Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 36 (2014) 210 – 217

Procedia Computer Science

Complex Adaptive Systems, Publication 4
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2014- Philadelphia, PA

# Approach to manage Complexity in Internet of Things

Angel Hernandez-Bravo[a], Jesús Carretero[b]

*b UC3M. Universidad Carlos III de Madrid, Av Universidad 30, 28911, Madrid, Spain*
*a IBM , Santa Hortensia 26, Madrid 28002, Spain*

**Abstract**

Complexity research is a main concern in disparate science fields such as computing, city planning, Internet, etc. In computing and programming, complexity became early an issue and a target for software engineering, but in fact the pattern concept was born in architecture and city planning environment. The universe is not made of "things," but of patterns of complex and interactive geometries. In this paper, we generalize such a concept to Internet of Things and Smart Cities domain and consider more complex abstract structures called semi lattices. Furthermore, an approach to model them is proposed.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
*Keywords*: Complex Systems; Smart Cities; Internet of Things; IoT; Systems of Systems; OWL; CL; Common Logic; Tree; Semi-lattice

## 1. Smart Cities modeling as a paradigm of Internet of Things (IoT) modeling

Ashton [1] was probably the first person to coin the phrase-term 'Internet of Things' (IoT) back in 1999 to describe this vision of Internet-connected sensors, devices, and citizens. Linked to the concept of Internet of Things is the concept domain of Smart Cities, where Internet and other relevant technologies continued to develop and mature a number of solutions from computer industry giants that today made IoT a feasible option for a good number of modern cities, as Boulos and Shorbaji describes [2]. Indeed, IoT is often perceived as a major enabler for the 'smart cities' of the present and the future. Modeling large scale IoT systems, as a Smart City could be, is a challenge due to the huge number of heterogeneous components and the even greater complexity of their relationships if a cooperative or distributed approach is used instead a centralized one. As Vermesan stated, this could be faced using simulation of System of Systems [3]. Nevertheless, we additionally propose another approach for modeling IoT and Smart Cities using the Semantic Web as a way of knowledge representation [4].

OWL2 (Ontology Web Language 2) is the Web Ontology Language recommended by the World Wide Web Consortium (W3C, http://www.w3c.org). The Semantic Web is an extension of the current web created to better enabling computers to work in cooperation. The official World Wide Web Consortium (W3C) document on OWL states that it was designed “for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates much better machine interpretability of internet content than that


Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2014.09.081

supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary, stronger syntax and formal semantics." [5]. Some commercial platforms to manage smart cities, as IBM IOC (Intelligent Operations Center) and many others platform providers, supports the use of OWL.

Many smart cities or IoT systems can be modeled using simple hierarchy trees where increasing the number of "things" does not increase complexity, because in fact it is a simple tree. Increasing complexity, either on Smart City or on a more generalized IoT, comes due to the introduction of additional concepts as, for example, space and time. Consider the following scenario:

*An operator instructs his intelligent **trash_truck** to go to the next trash container in priority list, , **trash_container_i,** according to some optimization parameters and correlating data from GPS geo-position, filled sensor, etc. He should take only **the recycling_plastic_container** and emptying it into **trash_truck**. This container is **next** to a street with a **traffic jam**. **Afterwards** he needs go to the **trash managing plant**.*

The concepts suitable for an ontology are in bold and additional relevant concepts, as space and time, are underlined. To model or represent this second scenario is obvious that a hierarchy OWL model has problems. In Fig. 1 a graph of both scenarios is shown.

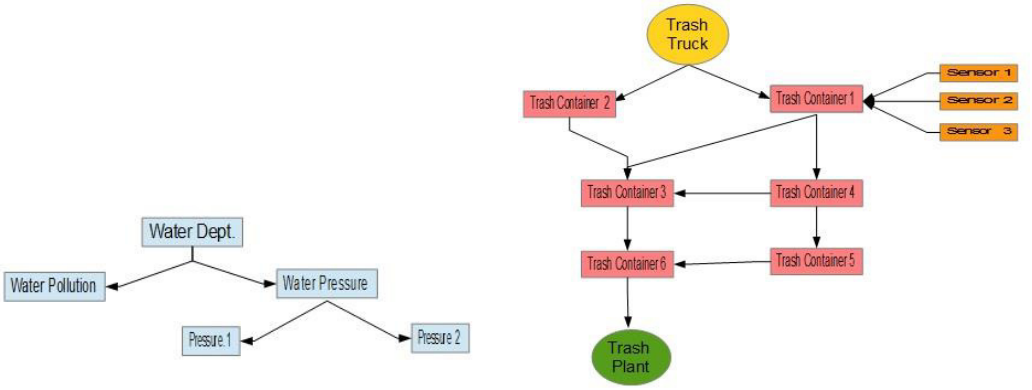


Fig. 1. (Left) IoT simple hierrarchy tree ; (Right) IoT complex relationships

We can see the that issues involved in modeling Smart Cities are the same than those in the more generalized domain of Internet of Things .

### *1.1. Learning from cities*

Complexity research is a main concern in disparate science fields such as computing, city planning, Internet, etc. In computing and programming, complexity became early an issue and a target for software engineering as Booch described [6]. Probably the most impacting concept used to try solving complexity in this field is the concept of pattern applied to the software design in object oriented programming by the Gang of Four [7]. But in fact the pattern concept was invented as a useful concept by Christopher Alexander, a mathematic and architect who considers that the universe is not made of "things," but of patterns of complex and interactive geometries [8]. In the theoretical work previous to Alexander's, the city was usually modeled as an abstract hierarchical tree structure. Alexander considered that a tree is not able to represent a city and proposed a more complex abstract structure called a semi-lattice, which must be used instead of the tree structure [9].

In addition, to correlate both abstract structures, trees and semi-lattices, to the nature of the city, there is a distinction that should be taken into account: on the one side, the cities which have arisen more or less spontaneously over many years, the natural cities, and, on the other side, the cities which have been created by planners, let's say, the artificial cities. A detected issue is the existence of some essential ingredient missing in artificial cities being present in natural cities and difficult to be modeled. This is related with human behavior and

human relationships.

Both the tree and the semi lattice structures are ways of modeling a large collection of many small systems linked to make up a large and complex system, or as it is called now, a system of system (SoS). Raising the abstraction level, both structures are structures of sets. A set is a collection of elements, which are thought as belonging together, for some reasons or criteria. From the point of view of a city design, we should be restricted to consider sets which are collections of material elements or assets such as blades of grass, cars, houses, gardens, water pipes, service vehicles, etc. When those elements of a set belong together because they cooperate working together somehow, then, that set of elements become is a system. They could be the water system, the traffic light system or many other that are even more interrelated as a person waiting in the traffic light while read information in a device deployed in the street. The common feature is that every system has elements that work together.

The physically unchanging part of a system is of special interest. A traffic light, a device in the street, a pipe of water, etc., being related among them, form the fixed part of the system and constitutes the receptacle in which the changing parts of the system – persons, information etc. - can work together.

Using Alexander´s terminology [8] this fixed part is a "unit of the city" (UOC), a concept coming from the forces which hold the elements together and from the dynamic coherence of the larger living system including it as a fixed invariant part. Those forces are familiar to everybody who works with software design patterns and are pretty similar to the ones in a city design (or in the context of this paper, in a IoT design). There are many fixed subsets of the city that can be considered as receptacles for its systems and can be thought as significant "physical units" (PU) , but a few of them can be choosen to provide a relevant picture of the city . Such a collection of subsets has a definite structure that allows the modeling of a city as a system of systems (SoS).

### *1.2. A generalized class of complexity problems*

The first hypothesis presented in this work is that the Alexander ideas to structure a city can be generalized to other domains such as internet, internet of things (IoT) , or related systems of systems as city operations, traffic operations, etc. Following the Alexander approach [9] we can divide the relationships in two ways: tree structures and no-tree structures. Alexander stated in his work, when the structure fulfill certain conditions it is called a semilattice and when fulfill other more restrictive ones, it is called a tree.

Semilattice axiom states that: A collection of sets forms a semilattice if and only if, when two overlapping sets belong to the collection, the set of elements common to both also belongs to the collection Mapping all those symbols against city components we can see (Fig. 2b) a set composed by water pipes, maintenance workers and pressure gauges and, in other hand, another set composed by traffic lights, maintenance workers and cabling. The two units overlap in the maintenance workers, which is a recognizable unit satisfying the semilattice axiom

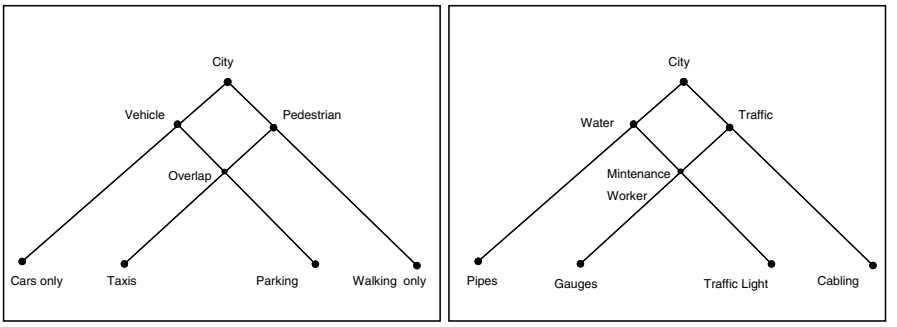


Fig. 2. (a) Overlapping of cars and pedestrian domains; (b) Overlapping of water and traffic domains

On the other side the tree axiom says: A collection of sets forms a tree if and only if, for any two sets that belong to the collection either one is wholly contained in the other, or else they are fully disjoint. . This is the description of a tree, that in fact can be seen as a simpler semilattice. We must consider the difference between a tree and more general semilattices which are not a tree because they contain overlapping units. The important thing is not only the overlap among elements, but the fact that the semilattice is a much more complex structure than a tree.

This much bigger variety is an index of the great structural complexity that a semilattice can have and, in this way, be useful for modeling when compared with the structural simplicity of the tree model. More generally, the lack of structural complexity of the tree model limits the understanding of complex systems of systems (SoS). City plans use to have tree structures, as for example the Greater London plan (1943), done by Abercrombie and Forshaw [10]. 'The parts of the city consist of several sub-units, generally with their own shops and schools, corresponding to the neighborhood units. The city is conceived as a tree with two principal levels. The communities are the larger units of that structure; the smaller sub-units are the neighborhoods. There are no overlapping units. In that kind of trees each unit is the fixed, unchanging residue of some system in the living city (i.e. a freeway is the residue of movement and commercial exchange). However, in cities there are more systems at work whose physical residue does not appear as a unit in these tree structures and the living real systems, whose existence actually makes the city live, that is to say, they are not properly modeled with the tree structure.

### *1.3. Modelling IoT complex problems*

A tree structure is easy to model and is the usual approach in semantic web as stated N. Shadbolt, Bernes-Lee et al. [12]. This is the way followed in most web semantic issues related with Internet of Things as reviewed by Barnaghi et al. [13]. Smart Cities, in a similar way than IoT in general, are starting to make use of same approaches. The main idea is going along these 4 steps: (i) the raw sensory data, (ii)evolve to structured data (with semantics) , (iii) take abstractions and perceptions, (iv)get actionable intelligence. A tree hierarchy modeling is useful for managing the two first steps but not the others.

In general terms a model is simply a representation of an environment or system. Often we think of a model as something that is graphic in nature, as with software modeling using something like UML, or an architect with the design of a building. But in this case often we are going the opposite direction and taking physical systems and distilling them down to a few simple key words and phrases of text. The goal is to identify some component, often spoken of as a piece of physical infrastructure, but it could be a person, place, or thing, and then uniquely identify that item within the context of everything else in the system. For example, taking the following real case used on a commercial middleware platform for city management named IBM Intelligent Water , you can see (Figure 3) that a physical meter located on some property can be defined within a simple XML structure. In this case we are building an instantiated model that represents real world assets and infrastructure.

```
<tns:WaterUsageMeter rdf:ID="WaterUsageMeter_1">
 <cim:RSM_IdentifiedObject.name>Water Usage Meter 1</cim:RSM_IdentifiedObject.name>
 <cim:RSM_UnnamedObject.description>Water Usage Meter 1</cim:RSM_UnnamedObject.description>
</tns:WaterUsageMeter>
```

Fig. 3. XML hierarchical description of sensors on IoT.

We can extend this model and attach locations and measurement values to each of the assets, building a full representation of the environment we are modeling. Obviously it will be noted that a semantic model is much more then just XML (eXtended Markup Language), however it is easier to understand if we think of it this way now. We can ask ourselves why not choosing a Relational Data Model. One of the key differences between a relational data model and a semantic model is adding meaning ( or better to say, semantics) to the data. The semantic model follows a predefined ontology that structures data and their relationships in a meaningful way. This can provide a similar context across all the data in the model while maintaining uniqueness of character for individual assets and data sources. For a simple set of infrastructure or entities this may not be a problem. and a programmer with

database skills can design a relational model that holds all the relevant information from a small system that is being managed. However expanding that system to include other systems, complex systems, and different protocols, will make that relational model quickly become unwieldy.

Semantic Models has been used on commercial platforms, especially when trying to understand them in the context of a Smart Cities or industry solutions engagements where lots of other things are going on, as described by J. Bernal [14]. The best way to focus on understanding the infrastructure is been trying to model and breaking it down into simple terms that focus on the following: Asset: Name, Location, Type, Measurement and Measurement-Value. This can be sophisticated using deepest semantic web concepts and semantic technologies based on machine-interpretable representation formalisms that have shown promise for describing objects, sharing and integrating information, such as OWL, for semantic, and RDF(S), for taxonomy and data model. RDF (Resource Definition Framework) is a simple language for expressing data models, which refer to objects (or web resources ) and their relationships. RDFSchema extends RDF with a vocabulary for describing properties and classes of RDF-based resources, with semantics for generalized hierarchies of such properties and classes. And OWL (Ontology Web language) adds more vocabulary for describing properties and classes: among others, relationships between classes (as dis-jointness), cardinality, equality, richer typing of properties, featuress of properties (as symmetry), and enumerated classes. Those tools are good for modeling at the semantic level, but are not the right tool for the last two steps of the four above mentioned.

## 2. Boosting up semantics

A conclusion right now: tree view based tools are the state of art on modeling IoT and Smart Cities, and this is a limitation on the complexity that can be managed and a limitation to the capability to take abstractions and perceptions (knowledge) and finally, get actionable intelligence (wisdom) on Internet of Things systems. Coming back to Figure 1 (b), we see there a typical ambient assisted living (AAL) scenario:

*Operator instructs his intelligent **trash_truck** to go to the next trash container in priority list according optimization parameters, **trash_container_i**, correlating data form GPS geo-position, filled sensor, etc. He should take only **the recycling_plastic_container** and empty it into **trash_truck**. This container is next to a street with a **traffic jam**. **Afterwards** he needs go to the **trash managing plant**.*

Current ontologies for ambient assisted living as OpenAAL OWL Ontology cover the core of the concepts (in **bold**) providing classes or generic superclasses for them [15] . But it does not cover the overall scenario, and some relevant concepts (as related with time and space, in underline ) are not covered with the required level of complexity. More precisely, OpenAAL establish that appointments have a fixed timeframe and containers in the streets are connected among them, but not say what exactly that means.

Nevertheless, foundational ontologies based on first-order logic are good candidates to cover space, time and other IoT concepts at the level of complexity that could be required, for example, in a central command and control system. That is the case where is needed to express knowledge of the arrangement of objects, streets, areas, etc. . A first-order formalization using region connection calculus (RCC) serves for qualitative spatial representation and reasoning. RCC abstractly describes regions, either in Euclidean space, or in a topological space, by the possible relations to each other. An example with two areas $a_1$ and $a_2$ can be seen in Figure 4. Two areas in a city, or two sensors in an area, are either separated, or bordering, or overlapping or be the same. Each area can be accessed thru an entry

$$\forall a_1, a_2.\text{equal}(a_1, a_2) \veebar \text{overlapping}(a_1, a_2) \veebar \text{bordering}(a_1, a_2) \veebar \text{disconnected}(a_1, a_2) \veebar \\ \text{proper\_part\_of}(a_1, a_2) \veebar \text{proper\_part\_of}(a_2, a_1)$$

Fig. 4. A RCC style description of city areas or area sensors arrangements.

Using the Heterogeneous Tool Set (HETS) from Universität Bremen [16] for logic translation we can have an heterogeneous specification of that environment allowing for reusing the OpenAAL OWL ontology, but at the same time formalizing a first-order logic (FOL) spatial calculus. To be more precise, the compact representation of mutual dis-jointness chosen in that case makes use of sequence markers from Common Logic (CL). The Common Logic (CL) module extends a previously imported OWL ontology so that it has access to all entities of the OWL ontology by name. We can specify that two areas in a city, or two sensors in an area, are connected (in terms of the OpenAAL language) if certain conditions in terms of HETS' Common Logic module, or certain conditions in terms of OpenAAL are met as can be seen in Figure 6. We can establish a link between an OWL ontology and a Common Logic ontology by reusing elements of the signature of the OWL ontology (in that case OpenAAL's is-in-area predicate) in the Common Logic (CL) ontology. We have included implicitly the use of Common Logic (CL) but can be demostrated.

Another interesting mapping between OWL and a first order logic as CL is the case of a view between the OWL Time ontology and a reimplementation in Common Logic, using the "OWL22CommonLogic" translation in HETS. OWL has the capability to model time concepts but we will see that OWL modeling generates important limitations on capturing real no-tree structures. Hobbs and Pan stated [17] the capability of OWL-Time in the treatment of temporal concepts that can reach a high complexity. OWL-Time is able to model Topological Temporal Relations with two subclasses of TemporalEntity: Instant and Interval. Intervals are things with extent and instants are point-like, that is to say, they have no interior points. The predicates "begins" and "ends" are relations between instants and temporal entities. It is possible to view those OWL-Time views on a different logic as Common Logic (CL) using HETS again using the "OWL22CommonLogic" translation. As we have just seen , complexity concepts as time and space , can be modeled with limitations by OWL, a tree structured language , but we begin to perceive the first order logic language fit better.

OWL is good on the field of ontologies or taxonomies, where in most cases the only available relation is a ????, and a few other predefined relations. Such limitations difficult seriously the ability of an ontology to satisfactorily represent a domain of knowledge. Despite OWL (and RDF) have the advantage of being decidable, but they lack sufficient expressivity to reflect the world. Consequently, much of the semantics of many concept domains cannot be represented in an ontology. Formulating an ontology using a more expressive language such as FOL is more difficult, but is capable to model reality much better. Tree based languages as OWL miss some properties beyond transitivity. Semi-lattices, a kind of Partial Ordered Set (POSET) , have additional propterties as Transitivity, Reflexivity, Anti-Symmetry, etc. Depending of the details of such mathematical properties we could have different kinds of semi-lattices that fit better depending the situations to model.

## 3. Ontology Languages

Nowadays, there is a dichotomy between ontologies based on a more straightforward implementation of First Order Logic FOL (as, CL, CLIF, etc) - which generally differentiate between "relations", "functions" and "objects" and other ontology languages (as OWL, RDFS) that instead use terms such as "classes", "instances", "attributes" and "relations." This discrepancy between first order logic and description logic leads to two loosely dissimilar understandings of what ontologies allow.

A widely used family of formal languages are description logics, as OWL, that represent formalisms that are less expressive than traditional propositional logic and predicate calculus, but work pretty well in categorizing and classifying "is_a" or sub. Nowadays, there is a dichotomy between ontologies based on a more straightforward implementation of First Order Logic FOL or similar ones as CL, CLIF, etc, which generally differentiate between "relations", "functions" and "objects", and other ontology languages (as OWL, RDFS) that instead use terms such as "classes", "instances", "attributes" and "relations."

Most of Description Logics (DL) have Boolean operators and some form of quantification. Description logics were developed to close the gap of the absence of formalism in many semantic networks. They are close to propositional and modal logics and they only employ pieces of first order logic (FOL), being used quite a lot on optimization of

tableau algorithms, where decidability is the main concern. Those ontologies, based on description logics, generally capture the notion of “is-a” or “contained in” relations, although the idea of “is-a” is not clear. Ontologies derived from this approach are "taxonomy biased", where they categorize objects in hierarchies (trees). Lacking the greater expressivity afforded by other logical formalisms, relations are simply cooked to naming something, missing important parts of reality and the behavior of objects. Their logical vocabulary is restricted "is-a", "contained-in" and some FOL pieces as transitive property as we saw in Figure 8 example, and for this reason the emerging model is too ambiguous in such a way that concepts being represented have a big amount of information that remains external to its ontological expression.

### *3.1. Common Logic (CL)*

In order to go deeper on FOL (First Order Logic) methods dealing with a semilattice approach, let us consider a language that, being similar to traditional FOL (TFOL), have some differences that makes it more usable as described on the standard ISO 24707 documentation [18]: *"Common Logic has a syntax which is signature-free and permits 'higher-order' constructions such as quantification over classes or relations while preserving a first order model theory, and a semantics which allows theories to describe intensional entities such as classes or properties"* .

It could be considered as a bridge between limited (but currently implemented) descriptive logics and much more powerful logics as FOL, potentially more useful on highly complex systems of systems (SoS) as IoT. Additionally, CL appears to have the higher number of good features over other ontology languages as Gruber states [19]. CL has several dialects, being CLIF (Common Logic Interchange Format) one of the most extended. The feature of all CL dialects is a greater expressivity that traditional FOL (TFOL) and, of course, descriptive logics, so any structure articulated using TFOL or XML/XDS, RDFS or OWL can be translated into CL, or specifically CLIF, although the reverse can not be said.

In addition to grammar and syntax of CL we need a conceptual framework to translate them to meaningful logical representations of reality. Other ingredient to establish a formal language is the need for a series of inference rules, which allows reasoning. These rules are functions which map sets of formulae to sets of conclusions but are not covered by CLIF or CL and are a future target in complexity management.

The traditional problem with FOL is that it is far more complex than more familiar languages as OWL or RDF. Nevertheless there are now some graphical tools providing acceptable usability such as EZPAL Protege [20][21]. That tool is used to manage ontologies but we have used it for modeling IoT and Smart Cities generating interesting underlying knowledge models. On complex ontologies can be detected categories of patterns that have been well catalogued in computer science. EZPAL is able to provide a intuitive graphical view of pattern representations using models, making easier communication of FOL knowledge capturing and development of suitable reasoning axioms. Also, users may draw the behavior of the concepts and then translate it to the most appropriate class of axioms, providing incorporation of FOL axioms into the ontologies.

Following the pattern approach of using ontologies and FOL, further development goes in the way of reusing patterns. A way to implement that idea are the Upper Level Ontologies (ULO), being ULO an ontology that serves as a basic starting point for more specialized ontologies [22]. Till now, only taxonomic biased of “is-a” and “contained-in,” representing domains in a hierarchical manner, are available; but the idea is useful because it is possible to develop logical building blocks allowing to model systems of systems with growing complexity.

## 4. From Axioms to Rules Model

EZPAL is a pretty easy interface of PAL (Protege Axiom Language), which is able to support ontology axioms and events that, it turn, can be transformed to information processing rules and then in executable rules on a run-time engine. PAL is a superset of first order logic capable of writing strong logical constraints on special purpose frames [23] . Those constraints are implemented as instances of a :PAL_CONSTRAINT class .The EZPAL plugin was used to acquire PAL constraints without knowing the language, by using a set of templates based on reusable patterns of

encoded axioms designs [20]. We have applied this tool for IoT applications: Let's suppose a SLA (Service Level Agreement) to be evaluated on the overlapped node of "maintenance worker". The SLA can be (not being exhaustive): i) a definition of the number of pipes, gauges, traffic lights or cables (city_object) , being maintained in a unit of time; or, ii) the rate of traffic lights or water gauges out of order. Because the overlapped character of this node of the semilattice (is not a tree) is difficult to model it with descriptive logic tools as OWL or RDFS. So we used the first order logic, PAL in this case, to model using an axiom.

Well this can be seen as a simple rule, but the difference is that the "maintenance worker" SLAs depend on two domains (is part of semilattice, not a node in a tree), and so, evaluation is far more complex. Here we consider only two domains, but reality can be far more complex with lot of lattice overlapping. After that, next step is to get a rules model. There are several options but SQL triggers are a straightforward method. Some authors [23] propose a mapping of general PAL constraints to SQL triggers as we have developed using a MySQL database.

## 5. Conclusion

Limitations are shown on the usual tree or hierarchical approaches to model complexity on System of Systems such as Internet of Things, Smarter Cities and similar ones. An alternative approach, based on semi-lattice structures and First Order Logic reasoning languages and tools, is proposed to solve those limitations. .,

## References

1. K. Ashton, That “Internet of Things” thing, RFID Journal No 6. 4986 (2009)
2. M.N.K . Boulos and N.M. Al-Shorbaji, On the Internet of Things, smart cities and the WHO Healthy Cities, International Journal of Health Geographics 2014, 13:10
3. O. Vermesan et al. , Internet of Things, 2.5, River Publishers Series in Communications, 2011.
4. T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web, Scientific American, May 2001.
5. D. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview." W3C Recommendation February 10, 2004.
[Online] Available: http://www.w3.org/TR/owl-features/
6. G. Booch., Measuring Architectural Complexity, Software, IEEE , vol.25, no.4, pp.14,15, . 2008
7. E. Gamma et al. , The Gang of Four , Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994
8. C. Alexander, A Pattern Language: Towns, Buildings, Construction. Oxford University Press, USA. p. 1216, 1977
9. C. Alexander, A city is not a tree, Architectural Forum, Vol 122, No 1,, pp 58-62 (Part I), Vol 122, No 2, pp 58-62 (Part II), 1965
10. P.Abercrombie and J.H. Forshaw, County of London Plan, Macmillan and Company Ltd, 1943.
11. E.Banfield, Political Influenece, Transaction Publishers, 1961
12. N. Shadbolt et al.. , The Semantic Web Revisited, IEEE Intelligent Systems, 1541-1672 April 13, 2007.
13. P. Barnaghi et al., Semantics for the Internet of Things: Early Progress and Back to the Future, Int. Journal on Semantic Web and Information Systems, Vol 8, issue 1, 2012
14. J Bernal, Semantic Models for Smart Cities, IBM Developer Works, Sept 23, 2012
15. http://openaal.org
16. T. Mossakowski et al. , The Heterogeneous Tool Set , Tools and Algorithms for the Construction and Analysis of Systems.
Lecture Notes in Computer Science Volume 4424, pp 519-522, Springer, 2007
17. Hobbs, J. R. and Pan, F. 2004. An Ontology of Time for the Semantic Web. ACMTransactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing, Vol. 3, No. 1, pp. 66-85.
18. https://www.iso.org/obp/ui/#iso:std:iso-iec:24707:ed-1:v1:en
19 T.R. Gruber, A translation approach to portable ontologies. Knowledge Acquisition, vol. 5, iss. 2, pages199-220, 1993.
20. CJ. Hou,et al. , EZPAL: environment for composing constraint axioms by instantiating templates. International Journal of Human-Computer Studies, vol., iss. 5, pp. 578 – 596, May 2005.
21. http://protege.stanford.edu
22. H. Herre et al., General Formal Ontology (GFO): A Foundational Ontology Integrating Objects and Processes, Onto-Med Report 8, Institute of Medical Informatics, University of Leipzig, 2006
23. G.Magyor et al., Advances in Information Systems Development: New Methods and Practice for the Networked Society. New York: Springer, 2007.