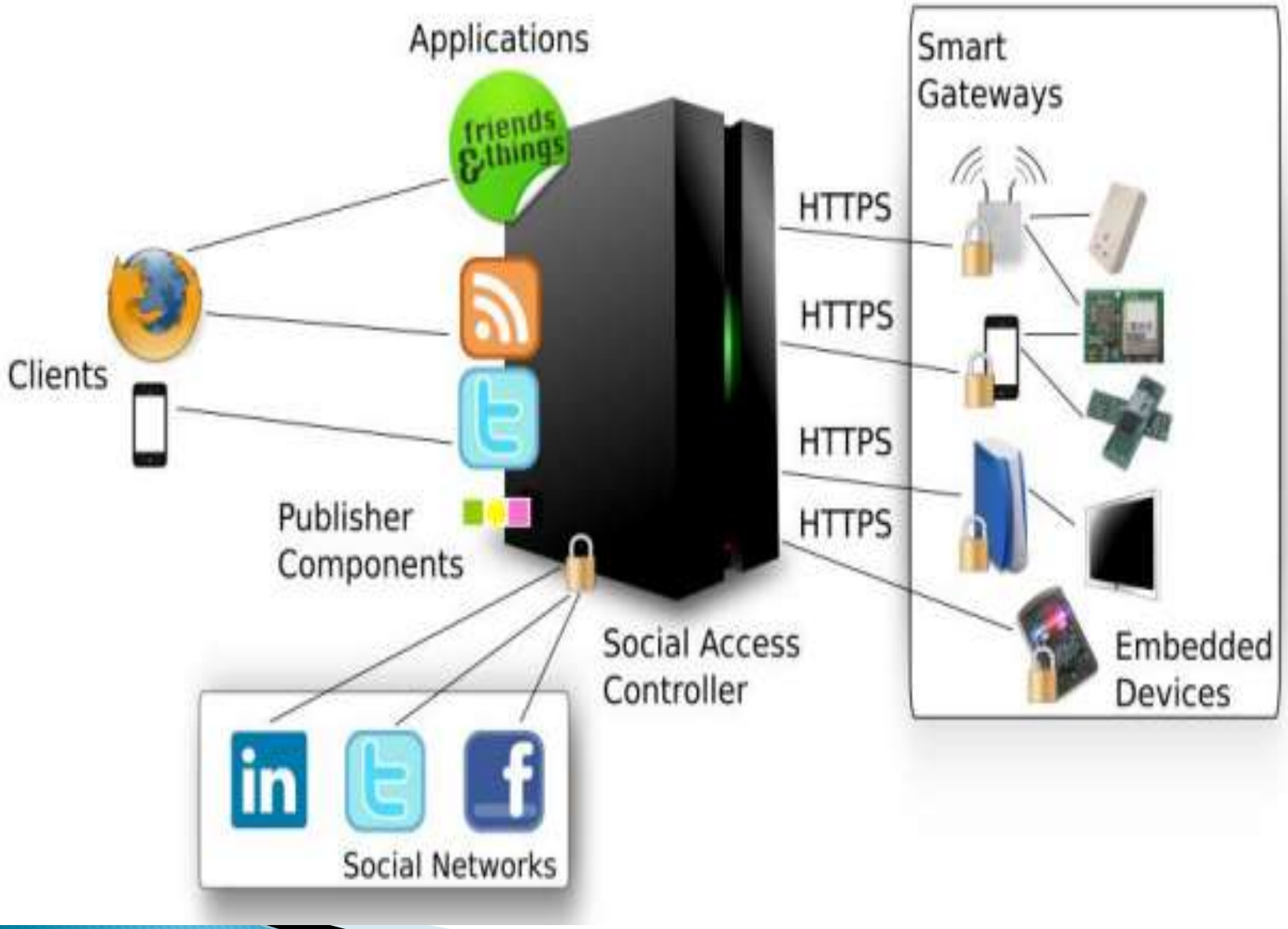
- To overcome the limited descriptive power of resources on the Web, several languages have been proposed, such as RDF(Resource Description Format) or Microformats, designed for both, human and machines.
- **Microformats** provide a simple way to add semantics to Web resources.
- There is not one single Microformat, but rather a number of them, each one for a particular domain; a “geo” and “adr” microformat for describing places or an “hProduct” and “hReview” microformat for describing products.
- Each Microformat undergoes a “standardization” process that ensures its content to be widely understood and used, if accepted.
- Microformats are especially interesting in a Web of Things for two reasons; first they are directly embedded into Web pages and thus can be used to semantically annotate the HTML representation of a thing’s RESTful API.

- Secondly, Microformats (as well as RDFa) are increasingly supported by search engines, such as Google and Yahoo, where it is used to enhance the search results.
- For example, the “Geo” microformat could be used to localize search results close to you or, in our context, to localize smart things in your direct vicinity.
- As an example, in Figure we use 5 microformats to describe a Sun SPOT and embed this semantic information directly in the HTML representation of the SPOT resources localize smart things in your direct vicinity.

Compound Microformats for Describing a Sun SPOT Using the Geo, hCard, hProduct and hReview Microformats

5.5.3 Sharing Smart Things

- Mashup developers often share their mashups on the Web and expose them through open APIs as well, making the service ecosystem grow with each application and mashup.
- The following Figure shows the simplified component architecture of a Social Access Controller (SAC), which serves as authentication proxy between clients and smart things.
- However, enabling such an open model for a Web of Things requires a sharing mechanism for physical things supporting access control to the RESTful services provided by devices.
- For example, one could share the energy consumption sensors in one's house with the community.
- However, this is a potentially risky process, given that these devices are part of our everyday life and their public sharing might result in serious privacy implications.


Applications
friends &things
Smart Gateways
HTTPS
HTTPS
HTTPS
HTTPS
Clients
Publisher Components
Social Access Controller
Embedded Devices
Social Networks

- A promising solution is to leverage existing social structures of social networks (e.g., Facebook, Linkedin, Twitter, etc.) and their (open) APIs to share things.
- Using social networks enables users to share things with people they know and trust (e.g., relatives, friends, colleagues, fellow researchers, etc.), without the need to recreate yet another social network or user database from scratch on a new online service.
- you can use various well-known social networks to inform your friends about the sensors you shared with them by automatically posting messages to their profile or newsfeed.
- The sharing process occurs in three phases.

1. First, the smart things owner accesses SAC by logging in, using at least one of his social networks credentials. SAC then uses delegated authentication with the social network to identify the owner.

2. Afterwards, the smart thing to be shared has to be crawled in order to identify the resources and capabilities of its RESTful services, i.e., which functionalities can be shared for that thing.

3. Finally, the user generates the access control list of the smart thing by selecting which friends can interact with what resource.

- In case of Twitter it simply tweets a message to the trusted connection (e.g., “Rachel shared her Ploggs Energy sensors with you”).
- The posted message also contains a link that redirects to the shared resource.
- The link does not point to the smart thing directly but to an instance of SAC that acts as the authentication proxy.

5.6 Discussing the Future Web of Things

- RESTful applications have rapidly become one of the most practical integration architectures, This makes it desirable to use Web standards for interacting with smart things.
- Although HTTP introduces a communication overhead and increases average response latency, it is still sufficient for many pervasive scenarios where longer delays do not affect user experience.
- Caching techniques can significantly improve the performance of concurrent sensor data reading by using tools used for massively scalable Web sites
- These techniques can be directly applied to Web devices, given that devices have on-board HTTP support.
- Web 2.0 mashups have significantly lowered the entry barrier for the development of Web applications, which is now accessible to non-programmers.

- Web standards allow any device to speak the same language as other services on the Web.
- This makes the integration of the real-world with any other Web content much easier, so that physical things can be bookmarked, browsed, searched for, and used just like any other Web resource.
- The developers mentioned a learning curve for REST but emphasized the fact that it was still rather simple and that once it was learnt, the same principles could be used to interact with a large number of services.
- They finally noted the direct integration to HTML and Web browsers as one of the most prevalent benefits.

SEMANTIC WEB

Definition

- The **Semantic web** is an evolving extension of the web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content

- The **Semantic Web** envisions an intelligent internet where the published content and data are adjoined with an additional set of machine-interpretable data - metadata and Linked Open Data - to provide semantics - meaning - as well as establish context - relationships between the data and content. These two types of data (**semantic metadata and Linked Open Data**) form the basis of the Semantic Web.

5.7 Semantic Web Services

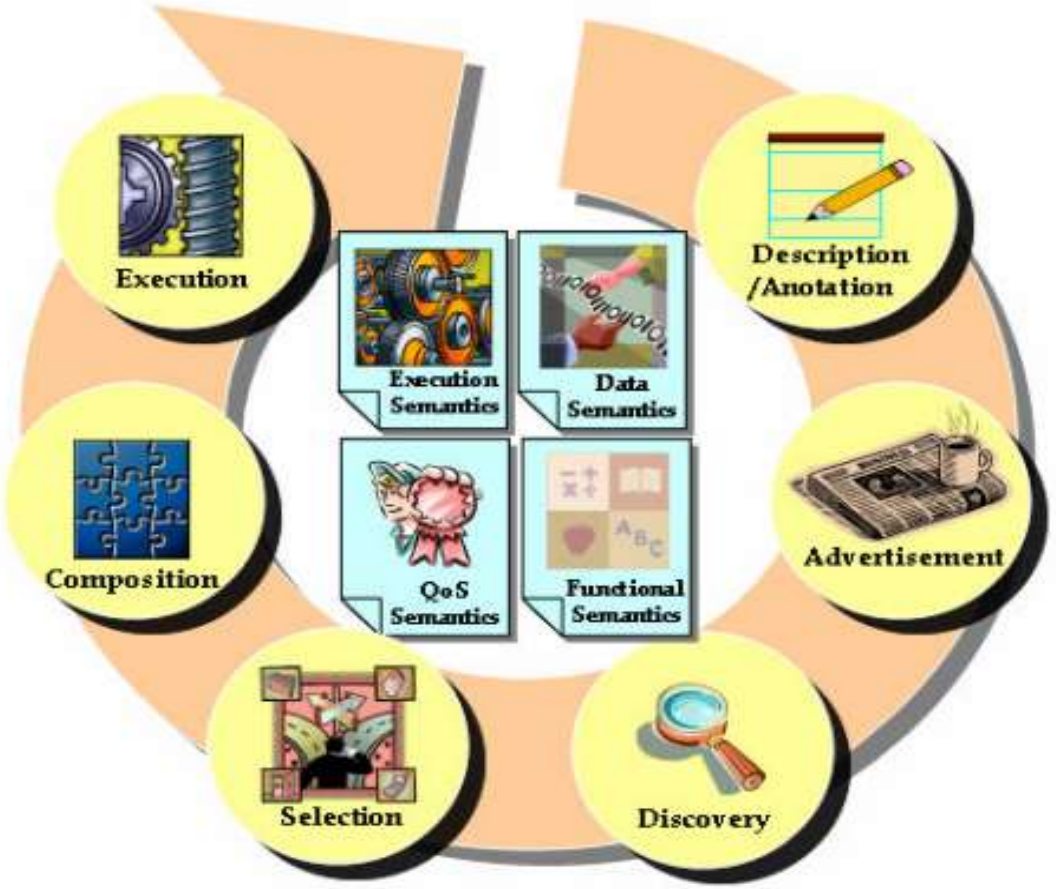
- The Semantic Web Services vision wants to combine the ideas behind the Semantic Web and the already available technologies for Web Services.
- This will enable automatic and dynamic interaction between software systems and, therefore, enable the implementation of the Internet of Things and Services.
- Current Web Services technology provides only support for a standard interface description, without machine-interpretable information about the software system's functionality.
- This issue is addressed by the concepts of Semantic Web Services by adding semantic information to Web Services.
- The goal is to describe them in machine-interpretable form, providing information on what the software does for a potential user and how it is done.

- This semantic information allows more sophisticated possibilities for discovering services than it is currently possible with UDDI.
- The combination of these technologies will support the development of more sophisticated applications, as services can be advertised and discovered more automatically and at high speed.

"**Semantic Web Services allow the semi-automatic and automatic annotation, advertisement, discovery, selection, composition, and execution of inter-organization business logic, making the Internet become a global common platform where organizations and individuals communicate among each other to carry out various commercial activities and provide value-added services." (Cardoso and Sheth 2005)**

5.8 Semantic Web Services Processes and Lifecycle

- Adding semantics to the web services can play an important role, as shown in the web process lifecycle



- According to Cardoso and Sheth (2005), the semantics for the Web Services can be differentiated into:

1. Functional semantics are describing the aims of the services and the operations, including information about the required inputs and outputs.

2. Data Semantics are descriptions of the input and output data format for discovery issues and a common understanding in communication, in which data format the data has to be sent and how the receiving data can be semantically interpreted.

3. Quality of Service (QoS) Semantics are necessary for the selection of the most suitable service. Services can have different quality aspects, e.g. cheapest offer, fastest delivery, etc., and this provides possibilities to locate and select the service which matches best a required quality criterion in terms of QoS.

4. Execution Semantics are describing the flow of message exchange (message sequences) and provide a conversation pattern for the web service execution, flow of actions, preconditions, and effects on web service invocation, etc.

For the success of the web process, these stages are all significant and should follow the next described steps

1. Semantic Web Service Annotation:

Description of the Web Services with the different required types of semantics for Web Services

2. Semantic Web Service Advertisement

Publication of the service's capabilities on syntactical and operational level, using semantics in a repository (different technologies are available, centralised, e.g. UDDI, and even peer-to-peer)

3. Semantic Web Service Discovery

Process of discovering appropriate services before selection, based on syntactic information, data, functional and QoS semantics

4. Semantic Web Services Selection

Selection of the most suitable service matching defined QoS metrics and resulting in the best quality criteria match

5. Semantic Process Composition

Web Services are highly autonomous and heterogeneous; therefore, composition is an important stage. Semantics enhance interoperability of Web Services, and in the web process composition (automatic, semi-automatic or manually)

6. Execution of Web Process

Execution of the services and the whole process based on the execution semantics, the sequential or parallel invocations of services and operations, specified by the flow of messages and data based on the defined semantics

For the implementation of Semantic Web Services a complete framework is necessary, consisting of three components:

1. **A conceptual model** (ontology)
2. **A formal language** to provide formal syntax and semantics for the conceptual model
3. **An execution environment** combining all components and carrying out the tasks for eventually service and process automation.

- As the idea of Semantic Web Services is relatively new, there currently exist different approaches for the realization of Semantic Web Services, Example
  - Web Service Modeling Ontology (WSMO)
  - Web Ontology Language for Web Services (OWL-S)
  - or Semantic Web Services Framework (SWSF)

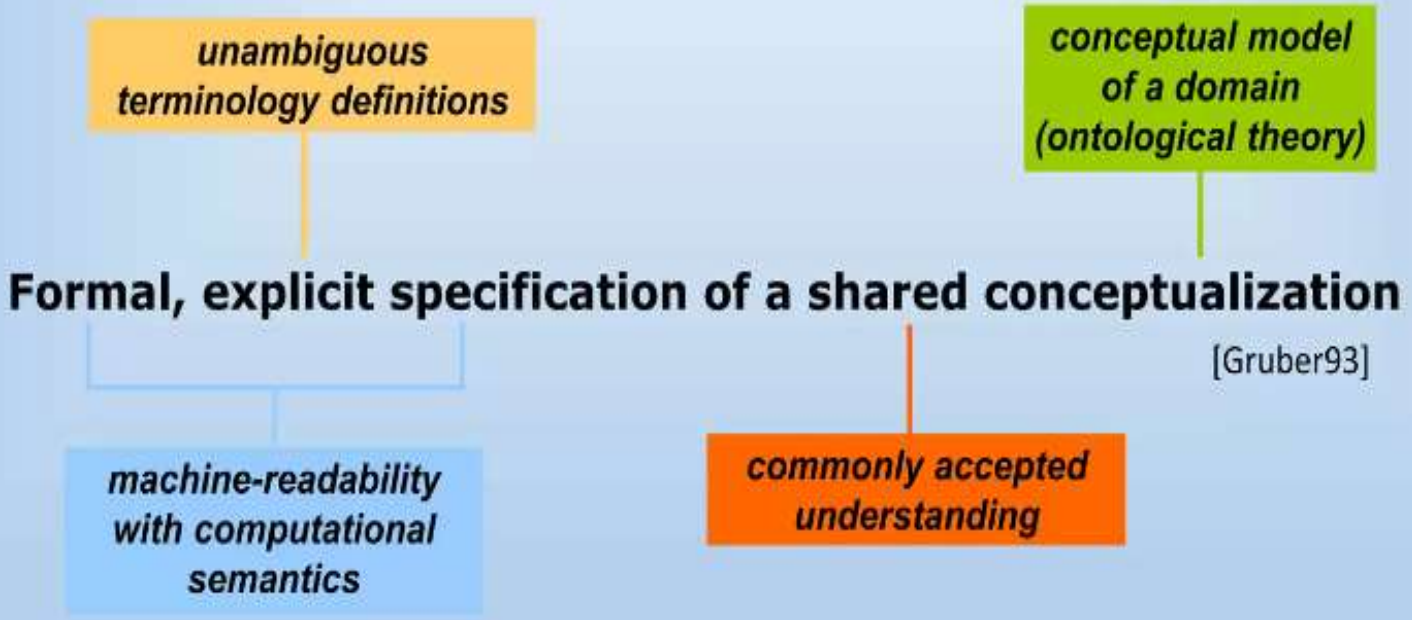
- The most promising approach for the realization of a system of Semantic Web Services seems to be the WSMO approach, as it consists of an extensible ontology modeling language, a Web Services Modeling Language and an execution environment.
- On the Operational Level, Semantic Web Services and Semantic Web Processes are the enabling technologies, which could fulfill the requirements of the Internet of Things and Services.

ONTOLOGY

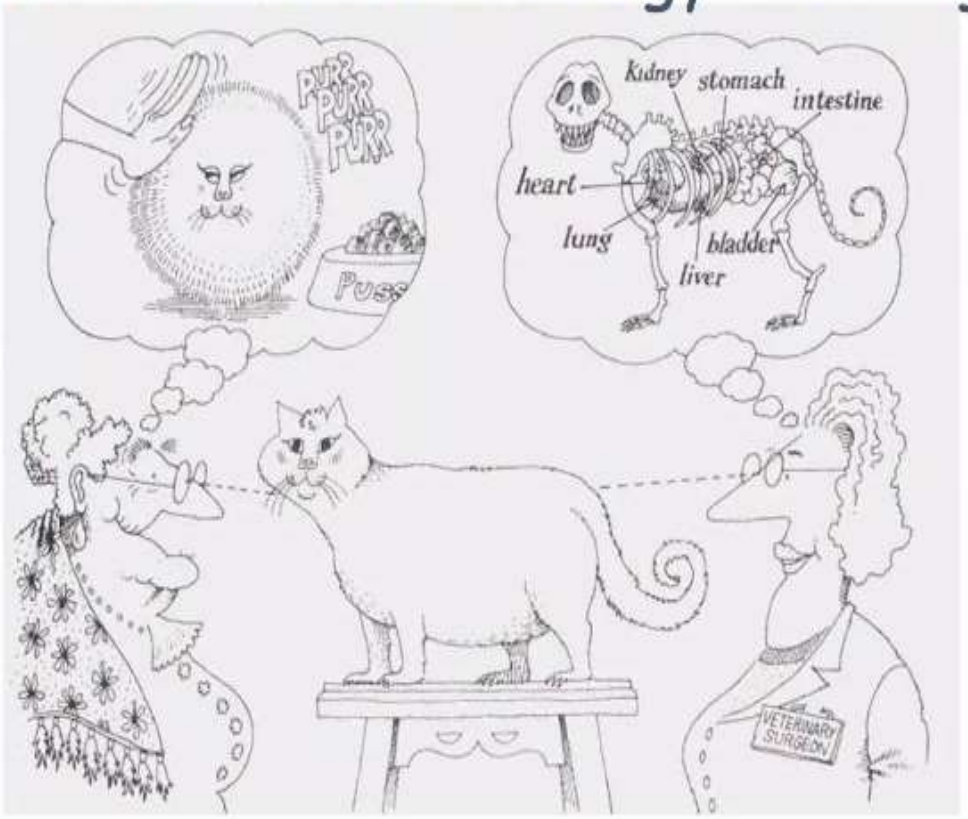
The Origin of Ontology

- study or concern about what **kinds** of things **exist**
- what entities there are in the universe.
- the ontology derives from the Greek *onto* (being) and *logia* (written or spoken). It is a branch of metaphysics , the study of first principles or the root of things.
- The term is borrowed from **philosophy**, (Not very useful definition for our purpose!!)
  - *What characterizes being?*
  - *Eventually, what is being?*

The Definitions of Ontology



World without ontology = Ambiguity Example (2)



Ambiguity for humans

Cat

The Vet and Grand beby associate different view for the concept cat.

5.9 Ontology

- Ontology, as a discipline, is the branch of philosophy that is concerned with the nature of things that exist in the universe.
- More specifically, it is the science that aims to provide an exhaustive and definitive classification of things based on their similarities and differences.
- By **exhaustive,** we mean that it provides an explanation for everything that is ongoing in the universe.
- By **definitive,** we mean that every type of things should be included in the classification.
- In this sense, the ontology discipline tries to provide answers to the question: What are the features common to all things?

- In **computing science**, an ontology is commonly defined as: “a formal, explicit specification of a shared conceptualization”.
- More specifically, an ontology is an engineering artefact composed of

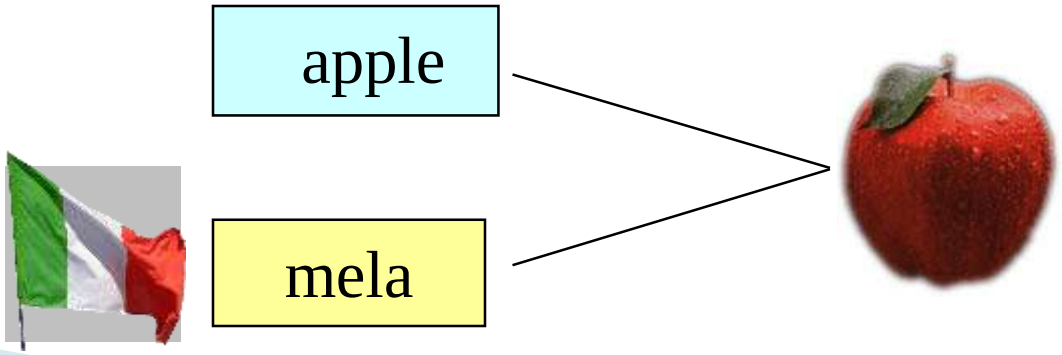
  (i) a vocabulary specific to a domain of discourse, and

  (ii) a set of explicit assumptions regarding the intended meaning of the terms in the vocabulary for that domain.
- This set of assumptions is generally expressed in terms of unary and binary predicates, by which concepts and the relations between them are expressed.

What is a conceptualisation

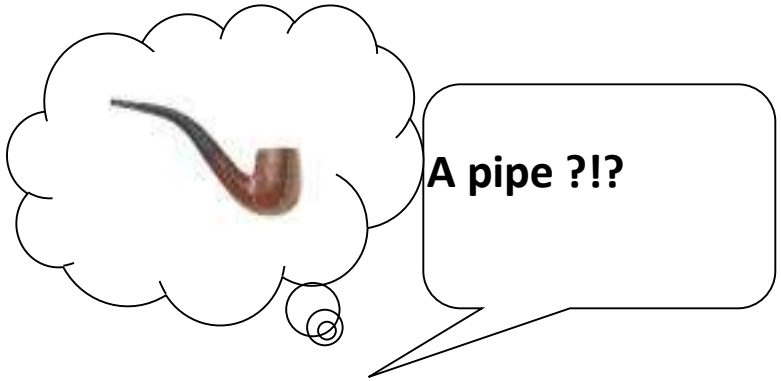
- Conceptualisation: the formal structure of reality as perceived and organized by an agent, independently of:
  - the **vocabulary** used (i.e., the ***language*** used)
  - the actual occurrence of a specific ***situation***

- Different situations involving the same objects, described by different vocabularies, may share the same conceptualisation.



Ontological commitment

- Agreement on the meaning of the vocabulary used to share knowledge.



Why using the ontology(2)

- **Inability to use the abundant information resources on the web**

  The WEB has tremendous collection of useful information however getting information from the web is difficult.

  Search engines are restricted to simple keyword based techniques. Interpretation of information contained in web documents is left to the human user.

- **Difficulty in Information Integration**

  The integration of data from various sources is a challenging task because of synonyms and homonyms.

- **Problem in Knowledge Management**

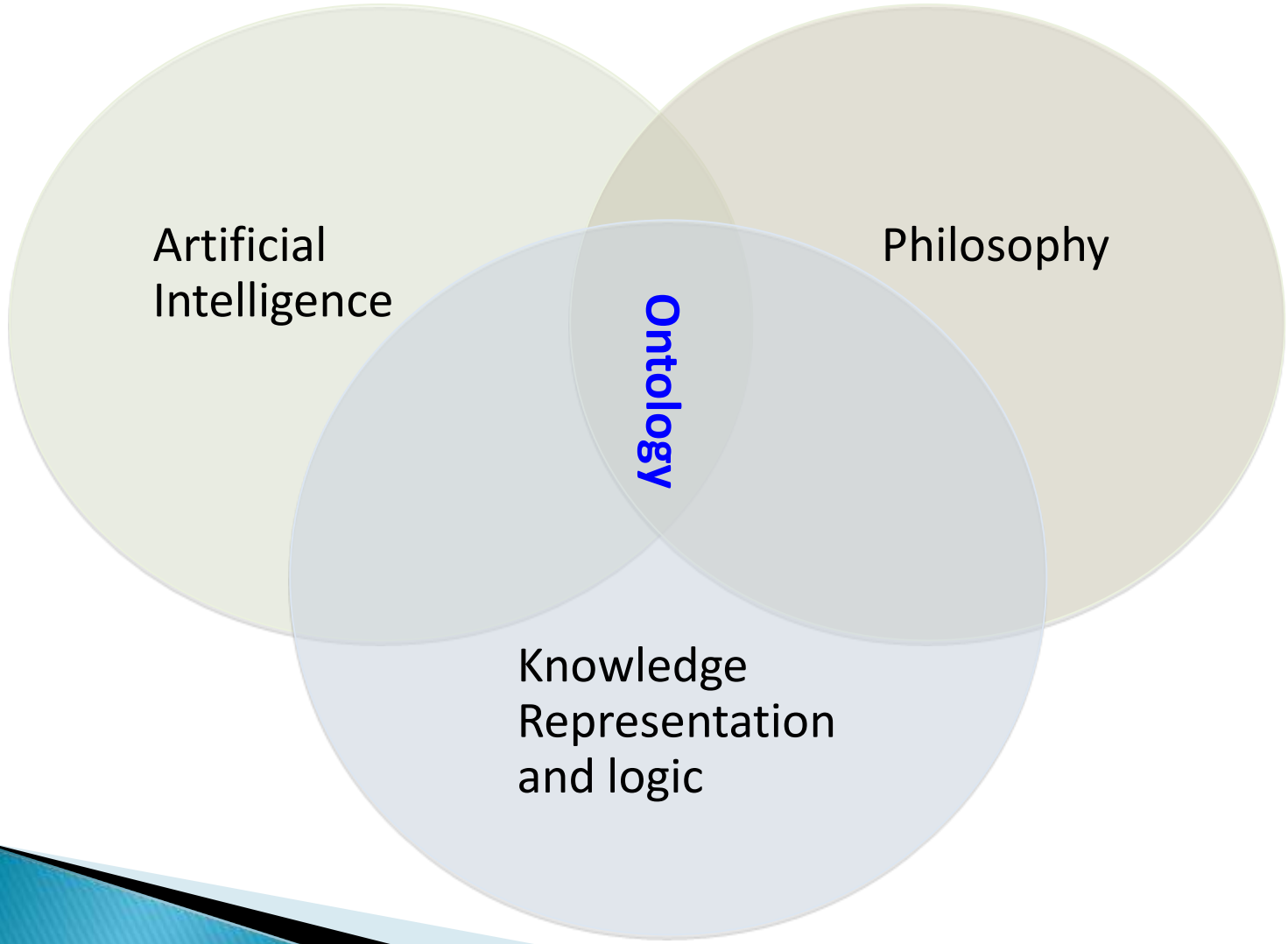
  Multi-actor scenario involved in distributed information production and management.

  "People as well as machines can't share knowledge if they do not speak a common language

  [T. Davenport]

Ontologies provide the required **conceptualizations** and **knowledge representation** to meet these challenges.

- Historical context



Different languages have been developed over the last two decades to represent ontologies.

- Simple HTML Ontology Extension (SHOE) was developed to annotate web pages with semantics.
- Ontology Exchange Language (XOL) was primarily developed to exchange ontologies in the bioinformatics domain.
- Resource Description Framework (RDF) allows users to describe the relation between different web resources,
- Web Ontology Language (OWL) extends on the RDF vocabulary to provide precise meaning through formal semantics.

5.10. Ontology Engineering Methodologies

What Is "Ontology Engineering"?

Defining terms in the domain and relations among them

- Defining concepts in the domain (classes)
- Arranging the concepts in a hierarchy (subclass-superclass hierarchy)
- Defining which attributes and properties (slots) classes can have and constraints on their values
- Defining individuals and filling in slot values

What Is "Knowledge Engineering"?

- **Knowledge engineering** is the application of logic and ontology to the task of building computable models of some domain for some purpose.

- Over the last two decades, several ontology engineering methodologies have been developed.

1. **Gruninger and Fox (1995)** propose a method inspired by knowledge based development using first order logic.

2. **Uschold and King (1995)** propose a method for building ontologies based on their experience with the Enterprise Ontology for system interoperation.

3. **Fernandez et al. (1997) builds Methontology** on the main activities of software development and knowledge engineering methods, proposing an ontology development life cycle based on evolving prototypes.

4. **CommonKADS (Schreiber et al. 1999)** is a methodology for knowledge engineering in general, which is used to design and analyze knowledge-intensive, structured systems.

5. Alternatively, **the Developing Ontology Grounded Methods and Applications framework (DOGMA)** is a formal ontology engineering framework inspired by various scientific disciplines, such as database semantics and natural language processing.

- Although DOGMA partly draws on the best practice of the other methodologies, it differs from these approaches by providing a strict separation between the lexical representation of concepts and their relationships and the semantic constraints.
- The Meaning Evolution Support System (DOGMA-MESS) is STAR Lab's methodology and tool to support community-driven ontology engineering.

- The following Figure illustrates how a domain ontology is created through the interaction among three different types of stakeholders, namely the domain expert, the core domain expert and the knowledge engineer.
- The domain expert is a professional within the domain of discourse,
- while the core domain expert has a deep understanding of the domain across different organizations.
- The knowledge engineer, who has excellent expertise in representing and analyzing formal semantics, is responsible to assist the domain experts and core domain experts in the processes of ontology creation, validation and evolution.

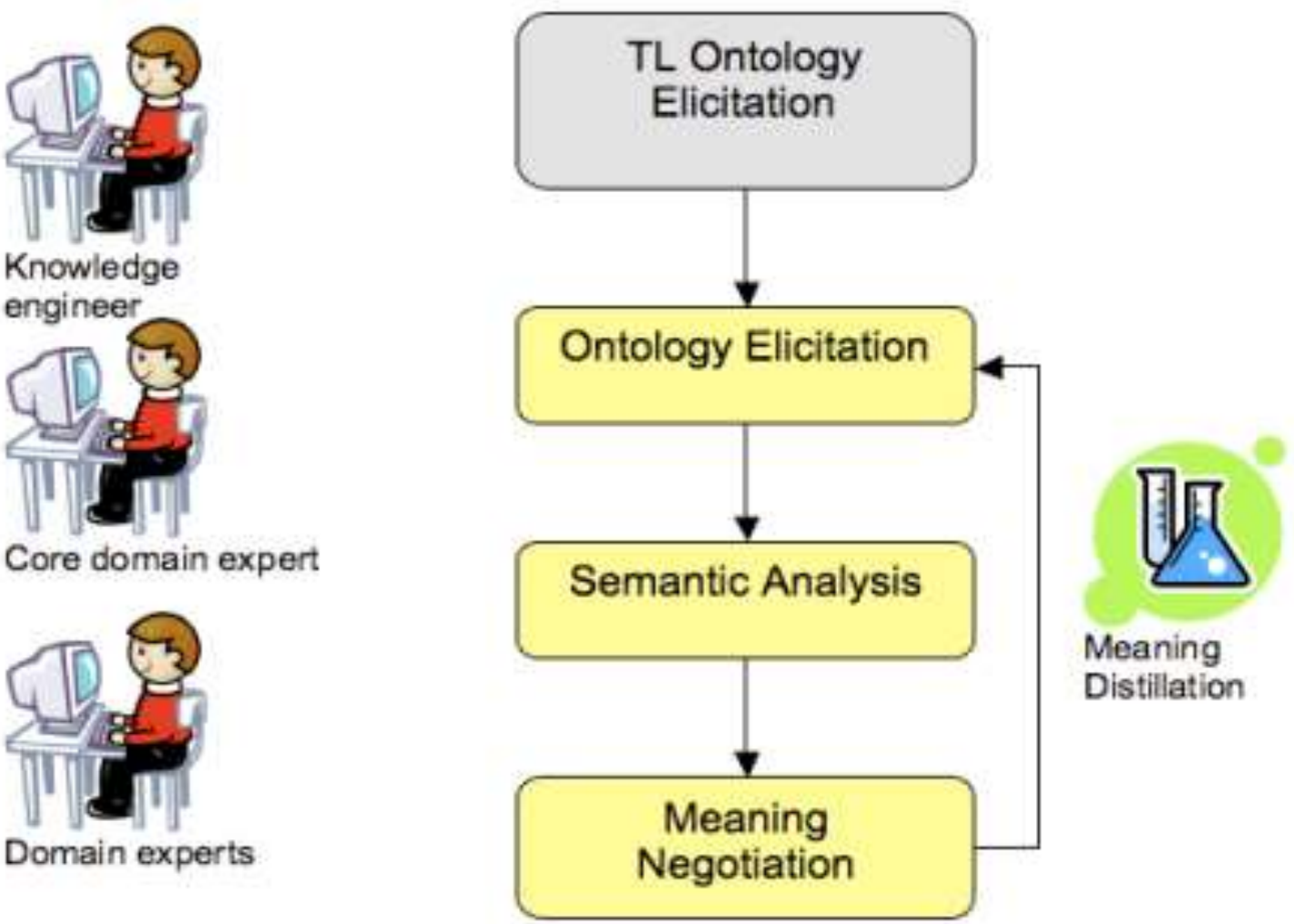


Fig. DOGMA-MESS Iterative Process

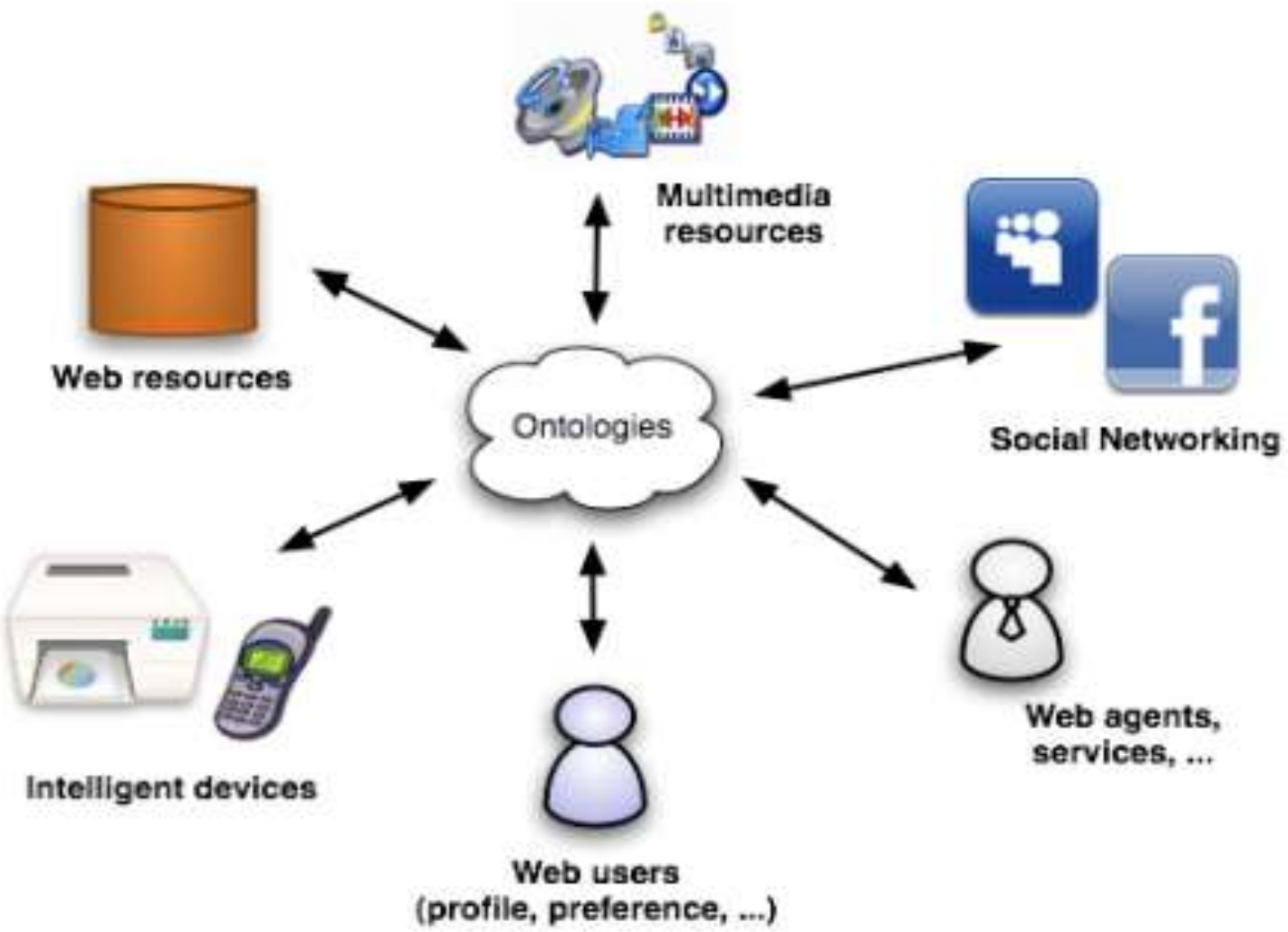
5.11. Application of Ontology Engineering in the Internet of Things

- Three ontology-based services have been described, that would enable three different areas of interoperability as needed for DiY (Do-it-Yourself) application creation in the Internet of Things.

  1. Knowledge Integration and Sharing
  2. Ontology-based Search
  3. Context-aware Computing

Knowledge Integration and Sharing

- As the Web has changed from a mere repository of documents to a highly distributed platform where new types of resources can be discovered and even easily shared.
- One can extrapolate the Web as an Internet of Things making everyday objects addressable via IPv6 as well as an Internet of Services making services easy to implement, consume, and trade.
- Making knowledge transparent to users and services thus requires the development of a formal and precise vocabulary that (i) defines concepts shared by a community and (ii) can be processed by machines.
- The following figure shows how a semantic layer can be used to facilitate knowledge integration and sharing on the Web.



Ontology-based Knowledge Integration and Sharing

- Similarly, Eid et al. (2007) have developed an ontology to discover sensor data like GPS or temperature, consisting of three components:

1. the **Suggested Upper Merged Ontology (SUMO)** is an upper level ontology developed by the IEEE to promote interoperability, information search and retrieval, automatic inference, and extendibility;
2. the **Sensor Hierarchy Ontology (SHO)** includes models for data acquisition units and data processing and transmitting units; whereas
3. the **Sensor Data Ontology (SDO)** describes the context of a sensor with regard to spatial and/or temporal observations.

- Alternatively, the Web Service Modeling Ontology (WSMO), provides a conceptualisation for the core elements of Web Services.

5.11.1 Ontology-based Search

- The **DiYSE (Do-it-Yourself Smart Experiences)** project aims to develop a framework to enable communities to create and exchange applications (i.e. software components) for ubiquitous computing and ambient intelligence, leveraging the Internet of Things.
- In practice, these applications are likely to come from a number of repositories and in a variety of formats.
- However, different repositories are likely to use different terminologies for the indexing.
- This is where ontology-based search comes in as a solution, as in that approach the indexes are expressed in terms of an ontology which translates and hides the different repositories that have committed to it.

- In DiYSE, the ontology-based search needs to serve two types of users. Technical users are likely to use technical terms to define the capabilities of a hardware or software component, whereas non-technical users will use other non-technical terms that are meaningful to them.
- As a result, semantic ‘translation’ methodologies are researched that resolve the differences between technical and nontechnical terminologies in order to get to a true DiY ecosystem on top of the Internet of Things.

5.11.2. Context-aware Computing

- Therefore, context-aware computing (Schilit et al. 1994) focuses on gathering information about users, like status, location, preferences and profile, next to environment factors such as lighting conditions, noise level, network connectivity, nearby things and even social aspects.
- This information is then used to adjust the behaviour of an application to suit user needs and preferences.
- As is done in the DiYSE framework, context management can be supported semantically by two core elements, namely the **contextual ontology** and **the context model.**
- The contextual ontology provides a conceptualisation of the characteristics of real world objects, while the context model provides access to the contextual knowledge.
- So, based on for example this ontology, agents and users are able to interoperate to provide context-aware services in the dynamically changing smart environments.

5.12. Ontology and the Organizational Perspective

- An ontological representation of a real-world domain requires a conceptualization of the reality.
- A conceptualization of the real world domain involves two steps,
  1. First, the selection of relevant parts (e.g. concepts, properties) and the relations between them.
  2. Second, The representation is accomplished by defining the vocabulary for the model which means to name the elements (concepts, classes, properties, relations…) of the model.
- When revising the problem from the **organizational perspective**, we can see that the technical implementation, or the used languages, encodings or data formats are not as crucial to this point.

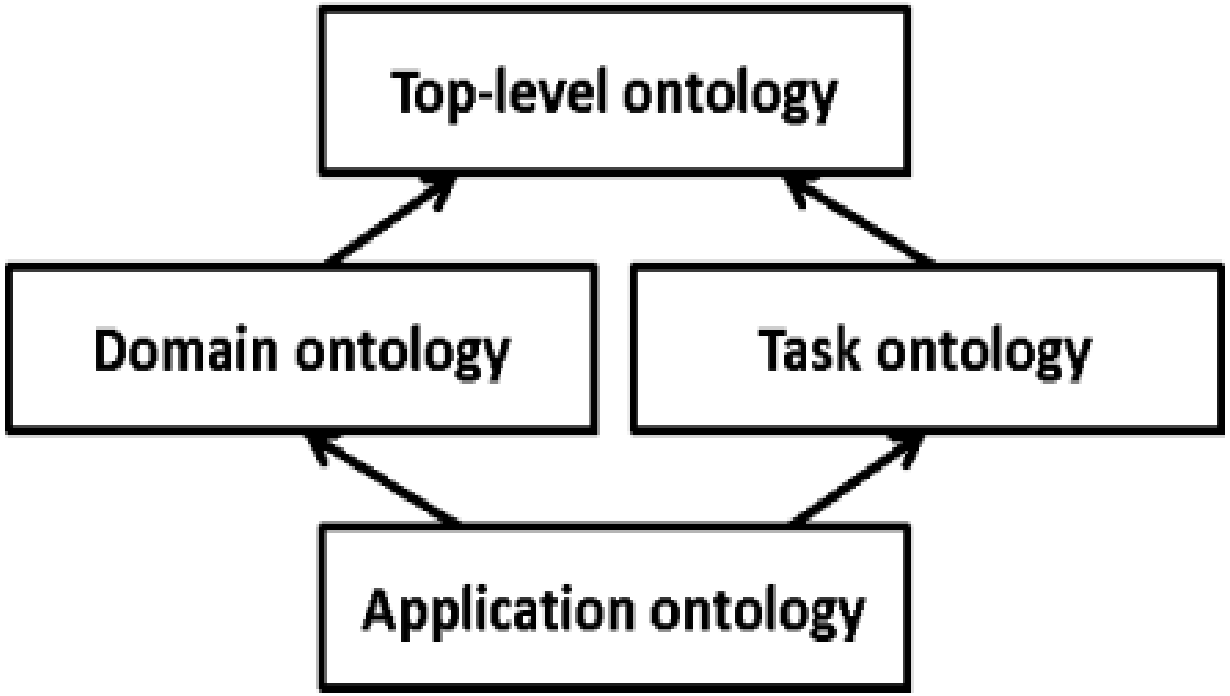
- Applications that label themselves as compliant to the Internet of Things and Services must be aware of this organizational problem, first.
- There are an infinite number of possible conceptualizations of a real world domain.
- Two organizations can conceptualize the same real-world domain in a different way or conceptualize two different real-world domains to the same model.
- Both scenarios would lead to serious interoperability problems that are very often hard to detect.
- Interoperable systems, or Internet of Things applications that must obtain openness must use the concepts of ontology to state their understanding of the real world domain.
- This allows other applications or organizations to adjust their own vision in order to correctly interoperate.

5.12.2. Ontology and the IT-System Perspective

- The IT-system perspective requires a more technical approach to guarantee the ability for cooperation to enable an Internet of Things and Service.
- Standard technologies like, for example, the SOA are able to provide open and accessible interfaces for applications.
- Ontology concepts already found their way into the world of SOA.
- Semantic Web Services are assisted by ontologies to fully semantically describe and understand the interfaces they offer and use.
- Computer programs can automatically discover and choose appropriate services, interfaces and data-structures to consume particular web resources.

5.12.3. Ontology and the Data Perspective

- When talking about interoperability with respect to the Data Perspective in the meaning of the ability to cooperate and to exchange data, one must consider the interoperability of the domain models used by them.
- Basically, there are the possibilities that the same representation of some data has a different intentional meaning or that there is a different representation of the same intentional meaning that cannot be translated offhand.
- Anyway, different representations of the same information can be an interoperability issue.
- The following figure shows how ontologies can be used to model an open and transparent data model.


Top-level ontology
Domain ontology
Task ontology
Application ontology

The data models are designed as follows:

1. A **Top-level ontology** represents very common and general concepts like space, time, object, matter, event, action, etc., which are independent of a particular application domain.

2. The **Domain ontology** and the **Task ontology** specialize the terms defined in the top-level ontology for a certain problem domain (e.g. logistics). These ontologies differ between the concepts inside the domain (e.g. cargo, vehicle, route…) and the tasks (load, unload, convey goods…) performed there.

3. The **Application ontology** describes concepts that depend on both, the domain concepts of a domain and the tasks that are performed. Typically, it is a specialisation of both ontologies. Usually, this ontology contains domain entities playing a certain role while performing a certain task (e.g. a cargo item which arrives in time at its destination).

Text Books

- *Internet of Things, A hands-ON approach, Arshdeep Bahga, Vijay Madisetti.*
- *Architecting the Internet of Things, Dieter Uckelmann,Mark Harrison, Florian Michahelles*

Referene Books

- *Networks, Crowds, and Markets: Reasoning About a Highly Connected World, 2010, David Easley and Jon Kleinberg, Cambridge University Press.*
- *The Internet of Things: Applications to the Smart Grid and Building Automation, 2012, Olivier Hersent, Omar Elloumi and David Boswarthick – Wiley.*
- *The Internet of Things – Key applications and Protocols, 2012, Olivier Hersent, David Boswarthick,*
- *Omar Elloumi, Wiley.*

2 Marks Questions:

1. State the main difference between the Internet of Things (IoT) and the Web of Things (WoT).
2. Name the four standard HTTP verbs used to design a RESTful Smart Thing and their physical actions.
3. What is meant by "Modeling Functionality as Linked Resources" in a smart device?
4. Why is a "Real-time Web of Things" necessary for smart systems?
5. How does the W3C Thing Description (TD) help machines find and understand smart objects?
6. Which open standard is used for securely "Sharing Smart Things" with third-party software?
7. Summarize the overall objective discussed regarding the "Future Web of Things".
8. Define the term "Semantic Web" in a physical smart environment.
9. What role do Semantic Web Services (SWS) play in automated IoT networks?
10. List the four sequential stages in the Semantic Web Services process and lifecycle.
11. Define an "Ontology" and explain its application in the Internet of Things.
12. How does the 'Grüninger & Fox' methodology use Competency Questions (CQs) in Ontology Engineering?
13. Explain the focus of Ontology from the "Organizational Perspective".
14. Describe the focus of Ontology from the "I-T System Perspective".
15. What is the primary concern of Ontology from the "Data Perspective"?

5 Marks Questions:

Q1.Explain the differences between the Internet of Things and the Web of Things and how devices connect using standard web tools.

Q2. Describe how to design RESTful Smart Things using standard web commands like GET and POST to control sensors and devices.

Q3. Explain how to model smart device features as linked resources so computers can discover capabilities automatically through web links.

Q4. Discuss why a Real-Time Web of Things is necessary and how WebSockets provide instant updates compared to standard web polling.

Q5. Explain how W3C Thing Descriptions and JSON-LD help computers automatically find, identify, and understand new smart objects.

Q6. Describe how the OAuth 2.0 framework allows users to securely share smart things and delegate limited device access to third-party applications.

Q7. Explain how the Semantic Web adds clear meaning to raw sensor data so different computer systems can understand it.

Q8. Detail the four main stages of the Semantic Web Services lifecycle which are discovery, selection, composition, and execution.

Q9. Compare the Methontology step-by-step lifecycle framework with the Grüninger and Fox methodology that uses competency questions.

Q10. Discuss the application of ontology engineering in IoT and how it helps devices from different manufacturers share data without errors.